

Szoftver technológia

Feladatgyűjtemény

UML

Megoldások

Készítették:

Goldschmidt Balázs, László Zoltán, Simon Balázs

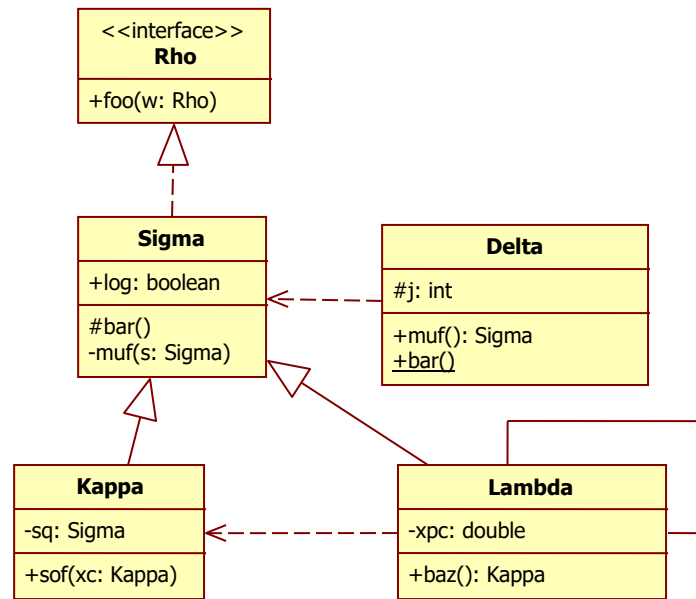
Copyright © BME IIT, 2017

# Tartalomjegyzék

1	Osztálydiagram .....	3
2	Komponensdiagram .....	46
3	Use-case diagram .....	51
4	Állapotátmenet diagram .....	58
5	Szekvenciadiagram .....	85
6	Kommunikációs diagram .....	119
7	Aktivitás diagram .....	125
8	Időzítési diagram .....	133
9	Package diagram .....	138
10	Kollaboráció .....	139
11	Rational Unified Process .....	140
12	Egyéb feladatok .....	143
12.1	Kollekciók .....	143
12.2	Konkurencia .....	144
12.3	Metaclass .....	146
12.4	Általános .....	148

# 1 Osztálydiagram

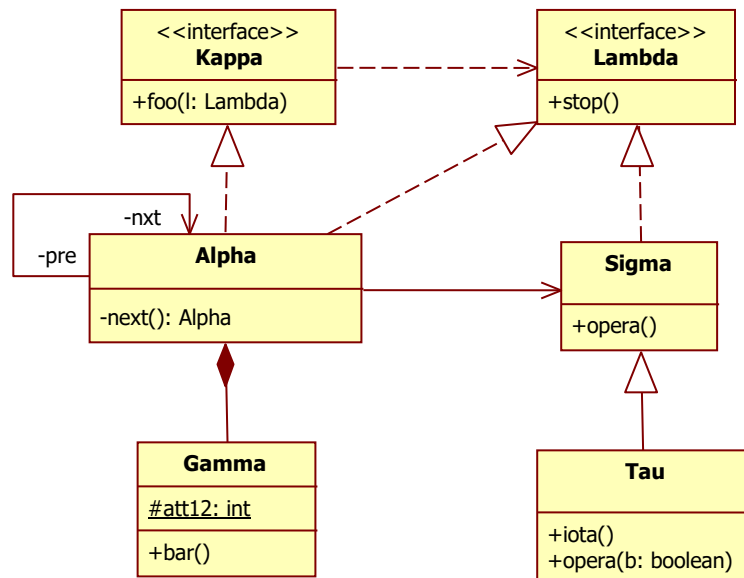
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [C] **Delta** `muf()` függvénye nem példányosíthat **Kappa** objektumot, mert **Kappa** nem függ **Delta**-tól.
- [A] **Lambda** `bar()` függvénye példányosíthat **Lambda** objektumot, mert **Lambda** a **Kappa** leszármazottja.
- [A] **Delta** `bar()` függvénye nem módosíthatja a **j** attribútum értékét, mert **j** privát.
- [C] **Kappa** `sof(xc: Kappa)` függvénye nem kaphat paraméterül **Lambda** objektumot, mert **Lambda** függ **Kappa**-tól.
- [E] **Lambda** `foo(w:Rho)` függvénye nem hívhatja meg **Kappa** `foo(w:Rho)` függvényét, mert egyiknek sincs `foo(w:Rho)` függvénye.
- [C] **Delta** `bar()` függvénye módosíthatja **Sigma** `log` attribútumának értékét, mert `log` nem statikus.
- [D] **Sigma** `bar()` függvénye nem hívhat meg minden `muf()` függvényt, mert **Sigma** nem ismeri **Delta**-ot.
- [A] **Sigma** `muf(s: Sigma)` függvénye meghívhatja a paraméterül kapott **Lambda** típusú objektum `bar()` függvényét, mert `bar()` absztrakt.

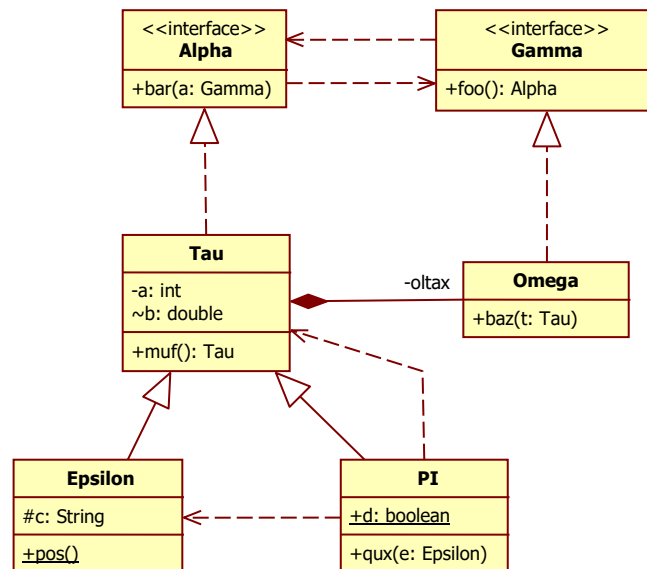
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- A - csak az első tagmondat igaz (+ -)  
 B - csak a második tagmondat igaz (- +)  
 C - mindkét tagmondat igaz, de a következtetés hamis (+ + -)  
 D - mindkét tagmondat igaz és a következtetés is helyes (+ + +)  
 E - egyik tagmondat sem igaz (- -)

- [D] Alfa **foo(l: Lambda)** metódusa kaphat paraméterül **Tau** objektumot, mert **Tau** megvalósítja a **Lambda** interfészt.
- [B] Az **Alpha** osztály **a1** példányának **next()** metódusa csak magára az **a1**-re vonatkozó referenciát tud visszaadni, mert a **next()** metódus privát.
- [B] **Alpha next()** metódusából elérhetjük **Gamma att12** attribútumát, mert **Gamma** az **Alpha** komponense.
- [C] **Alpha** meghívhatja **Sigma opera()** metódusát, mert mindketten implementálják **Lambda**-t.
- [E] **Gamma bar()** metódusából meghívhatjuk **Alpha next()** metódusát, mert a komponensből elérhetjük az őt tartalmazó osztály minden metódusát.
- [B] **Tau** helyettesíthető **Sigma**val, mert mindkettőnek van **opera()** metódusa.
- [B] **Tau opera(b: boolean)** metódusa felüldefiniálja **Sigma opera()** metódusát, mert **Tau** a **Sigma** specializációja.
- [C] **Tau stop()** metódusából meghívhatjuk az **iota()** metódust, mert az **iota()** metódus publikus.

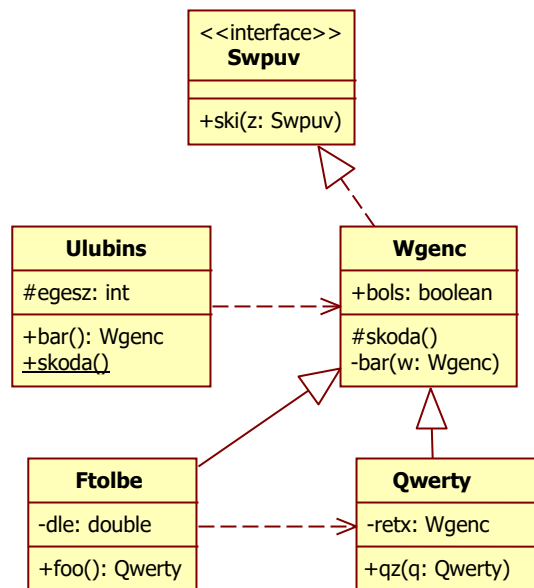
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [C] **Epsilon** **bar(a: Gamma)** függvénye nem hívhatja meg a paraméterül kapott **PI** objektum **qux(e: Epsilon)** függvényét, mert **Epsilon** nem ismeri **PI**-t.
- [B] **PI** **qux(e: Epsilon)** függvénye kaphat paraméterül **Tau** objektumot, mert **PI** ismeri **Tau**-t.
- [A] **Epsilon** **pos()** függvénye nem módosíthatja a **c** attribútum értékét, mert **Epsilon** **c** attribútuma package láthatóságú.
- [C] **Tau** **muf()** függvénye meghívhatja **Omega** **baz(t: Tau)** függvényét, mert **Omega** ismeri **Tau**-t.
- [D] **Omega** **baz(t: Tau)** függvénye módosíthatja **Tau** **b** attribútumát, mert **Tau** **b** attribútuma nem privát.
- [B] **Omega** **foo()** függvénye nem hozhat léte **Tau** objektumot, mert a **foo()** függvényt deklaráló **Gamma** interfész nem ismeri **Tau**-t.
- [B] **PI** **qux(e: Epsilon)** függvénye nem módosíthatja a **d** attribútum értékét, mert **PI** **d** attribútuma statikus.
- [C] **Epsilon** **pos()** függvénye nem módosíthatja az **a** attribútum értékét, mert **Tau** **a** attribútuma nem statikus.

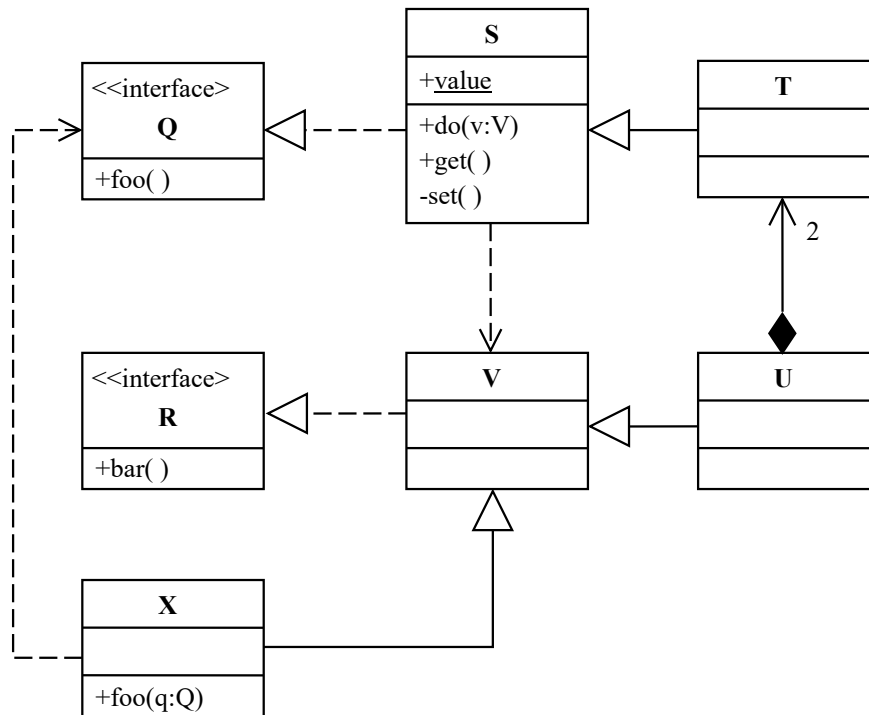
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [A] **Ftolbe** **ski(z:Swpuv)** metódusa a paraméterül kapott objektumnak csak a **ski(z:Swpuv)** metódusát hívhatja meg, mert a kapott objektumnak nem lehet más metódusa.
- [B] **Qwerty** **qz(q:Qwerty)** metódusa kaphat paraméterül **Ftolbe** objektumot, mert **Ftolbe** **Qwerty** ösének leszármazottja.
- [C] **Ulubins** **bar()** metódusa nem példányosíthat **Qwerty** objektumot, mert **Qwerty** nem függ **Ulubins**-től.
- [B] Van olyan **skoda()** metódus, amely nem módosíthatja a **bols** attribútumot, mert **bols** nem statikus.
- [E] **Ulubins** **skoda()** metódusának visszatérési értéke **Ulubins**, mert minden statikus metódus predefinit visszatérési értéke a saját osztályának egy példánya.
- [A] **Wgenc** **bar(w:Wgenc)** metódusa meghívhatja a paraméterül kapott **Ftolbe** típusú objektum **skoda()** metódusát, mert **skoda** package láthatóságú.
- [C] **Ftolbe** **foo()** metódusa példányosíthat **Ftolbe** objektumot, mert **foo()** nem statikus.
- [D] **Wgenc** **skoda()** metódusa nem hívhat meg minden **bar** nevű metódust, mert **Wgenc** nem ismeri **Ulubins**-t.

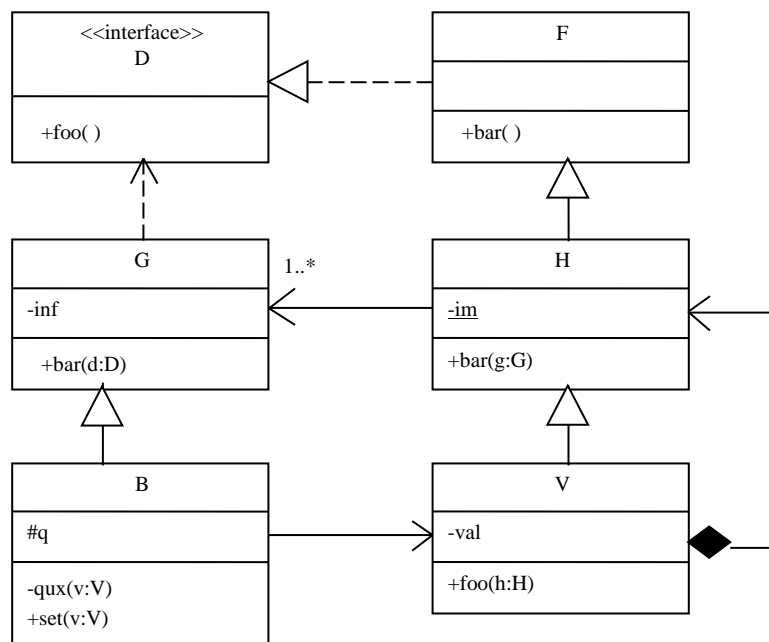
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [C] X foo(q:Q) metódusa kaphat paraméterül T-t, mert T-nek is van foo nevű metódusa.
- [B] S set() metódusa nem módosíthatja a value attribútumot, mert a láthatóságuk különböző.
- [B] V törlésekor törölni kell két T-t is, mert egy U-nak két T komponense van és U a V leszármazottja.
- [B] S meghívhatja T foo() metódusát, mert T az S leszármazottja.
- [A] X meghívhatja egy Q interfészű objektum foo() metódusát, mert X implementálja Q-t.
- [B] T létrehozhat U osztályú objektumot, mert S létrehozhat V-t és T az S-nek, U a V-nek leszármazottja.
- [B] X foo(q:Q) metódusa meghívhatja a paraméterül kapott S get() metódusát, mert S megvalósítja a Q interfészt.
- [B] X bar() metódusából meghívhatjuk egy Q interfészű objektum foo() metódusát, mert X foo(q:Q) metódusából is hívhatjuk egy Q interfészű objektum foo() metódusát.

Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !

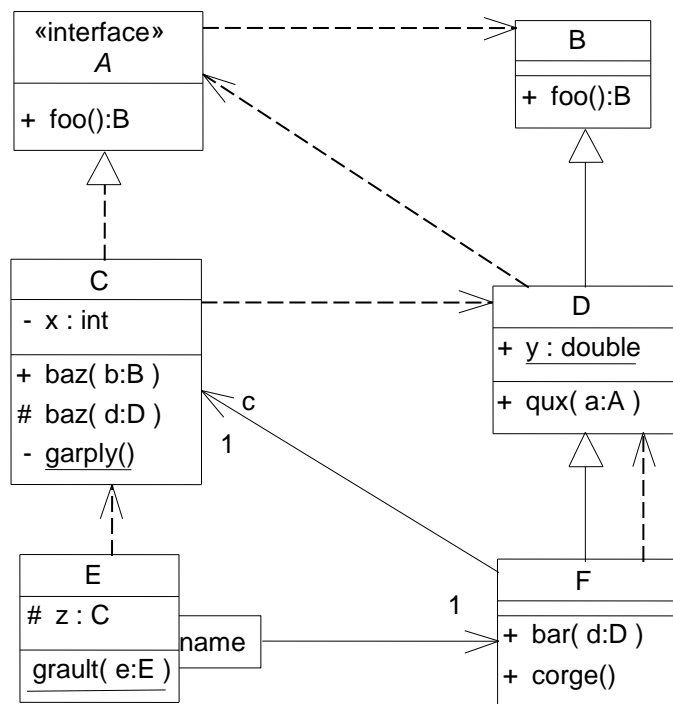


- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [E] B objektum nem hívhatja meg egy V objektum **foo()** metódusát, mert V-nek nincs ilyen szignatúrájú metódusa.
- [B] G **bar(d:D)** metódusa meghívhatja egy paraméterül kapott F objektum **bar()** metódusát, mert F megvalósítja a D interfészt.
- [C] G **bar(d:D)** metódusa kaphat paraméterül H objektumot, mert H-nak van **bar()** metódusa.
- [B] H **bar(g:G)** metódusa kaphat paraméterül V objektumot, mert V megvalósítja a D interfészt.
- [B] B **qux(v:V)** metódusa módosíthatja a paraméter **val** attribútumát, mert mind a metódus, mind az attribútum privát.
- [C] H **bar(g:G)** metódusa módosíthatja az **im** attribútumot, mert az attribútum statikus.
- [B] B **set(v:V)** metódusa nem módosíthatja a **q** attribútumot, mert a láthatóságuk különböző.
- [B] V tartalmazhat G-t, mert V tartalmazhat H-t és minden H-hoz tartozik G.



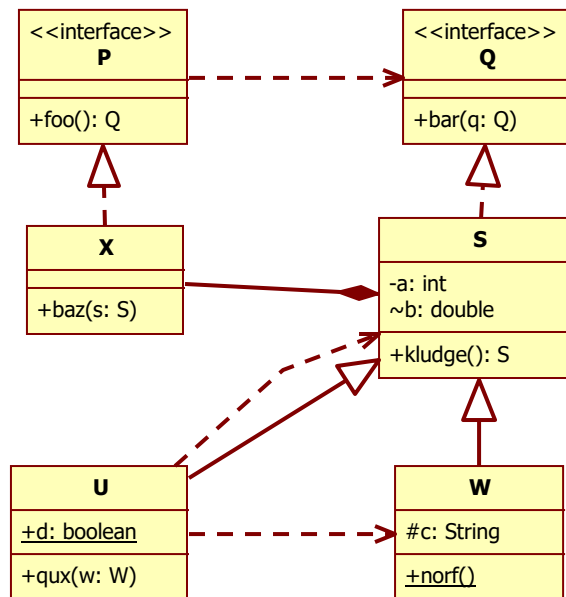
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [B] D implementálja az A interfészt, mert A és B interfésze megegyezik.
- [B] F corge() metódusa nem módosíthatja D y attribútumát, mert D y attribútuma statikus.
- [B] E grault(e:E) metódusa nem hívhatja meg e z attribútumának baz(b:B) metódusát, mert a grault() statikus.
- [E] Egy F objektum pontosan egy E objektumot ismer, mert egy E objektum pontosan egy F objektumot ismer.
- [C] C foo() metódusa példányosíthat B típusú objektumot, mert C függ D-től és D a B leszármazottja.
- [E] F bar(d:D) metódusából nem hívható meg c baz(b:B) metódusa, mert C baz(b:B) metódusa nem kaphat paraméterül D típusú objektumot.
- [B] C baz(d:D) metódusa nem hívhatja meg C garply() metódusát, mert C baz (d:D) metódusa nem statikus.
- [A] D qux(a:A) metódusa nem hívhatja meg egy paraméterül kapott C típusú objektum baz(b:B) metódusát, mert A nem helyettesíthető C-vel.

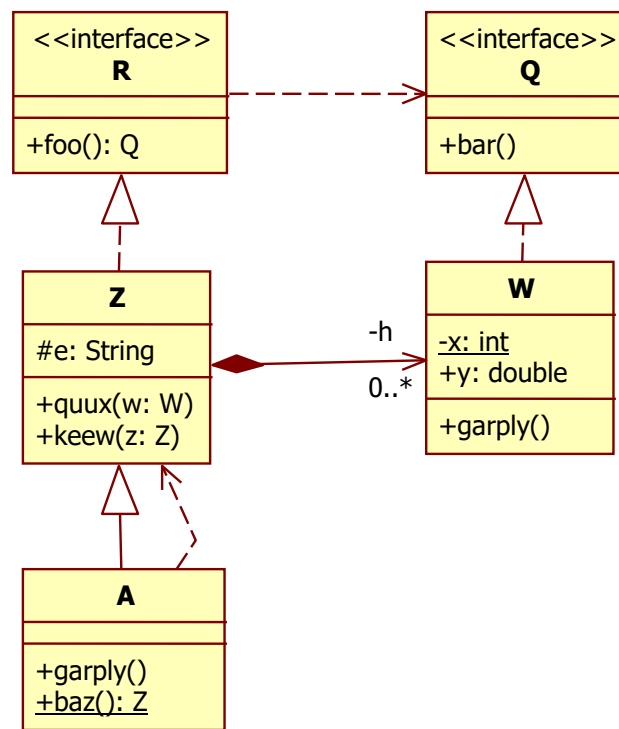
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [B] U qux() függvénye kaphat paraméterül S objektumot, mert U ismeri S-et.
- [A] W norf() függvénye nem módosíthatja a c attribútum értékét, mert W c attribútuma package láthatóságú.
- [D] X baz() függvénye módosíthatja S b attribútumát, mert S b attribútuma nem privát.
- [B] X foo() függvénye nem hozhat létre S objektumot, mert a foo() függvényt deklaráló P interfész nem ismeri S-et.
- [B] U qux() függvénye nem módosíthatja a d attribútum értékét, mert U d attribútuma statikus.
- [C] S kludge() függvénye meghívhatja X baz() függvényét, mert X ismeri S-et.
- [C] W bar() függvénye nem hívhatja meg a paraméterül kapott U objektum qux() függvényét, mert W nem ismeri U-t.
- [C] W norf() függvénye nem módosíthatja az a attribútum értékét, mert S a attribútuma nem statikus.

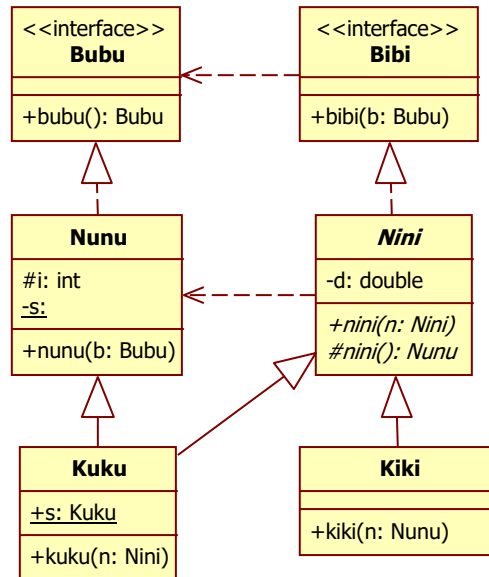
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [C] Z foo() függvénye nem hozhat létre A objektumot, mert A nem implementálja a Q interfészt.
- [C] Van olyan garply() függvény, amely nem módosíthatja W x attribútumát, mert x statikus.
- [E] A baz() függvénye nem példányosíthat W objektumot, mert W absztrakt.
- [B] Z keew(). függvénye nem kaphat paraméterül A objektumot, mert Z nem ismeri A-t.
- [D] Z quux() függvénye módosíthatja a paraméterül kapott w y attribútumát, mert y publikus..
- [B] A és Z interfésze megegyezik, mert mindkettő implementálja az R interfészt.
- [B] Nincs olyan garply() függvény, amely módosíthatja az e attribútumot, mert e nem statikus.
- [C] W bar() függvénye nem hívhatja meg A garply() függvényét, mert W nem ismeri Z-t.

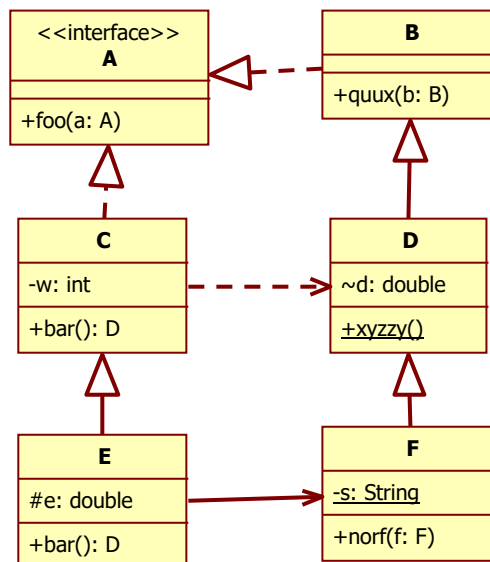
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [B] Nini bibi függvénye nem kaphat paraméterül Kuku objektumot, mert Bibi nem ismeri Kuku-t.
- [C] Van olyan s attribútum, amelyet nem módosíthat Kiki kiki függvénye, mert nem minden s attribútum publikus.
- [C] Kuku implementálja a Bubu és Bibi interfészeket, mert Java-ban megengedett az interfészek többszörös öröklése.
- [C] Nunu nunu függvénye nem példányosíthat Kiki objektumot, mert Kiki függ Nunu-tól.
- [E] Kuku nem példányosítható, mert nem implementálja az absztrakt nini függvényeket.
- [B] Kuku és Kiki interfésze megegyezik, mert mindkettő megvalósítja a Bibi interfészt.
- [E] Kiki kiki függvénye nem kaphat paraméterül Kuku objektumot, mert ha kapna, az sértené a Liskov-elvet.
- [E] Kuku bubu függvénye nem módosíthatja az i attribútum értékét, mert i statikus.

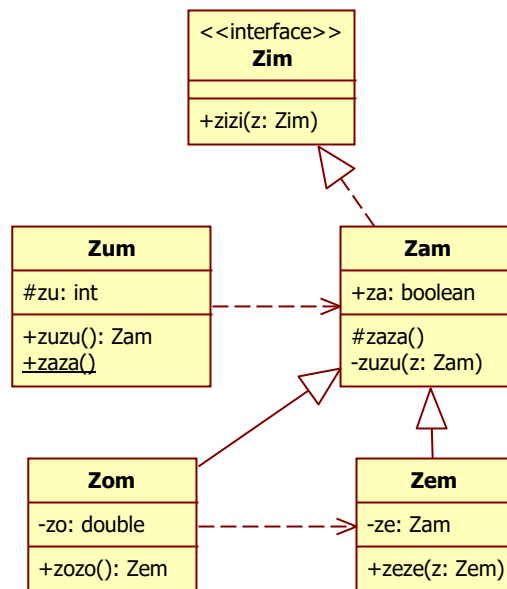
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [C] D xyzyz() függvénye nem módosíthatja a d attribútum értékét, mert d nem protected.
- [B] E bar() függvénye nem hozhat létre F objektumot, mert C bar() függvénye nem hozhat létre F objektumot.
- [B] F norf() függvénye nem módosíthatja az s attribútumot, mert F norf() függvénye nem statikus.
- [A] B nem ismeri F-et, ezért B quux() függvénye nem kaphat paraméterül F objektumot.
- [B] C és D interfésze megegyezik, mert mindkettő implementálja az A interfészt.
- [D] F norf() függvénye nem kaphat paraméterül D objektumot, mert D nem az F leszármazottja.
- [C] F foo() függvénye nem módosíthatja egy paraméterül kapott E objektum e attribútumát, mert F nem az E leszármazottja.
- [E] E tartalmaz D-t, mert C tartalmaz D-t.

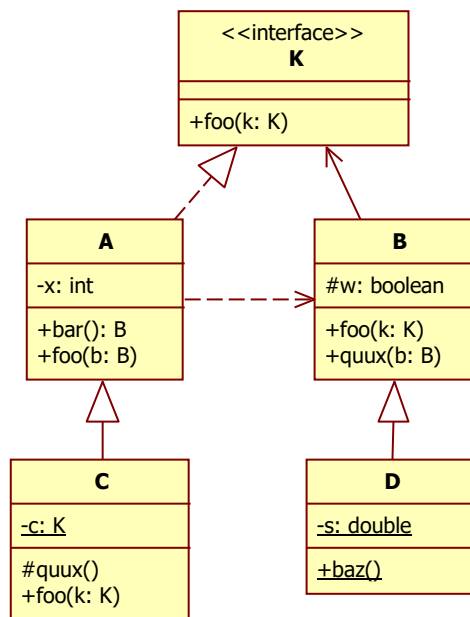
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |  |         |
|--|---------|
| <b>A</b> - csak az első tagmondat igaz                         | (+ -)   |
| <b>B</b> - csak a második tagmondat igaz                       | (- +)   |
| <b>C</b> - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| <b>D</b> - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| <b>E</b> - egyik tagmondat sem igaz                            | (- -)   |

- [B]** Van olyan **zaza** függvény, amely nem módosíthatja a **za** attribútumot, mert **za** nem statikus.
- [C]** **Zum zuzu** függvénye nem példányosíthat **Zem** objektumot, mert **Zem** nem függ **Zum**-tól.
- [A]** **Zom zozo** függvénye példányosíthat **Zom** objektumot, mert **Zom** a **Zem** leszármazottja.
- [A]** **Zum zaza** függvénye nem módosíthatja a **zu** attribútum értékét, mert **zu** privát.
- [E]** **Zom zizi** függvénye nem hívhatja meg **Zem zizi** függvényét, mert egyiknek sincs **zizi** függvénye.
- [C]** **Zem zeze** függvénye nem kaphat paraméterül **Zom** objektumot, mert **Zom** függ **Zem**-től.
- [A]** **Zam zuzu** függvénye meghívhatja a paraméterül kapott **Zom** típusú objektum **zaza** függvényét, mert **zaza** absztrakt.
- [D]** **Zam zaza** függvénye nem hívhat meg minden **zuzu** függvényt, mert **Zam** nem ismeri **Zum**-ot.

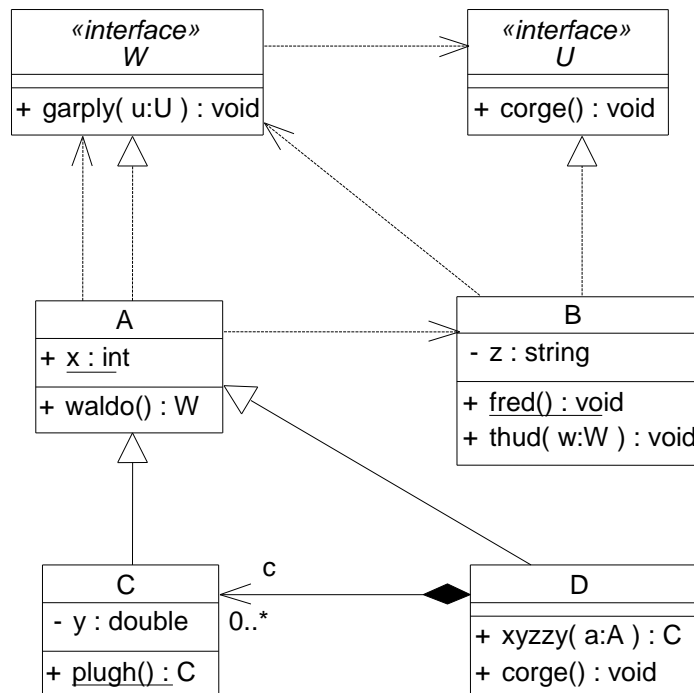
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |  |         |
|--|---------|
| <b>A</b> - csak az első tagmondat igaz                         | (+ -)   |
| <b>B</b> - csak a második tagmondat igaz                       | (- +)   |
| <b>C</b> - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| <b>D</b> - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| <b>E</b> - egyik tagmondat sem igaz                            | (- -)   |

- [A] C egyetlen függvénye sem módosíthatja egy paraméterül kapott B típusú objektum w attribútumát, mert C nem függ B-től.
- [A] D baz függvénye nem módosíthatja B w attribútumát, mert w statikus.
- [D] Van olyan foo függvény, amely nem kaphat paraméterül B típusú objektumot, mert B nem implementálja a K interfészt.
- [C] A bar függvénye nem példányosíthat D típusú objektumot, mert D nem függ A-tól.
- [B] C quux függvénye nem módosíthatja a c attribútum értékét, mert quux nem privát.
- [B] D foo függvénye nem hívhatja meg a paraméterül kapott C objektum foo(k:K) függvényét, mert D nem ismeri C-t.
- [D] minden D-ben deklarált függvény módosíthatja az s attribútumot, mert s statikus.
- [C] C és D interfésze különbözik, mert D nem implementálja a K interfészt.
- [C] D baz metódusa nem módosíthatja B w attribútumát, mert baz statikus.

Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !

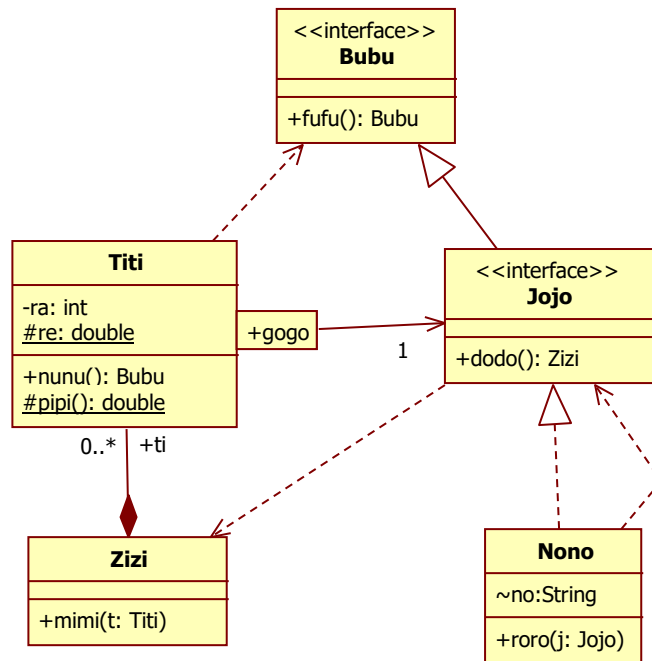


- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [B] D törlésekor legalább egy C objektumot is törölni kell, mert D tartalmaz C-t.
- [A] U interfésze részhalmaza D interfészének, ezért D megvalósítja az U interfészt.
- [E] C nem függ U-tól, mert C ősszánya (A) sem függ U-tól.
- [B] A waldo függvénye nem példányosíthat B típusú objektumot, mert B nem implementálja a W interfészt.
- [E] C plugh függvénye nem módosíthatja A x attribútumát, mert A x attribútuma protected.
- [B] D xyzy függvénye visszaadhatja eredményként a paraméterként kapott a objektumot, mert C az A leszármazottja.
- [C] B fred függvénye nem módosíthatja a z attribútum értékét, mert B z attribútuma nem protected.
- [E] B thud függvénye meghívhatja egy paraméterül kapott C típusú objektum plugh függvényét, mert C plugh függvénye virtuális.



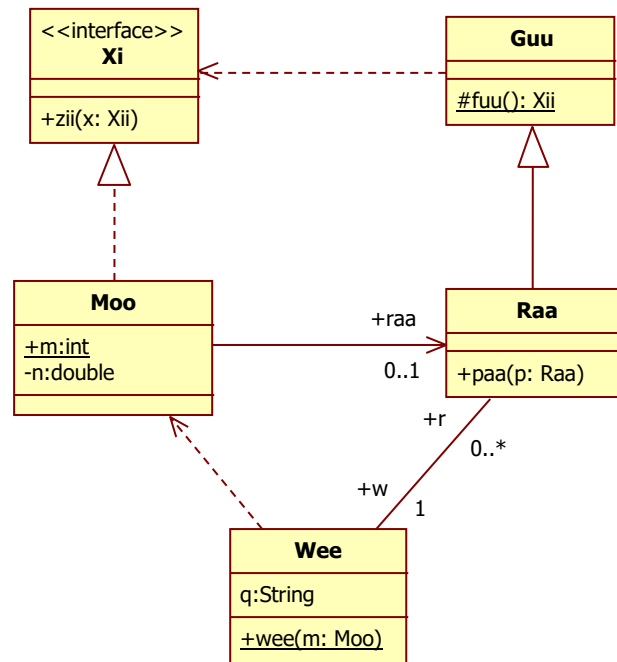
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- A - csak az első tagmondat igaz (+ -)  
 B - csak a második tagmondat igaz (- +)  
 C - mindkét tagmondat igaz, de a következtetés hamis (+ + -)  
 D - mindkét tagmondat igaz és a következtetés is helyes (+ + +)  
 E - egyik tagmondat sem igaz (- -)

- [C] **Zizi mimi(t:Titi)** függvénye nem hívhatja meg a **t** paraméter **nunu()** függvénye által visszaadott **Jojo** interfészű objektum **dodo()** függvényét, mert az sértené a Demeter-törvényt.
- [A] **Titi pipi()** függvénye nem szorozhatja össze a **ra** és **re** attribútum értékét, mert privát változóhoz csak privát függvény férhet hozzá.
- [B] **Zizi** a **ti.nunu()** hívás eredményén hívhat **roro(j:Jojo)** metódust, mert **Nono** implementálja a **Bubu** interfészt.
- [B] Egy **Titi** objektum csak egy **Jojo** interfészű objektumot ismer, mert közöttük lévő asszociáció **Jojo** oldalán 1-es számosság szerepel.
- [E] **Nono roro(j:Jojo)** függvénye kaphat paraméterül **Bubu** interfészű objektumot, mert **Jojo** implementálja a **Bubu** interfészt.
- [E] **Nono dodo()** függvénye nem hozhat létre **Zizi** objektumot, mert **Nono** nem ismeri **Zizi**-t.
- [B] **Zizi** függ **Bubu**-tól, mert **Titi** függ **Bubu**-tól.
- [B] **Nono dodo()** függvénye nem módosíthatja a **no** attribútum értékét, mert Javában a **String** típus immutábilis.

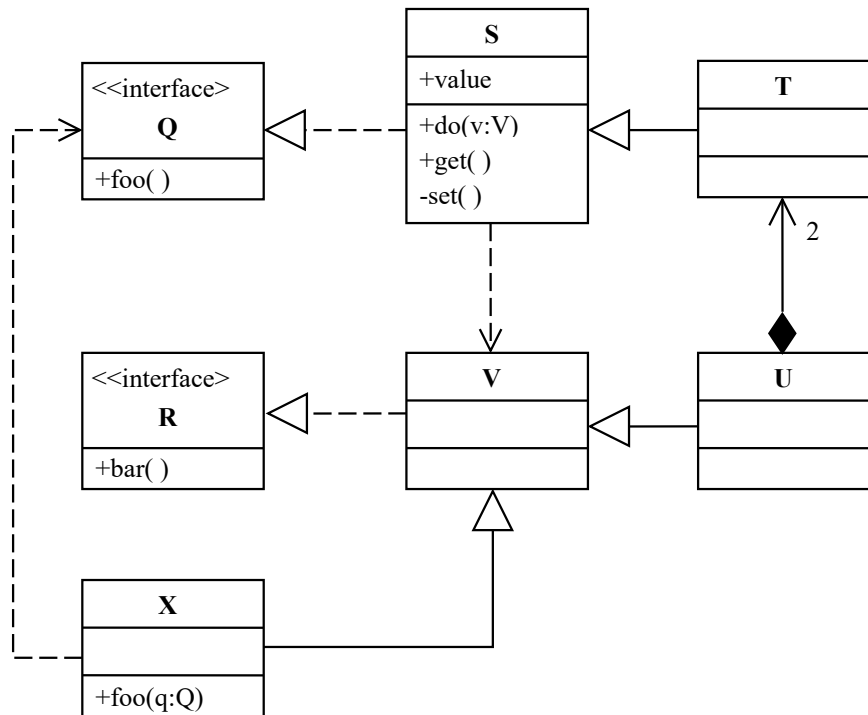
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [B] Raa nem függ Xii-től, mert Guu nem implementálja a Xii interfészt.
- [E] Raa paa(p:Raa) függvénye kaphat paraméterül Guu objektumot, mert függvényparaméterként Raa helyettesíthető Guu-val.
- [C] Raa paa(p:Raa) függvénye nem módosíthatja Moo m attribútumát, mert Moo m attribútuma statikus.
- [B] Moo zii(x:Xii) függvénye nem szorozhatja össze az m és n attribútumok értékeit, mert az n attribútum privát.
- [E] Raa paa(p:Raa) függvénye nem hívhatja meg a fuu():Xii függvényt, mert az sértene a Pauli-elvet.
- [A] Wee wee(m:Moo) függvénye nem módosíthatja a q attribútumot, mert q package láthatóságú.
- [C] Wee wee(m:Moo) függvénye meghívhatja az m paraméter zii(x:Xii) függvényét, mert az nem sérti a Demeter-törvényt.
- [D] Moo és Xii interfésze megegyezik, mert Moo nem definiál újabb függvényt.

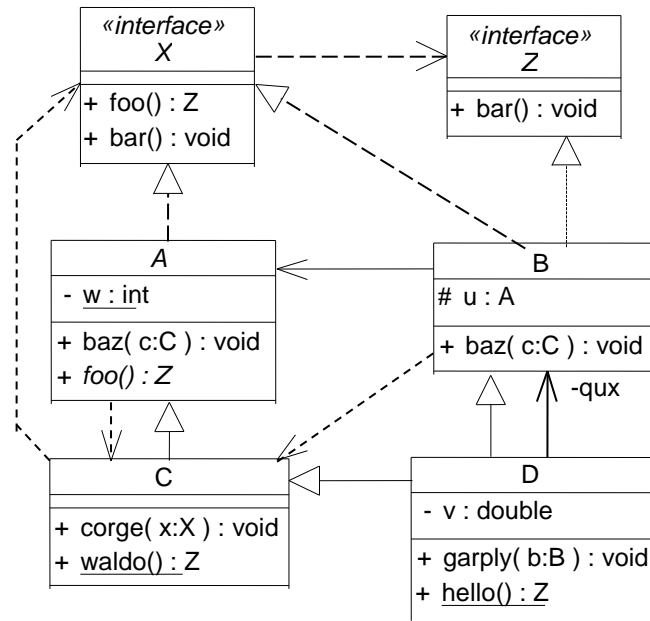
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [B] T létrehozhat U osztályú objektumot, mert S létrehozhat V-t és T az S-nek, U a V-nek leszármazottja.
- [C] X foo(q:Q) metódusa kaphat paraméterül T-t, mert T-nek is van foo() metódusa.
- [B] X foo(q:Q) metódusa meghívhatja a paraméterül kapott S get() metódusát, mert S megvalósítja a Q interfészt.
- [B] V törlésekor törölni kell két T-t is, mert egy U-nak két T komponense van és U a V leszármazottja.
- [A] T nem függ U-tól, mert T nem függ V-től sem.
- [A] X meghívhatja egy Q interfészű objektum foo() metódusát, mert X implementálja Q-t.
- [B] X bar() metódusából meghívhatjuk egy Q interfészű objektum foo() metódusát, mert X foo(q:Q) metódusából is hívhatjuk egy Q interfészű objektum foo() metódusát.
- [B] S set() metódusa nem módosíthatja a value attribútumot, mert a láthatóságuk különböző.

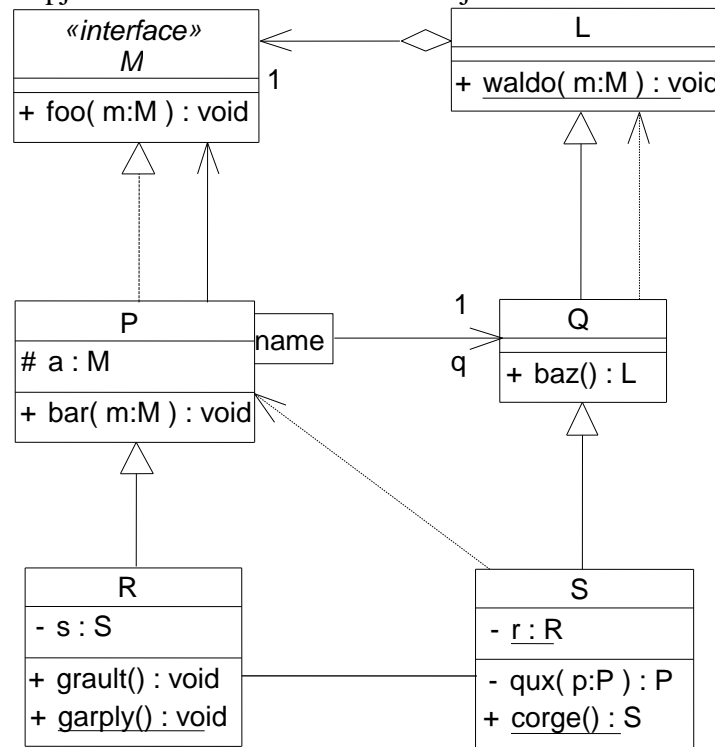
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- A - csak az első tagmondat igaz (+ -)  
 B - csak a második tagmondat igaz (- +)  
 C - mindkét tagmondat igaz, de a következtetés hamis (+ + -)  
 D - mindkét tagmondat igaz és a következtetés is helyes (+ + +)  
 E - egyik tagmondat sem igaz (- -)

- [E] D garply metódusa nem módosíthatja a b paraméter u attribútumát, mert protected attribútumhoz csak privát és protected metódusok férhetnek hozzá.
- [A] C corge metódusa kaphat paraméterül D típusú objektumot, ezért a metódus meghívhatja a kapott objektum garply metódusát.
- [B] D garply metódusa kaphat paraméterül A típusú objektumot, mert A és B interfésze megegyezik.
- [B] C waldo metódusa virtuális, ezért a B osztály baz függvénye egy paraméterül kapott D típusú objektumon meghívhatja a waldo metódust.
- [B] A baz metódusa nem módosíthatja A w attribútumát, mert A baz metódusa nem statikus.
- [A] C-nek van bar metódusa, ezért C implementálja a Z interfészt.
- [C] D hello metódusa nem módosíthatja D v attribútumát, mert D v attribútuma privát.
- [B] B baz metódusa nem hívhatja meg B u attribútumának foo metódusát, mert az A osztály foo metódusa absztrakt.

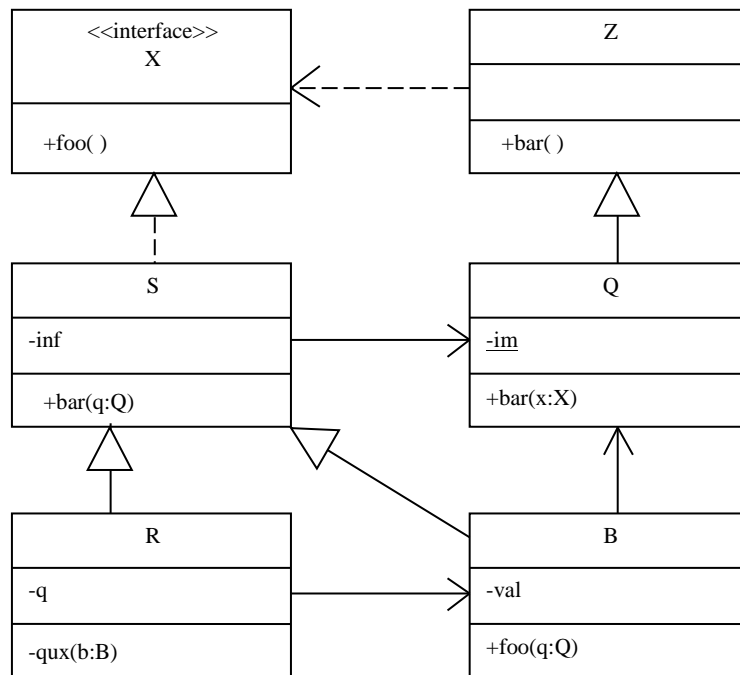
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [B] R garply metódusa meghívhatja az s attribútum corge metódusát, mert a garply és a corge metódus is statikus.
- [B] P nem hívhat Q osztályon waldo függvényt, mert L waldo függvénye nem virtuális.
- [A] Q függ M-től, mert Q ismeri P-t.
- [C] Az M típus közvetlenül nem példányosítható, ezért R grault metódusa nem hívhatja meg P a attribútumának foo metódusát.
- [B] Egy P típusú objektum pontosan egy Q típusú objektumot ismer, ezért P függ Q-től.
- [A] S az L leszármazottja, ezért Q baz függvénye példányosíthat S típusú objektumot.
- [D] S qux függvénye nem módosíthatja az r attribútum s attribútumát, mert R s attribútuma privát.
- [D] S ismeri M-et, mert az S osztály L őse függ M-től.

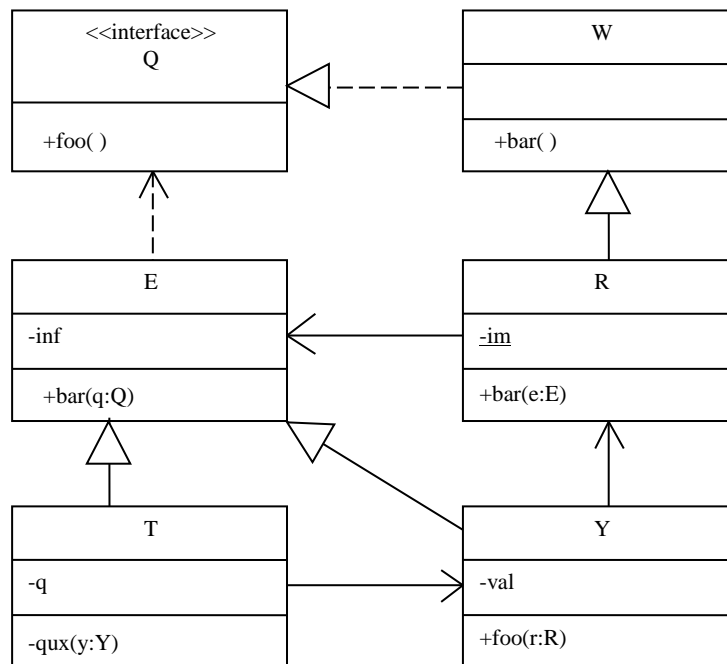
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [B] R helyettesíthető B-vel, mert R függ B-től
- [E] Q meghívhatja S **bar(q:Q)** metódusát, mert mindketten megvalósítják az X interfészt.
- [E] Q helyettesíthető S-sel, mert S a Q leszármazottja
- [E] B interfésze tartalmazza a **bar(x:X)** metódust, mert a metódus statikus.
- [B] B **foo(q:Q)** metódusa nem látja saját **val** attribútumának értékét, mert az attribútum privát.
- [C] Q **bar()** metódusa nem módosíthatja az **im** attribútumot, ezért az attribútum statikus..
- [A] Q nem implementálja a **foo()** metódust, ezért nem függ az X interfésztől.
- [A] B átvihető paraméterül Q **bar(x:X)** metódusának, mert Q és S interfésze megegyezik.
- [C] S helyettesíthető B-vel, mert B megvalósítja az X interfészt
- [A] R átvihető paraméterül Q **bar(x:X)** metódusának, mert Q és S interfésze megegyezik.

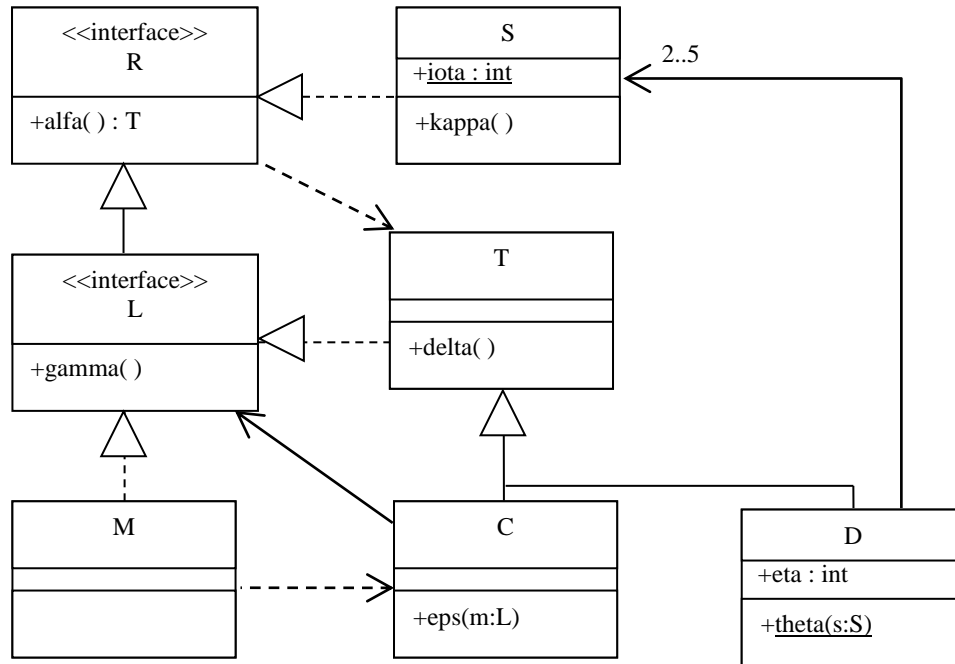
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |  |         |
|--|---------|
| <b>A</b> - csak az első tagmondat igaz                         | (+ -)   |
| <b>B</b> - csak a második tagmondat igaz                       | (- +)   |
| <b>C</b> - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| <b>D</b> - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| <b>E</b> - egyik tagmondat sem igaz                            | (- -)   |

- [C]** **Y** `bar(q:Q)` metódusa kaphat paraméterül **R** objektumot, mert **Y** függ **R**-től.
- [B]** **T** `qux(y:Y)` metódusa módosíthatja a paraméter **val** attribútumát, mert a metódus privát.
- [E]** **E** bárhol helyettesíthető **R**-rel, mert azonos az interfészük.
- [C]** **Y** `foo(r:R)` metódusa nem módosíthatja a paraméter **im** attribútumát, mert az attribútum statikus.
- [E]** **E** `bar(q:Q)` metódusa kaphat **E** objektumot paraméterül, mert az **E** megvalósítja a **Q** interfészt.
- [E]** **E** `bar(q:Q)` metódusa nem hívhatja meg egy paraméterül kapott **W** `foo()` metódusát, mert **W**-nek nincs ilyen szignatúrájú metódusa.
- [B]** **R** `bar(e:E)` metódusa nem kaphat paraméterül **Y** objektumot, mert az **Y**-**R** asszociációban csak **Y** hívhatja **R**-t.
- [B]** **R** nem valósítja meg a **Q** interfészt, mert van olyan szignatúrájú metódusa, ami nem szerepel a **Q** metódusai között.

Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !

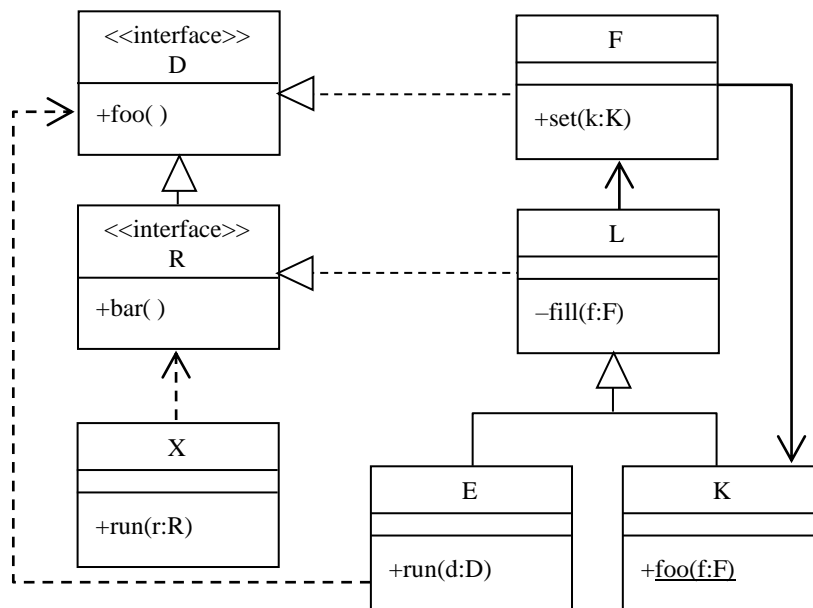


- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [A] M **alfa():T** metódusa visszaadhat C objektumot, mert C függ M-től.
- [C] D **theta(s:S)** metódusa nem kaphat paraméterül C objektumot, mert S és C is megvalósítja az R interfészt.
- [E] C **eps(m:L)** metódusa nem hívhatja meg a paraméter **gamma()** metódusát, mert az utóbbi metódus protected láthatóságú.
- [B] D **theta(s:S)** metódusa nem módosíthatja a paraméter **iota** attribútumát, mert a **theta(s:S)** statikus.
- [E] S nem valósítja meg az **alfa():T** szignatúrájú metódust, mert S nem függ T-től.
- [B] D **theta(s:S)** metódusa legfeljebb 5-ször hívható meg, mert D objektum legfeljebb 5 S-sel állhat asszociációban.
- [B] M bárhol helyettesíthető C-vel, mert mindketten megvalósítják az R interfészt.
- [D] T osztálynak van **alfa():T** szignatúrájú metódusa, mert T megvalósítja az R interfészt.
- [C] D **theta(s:S)** metódusa módosíthatja a paraméter **iota** attribútumát, mert a **theta(s:S)** statikus.
- [E] T osztálynak nincs **alfa():T** szignatúrájú metódusa, mert T nem valósítja meg az R interfészt.
- [B] D **theta(s:S)** metódusa kaphat paraméterül C objektumot, mert S és C is megvalósítja az R interfészt.



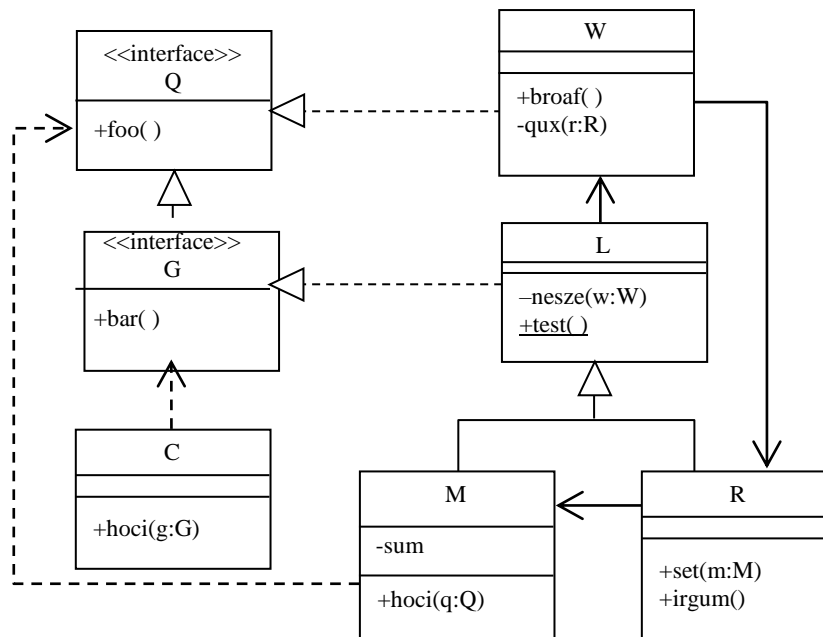
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [B] E bárhol helyettesíthető K-val, mert van közös ősük.
- [B] L bárhol helyettesíthető F-fel, mert mindketten megvalósítják a D interfészt.
- [B] L nem helyettesíthető E-vel, mert L-nek van privát metódusa.
- [B] F set(k:K) metódusa meghívhatja egy paraméterül kapott K fill(f:F) metódusát, mert K függ F-től.
- [B] X run(r:R) metódusa kaphat paraméterül F osztályú objektumot, mert X függ R-től.
- [E] K-nak nincs foo() szignatúrájú metódusa, mert K-t nem lehet példányosítani.
- [B] X run(r:R) metódusa nem kaphat paraméterül K objektumot, mert K-nak van statikus metódusa.
- [B] K foo(f:F) metódusa nem hívhatja meg a paraméter foo() metódusát, mert az utóbbi metódus nem statikus.
- [E] L bárhol helyettesíthető F-fel, mert mindketten megvalósítják az R interfészt.
- [C] F set(k:K) metódusa nem hívhatja meg egy paraméterül kapott K fill(f:F) metódusát, mert K függ F-től.

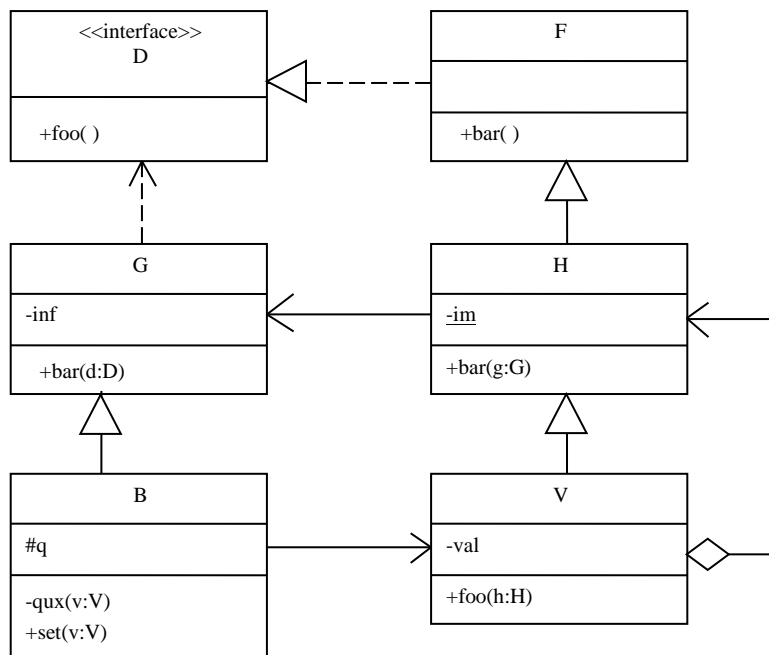
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [E] M hoci(q:Q) függvénye meghívhatja egy paraméterül kapott W broaf() metódusát, mert a broáf metódus statikus.
- [B] R set(m:M) metódusa kaphat paraméterül L objektumot, mert M az L leszármazottja.
- [B] L nesze(w:W) metódusa meghívhatja a paraméterül kapott objektum qux(r:R) metódusát, mert mindkét metódus privát.
- [B] W bárhol helyettesíthető L-lel, mert mindketten megvalósítják a Q interfészt.
- [E] R-nek nincs foo() szignatúrájú metódusa, mert nem valósítja meg a G interfészt.
- [A] C hoci(g:G) metódusa kaphat paraméterül M objektumot, mert M hoci(q:Q) metódusa is kaphat paraméterül C-t.
- [B] L nesze(w:W) metódusa nem hívhatja meg a test() metódust, mert a test() statikus.
- [A] W qux(r:R) metódusából bármikor meghívható a paraméter irgum() metódusa, mert a két osztály nem függ egymástól.

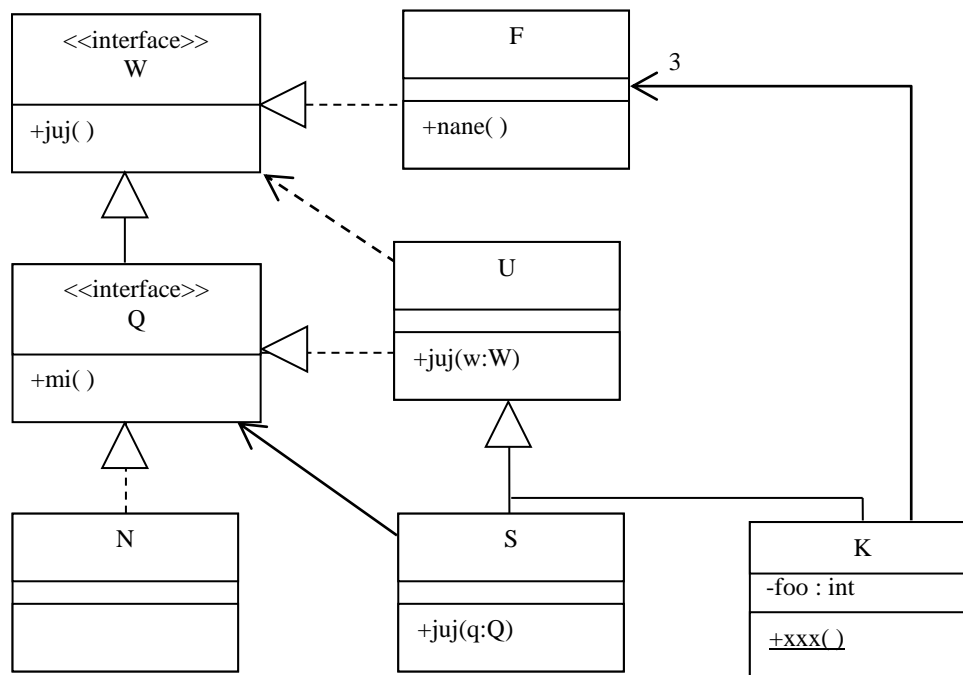
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |  |         |
|--|---------|
| <b>A</b> - csak az első tagmondat igaz                         | (+ -)   |
| <b>B</b> - csak a második tagmondat igaz                       | (- +)   |
| <b>C</b> - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| <b>D</b> - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| <b>E</b> - egyik tagmondat sem igaz                            | (- -)   |

- [E]** **G** `bar(d:D)` metódusa kaphat paraméterül **B** objektumot, mert **G** a **B** leszármazottja
- [B]** **H** `bar(g:G)` metódusa kaphat paraméterül **V** objektumot, mert **V** megvalósítja a **D** interfészt.
- [B]** **B** `qux(v:V)` metódusa módosíthatja a paraméter **val** attribútumát, mert mind a metódus, mind az attribútum privát.
- [E]** **H** `bar(g:G)` metódusa nem módosíthatja az **im** attribútumot, mert az attribútum konstans.
- [E]** **B** objektum nem hívhatja meg egy **V** objektum `foo()` metódusát, mert **V**-nek nincs ilyen szignatúrájú metódusa.
- [E]** **G** `bar(d:D)` metódusa meghívhatja egy paraméterül kapott **F** objektum `bar()` metódusát, mert a két metódus azonos szignatúrájú
- [B]** **B** `set(v:V)` metódusa nem módosíthatja a **q** attribútumot, mert a láthatóságuk különböző.
- [E]** **B**-nek van `foo()` szignatúrájú metódusa, mert **B** megvalósítja a **D** interfészt

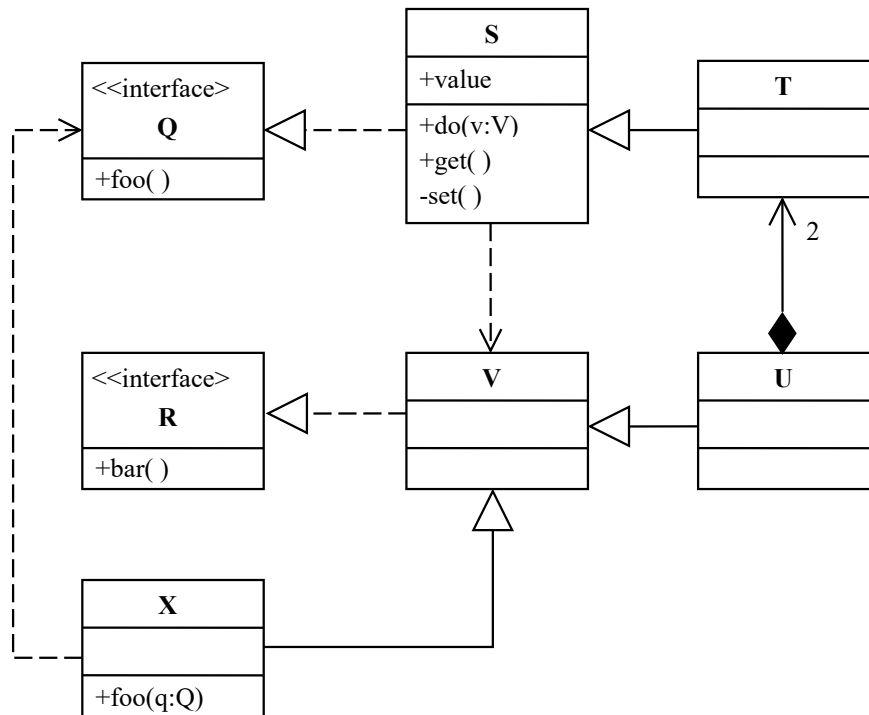
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [B] K helyettesíthető S-sel, mert közös ősük U.
- [A] K `juj(w:W)` metódusa kaphat paraméterül S-t, mert K az F leszármazottja.
- [B] K `xxx()` metódusa módosíthatja bármely K objektum `foo` attribútumát, mert a metódus statikus.
- [B] F nem implementálja a `juj()` metódust, mert nem U leszármazottja.
- [C] S `juj(q:Q)` metódusában meghívható egy paraméterül kapott N objektum `mi()` metódusa, mert N megvalósítja a W interfészt.
- [E] S-nek nincs `juj(w:W)` metódusa, mert a `juj(q:Q)` metódusnak ugyanaz a szignatúrája.
- [E] F helyettesíthető U-val, mert K mindkettejük leszármazottja.
- [E] U `juj(w:W)` metódusából meghívhatjuk egy paraméterül kapott F `nane()` metódusát, mert F megvalósítja a Q interfészt.

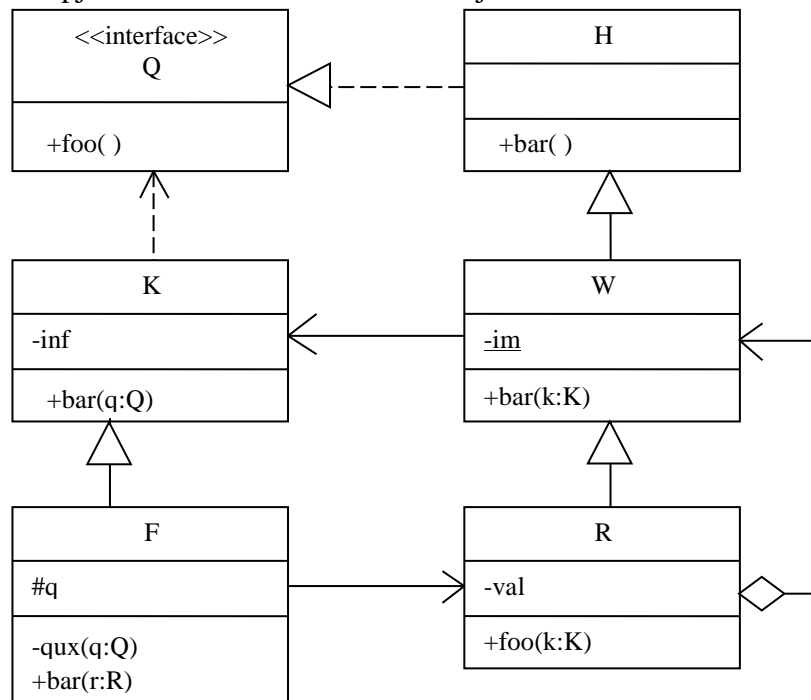
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [A] S létrehozhat V osztályú objektumot, mert V függ az S-től.
- [D] X foo(q:Q) metódusa kaphat paraméterül T-t, mert T megvalósítja a Q interfészt.
- [B] X foo(q:Q) metódusa meghívhatja a paraméterül kapott S get() metódusát, mert S megvalósítja a Q interfészt.
- [D] T-ből legalább kétszer annyi példány van, mint U-ból, mert egy T példány nem tartozhat két U-hoz.
- [B] T meghívhatja U bar() metódusát, mert U-nak van bar () metódusa.
- [A] X meghívhatja egy Q interfészű objektum foo() metódusát, mert X implementálja Q-t
- [C] V helyettesíthető U-val, mert mindketten megvalósítják az R interfészt
- [B] S set() metódusa nem módosíthatja a value attribútumot, mert a láthatóságuk különböző

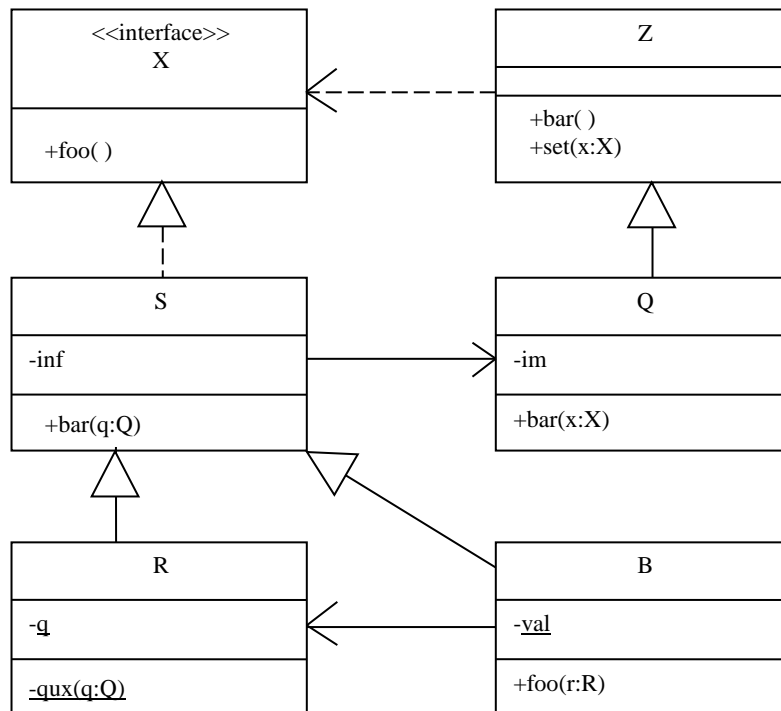
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [D] H bármikor helyettesíthető R-rel, mert R a H leszármazottja.
- [A] W nem módosíthatja K inf attribútumát, mert az attribútum protected.
- [E] F bar(r:R) metódusa nem hívhatja meg a qux(q:Q) metódust, mert az utóbbi statikus.
- [B] F qux(q:Q) meghívhatja a bar(r:R) metódust a q paraméterrel, mert az R megvalósítja a Q interfészt.
- [E] Az ábrán szereplő összes egy-paraméteres bar metódus kaphat R objektumot paraméterül, mert R megvalósítja az összes említett metódust.
- [E] W bar(k:K) metódusa nem módosíthatja az osztály im attribútumát, mert az attribútum konstans.
- [C] W rendelkezik foo() szignatúrájú metódussal, mert függ K-tól.
- [D] Egy F objektum meghívhatja saját magával mint paraméterrel egy R foo(k:K) metódusát, mert F a K leszármazottja.

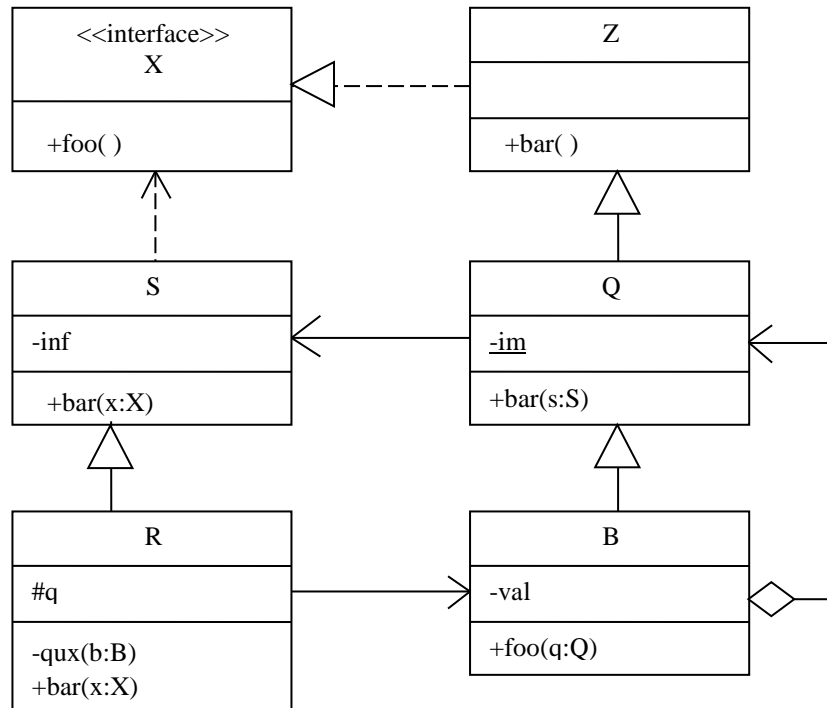
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [E]** Q bárhol helyettesíthető S-sel, mert S a Q leszármazottja.
- [A]** R qux(q:Q) metódusa nem kaphat paraméterül Z objektumot, mert a metódus absztrakt
- [E]** B foo(r:R) metódusa nem hívhat meg a paraméterül kapott objektumon foo() metódust, mert R-nek nincs ilyen metódusa
- [E]** S bar(q:Q) metódusa nem módosíthatja az S inf attribútumát, mert az attribútum konstans.
- [E]** S bar(q:Q) metódusa nem hívhatja meg a paraméterül kapott objektum bar() metódusát, mert a Q osztálynak nincs ilyen szignatúrájú metódusa.
- [B]** Z set(x:X) metódusa nem kaphat paraméterül B objektumot, mert B megvalósítja az X interfészt.
- [B]** B módosíthatja egy Q objektum im attribútumát, mert S függ Q-tól.
- [B]** R és B bárhol felcserélhetők, mert közös az ősük.

Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !

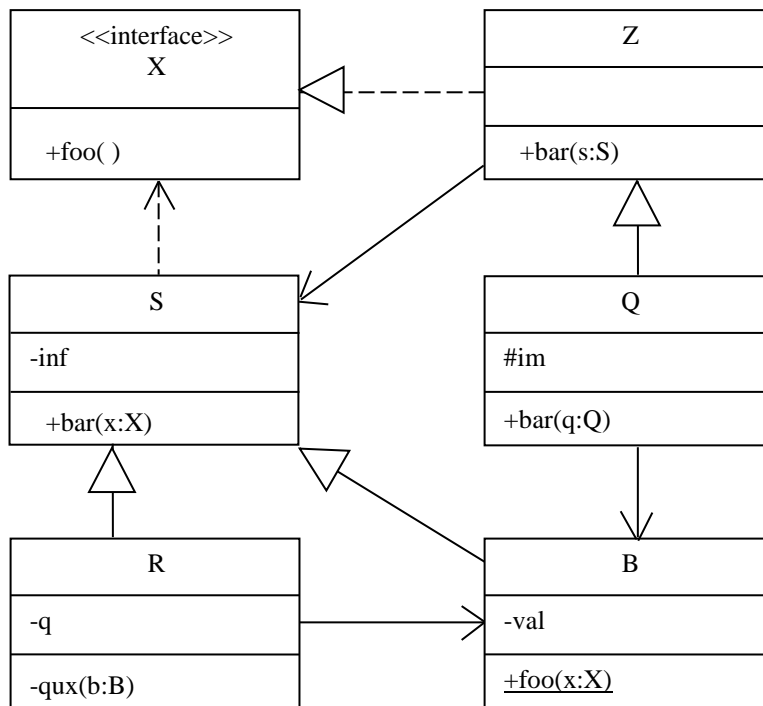


- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [E] R helyettesíthető B-vel, mert mindketten megvalósítják az X interfészt.
- [D] Z helyettesíthető B-vel, mert B a Z leszármazottja.
- [B] Q bar(s:S) metódusa nem módosíthatja Q im attribútumát, mert az attribútum statikus.
- [E] R qux(b:B) metódusa nem hívhat meg a paraméteren foo( ) metódust, mert B nem implementálja az X interfészt.
- [B] S bar(x:X) metódusa meghívhatja egy paraméterül kapott Z bar( ) metódusát, mert Z megvalósítja az X interfészt.
- [E] R qux(b:B) metódusa nem módosíthatja a q attribútumot, mert az attribútum privát.
- [B] B bar(s:S) metódusa nem kaphat paraméterül R objektumot, mert B nem ismeri az R osztályt.
- [B] S bar(x:X) metódusa kaphat paraméterül R objektumot, mert R függ X-től.



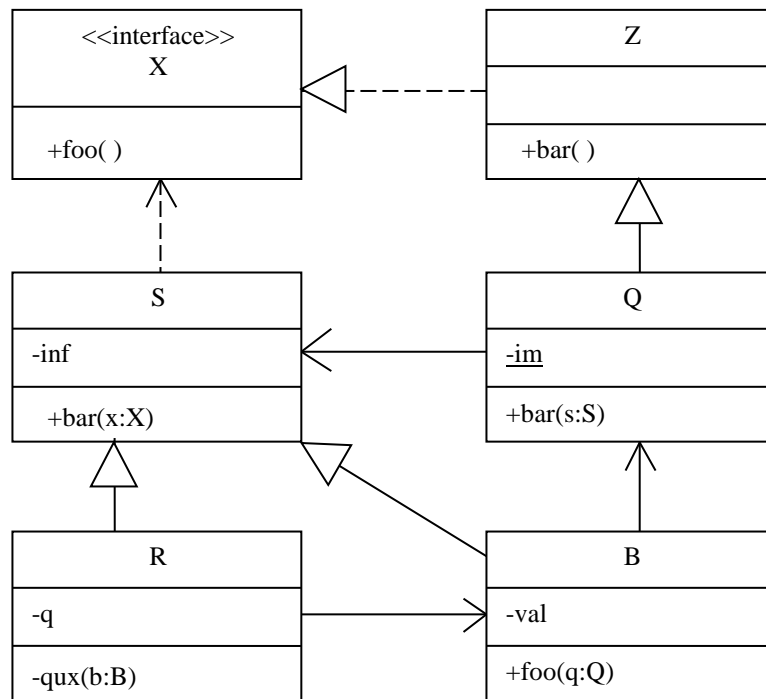
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |  |         |
|--|---------|
| <b>A</b> - csak az első tagmondat igaz                         | (+ -)   |
| <b>B</b> - csak a második tagmondat igaz                       | (- +)   |
| <b>C</b> - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| <b>D</b> - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| <b>E</b> - egyik tagmondat sem igaz                            | (- -)   |

- [E]** R helyettesíthető S-sel, mert S az R leszármazottja
- [E]** R helyettesíthető B-vel, mert mindketten megvalósítják az X interfészt
- [A]** R átadható paraméterül Q **bar (s : S)** metódusának, mert Q és S interfésze megegyezik.
- [D]** B **foo (x : X)** metódusa nem látja a **val** attribútum értékét, mert az attribútum nem statikus.
- [A]** Q meghívhatja S **bar (x : X)** metódusát, mert mindketten megvalósítják az X interfészt.
- [B]** B interfésze tartalmaz **qux (b : B)** metódust, mert B-nek van R-rel közös őse.
- [A]** Q **bar (s : S)** metódusa nem módosíthatja az **im** attribútumot, mert az attribútum privát.
- [B]** B meghívhatja R **qux (b : B)** metódusát, mert a metódus paramétere B osztályú.

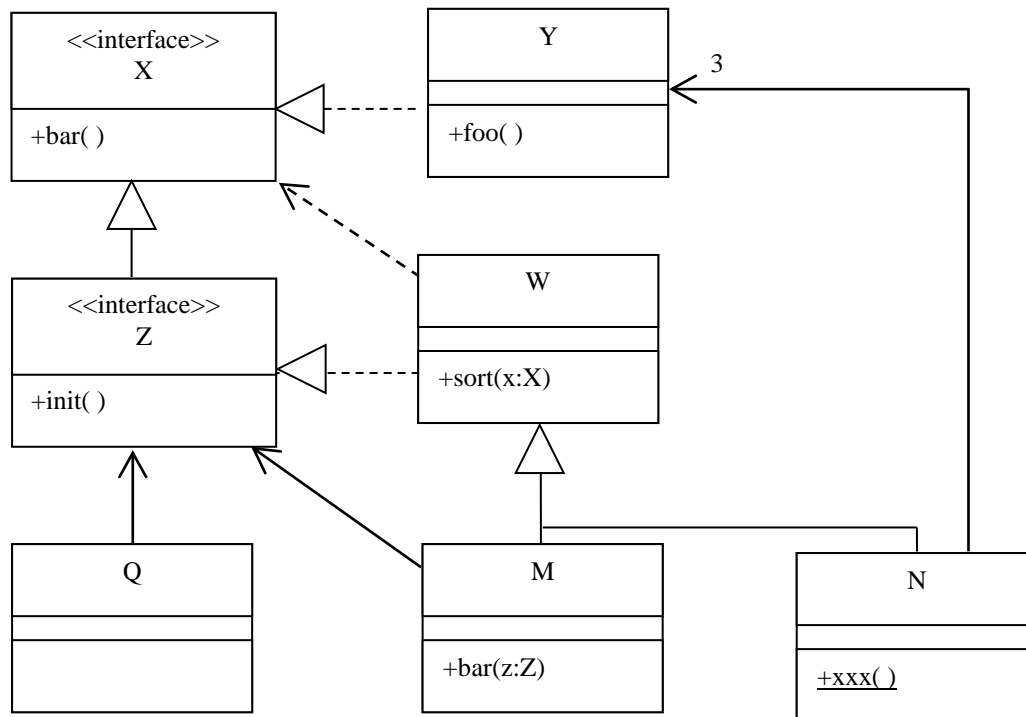
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |  |         |
|--|---------|
| <b>A</b> - csak az első tagmondat igaz                         | (+ -)   |
| <b>B</b> - csak a második tagmondat igaz                       | (- +)   |
| <b>C</b> - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| <b>D</b> - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| <b>E</b> - egyik tagmondat sem igaz                            | (- -)   |

- [E]** S helyettesíthető Q-val, mert Q az S leszármazottja
- [A]** S helyettesíthető B-vel, mert B megvalósítja az X interfészt
- [A]** R átadható paraméterül Q **bar(s:S)** metódusának, mert Q és S interfésze megegyezik.
- [B]** B **foo(q:Q)** metódusa nem látja saját **val** attribútumának értékét, mert az attribútum privát.
- [A]** Q meghívhatja S **bar(x:X)** metódusát, mert mindketten megvalósítják az X interfészt.
- [E]** B interfésze tartalmazza **bar(s:S)** metódust, mert a metódus statikus.
- [A]** Q **bar()** metódusa nem módosíthatja az **im** attribútumot, ezért az attribútum konstans.
- [E]** B-nek nincs **bar(x:X)** metódusa, ezért nem függ az X interfésztől.

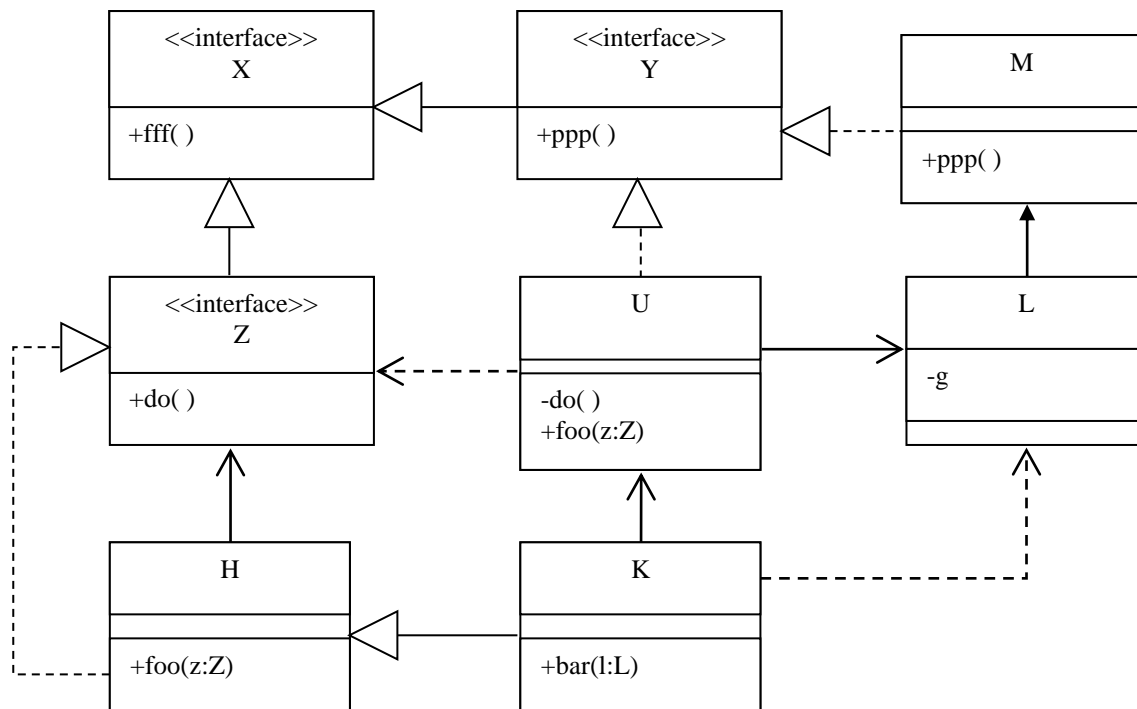
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [B] Y helyettesíthető W-vel, mert mindketten megvalósítják az X interfészt.
- [C] N meghívhatja Y foo( ) metódusát, mert N megvalósítja a Z interfészt.
- [C] M bar(z:Z) metódusa kaphat paraméterül N objektumot, mert van közös ősök.
- [B] W nem helyettesíthető M-mel, mert W-nek nincs bar(z:Z) szignatúrájú metódusa.
- [E] Q helyettesíthető M-mel, mert mindkettő megvalósítja a Z interfészt.
- [B] N xxx( ) metódusa meghívható a W osztály sort(x:X) metódusából, mert az N.xxx( ) statikus.
- [C] W sort(x:X) metódusa meghívhatja egy paraméterül kapott Y objektum bar( ) metódusát, mert W-nek is van ugyanilyen szignatúrájú metódusa.
- [A] M-nek és N-nek különböző az interfésze, mert N nem valósítja meg X-t.

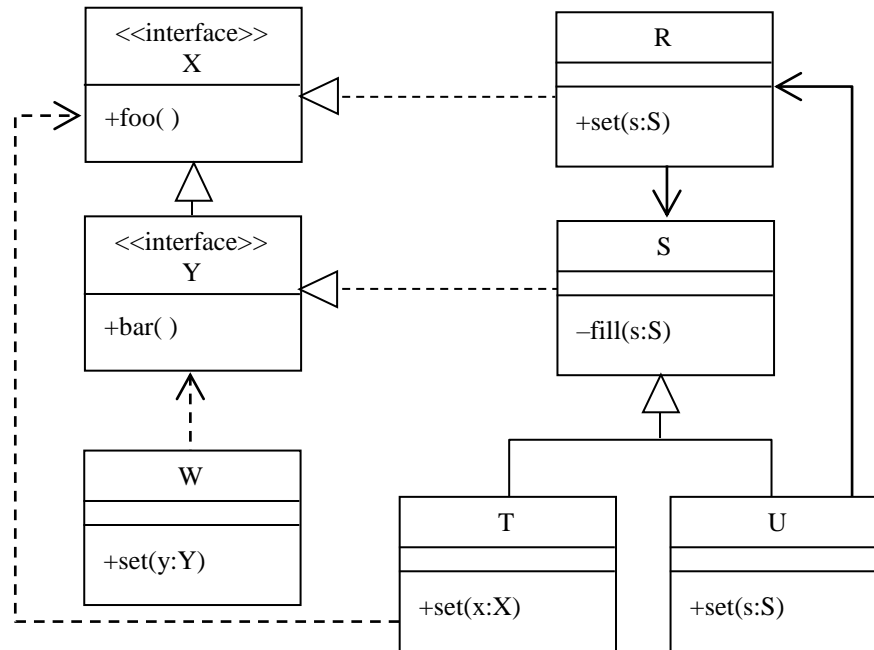
Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !



- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [B] H bárhol helyettesítheti U-t, mert mindketten megvalósítják az X interfészt.
- [B] H foo(z:Z) metódusa meghívható egy U-val, mert U megvalósítja az Y interfészt.
- [A] U nem hívhatja meg K bar(l:L) metódusát, mert K nem függ L-től.
- [A] K implementálja a Z interfészt, ezért K meghívhatja U do() metódusát.
- [B] K nem hozhat létre L objektumot, mert az L g attribútuma privát.
- [C] U foo(z:Z) metódusa nem hívhatja meg egy paraméterül kapott H foo(z:Z) metódusát, mert U nem implementálja a Z interfészt.
- [E] K bárhol helyettesítő U-val, mert K az U leszármazottja.
- [A] L nem ismeri az Y interfészt, ezért L nem hívhatja meg M ppp() metódusát.

Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat !

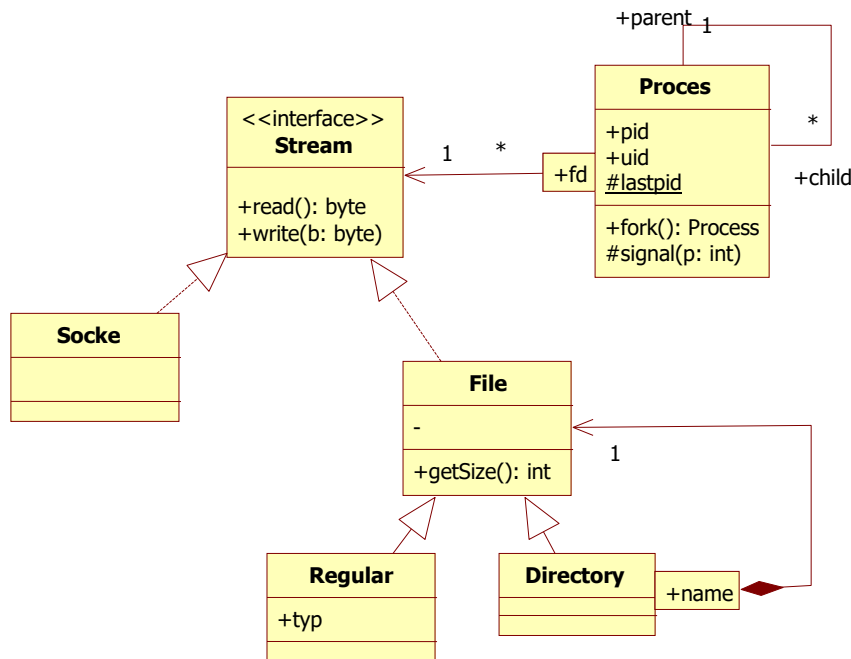


- |   |         |
|---|---------|
| A - csak az első tagmondat igaz                         | (+ -)   |
| B - csak a második tagmondat igaz                       | (- +)   |
| C - mindkét tagmondat igaz, de a következtetés hamis    | (+ + -) |
| D - mindkét tagmondat igaz és a következtetés is helyes | (+ + +) |
| E - egyik tagmondat sem igaz                            | (- -)   |

- [E] Y bárhol helyettesíthető W-vel, mert W az Y leszármazottja.
- [D] U bármikor lehet T.set(x:X) paramétere, mert U megvalósítja az X interfészt.
- [B] R meghívhatja saját set(s:S) metódusából egy W set(y:Y) metódusát, mert S megvalósítja Y-t.
- [E] S fill(s:S) metódusa nem kaphat paraméterül T-t, mert a metódus protected.
- [A] T megvalósítja az X interfészt, mert T az R leszármazottja.
- [B] T pontosan egy U-t tartalmazhat, mert csak egy közvetlen ősük van.
- [E] T bárhol helyettesíthető U-val, mert egyforma az interfészük.
- [B] U meghívhatja S fill(s:S) metódusát, mert R asszociációban van S-sel.

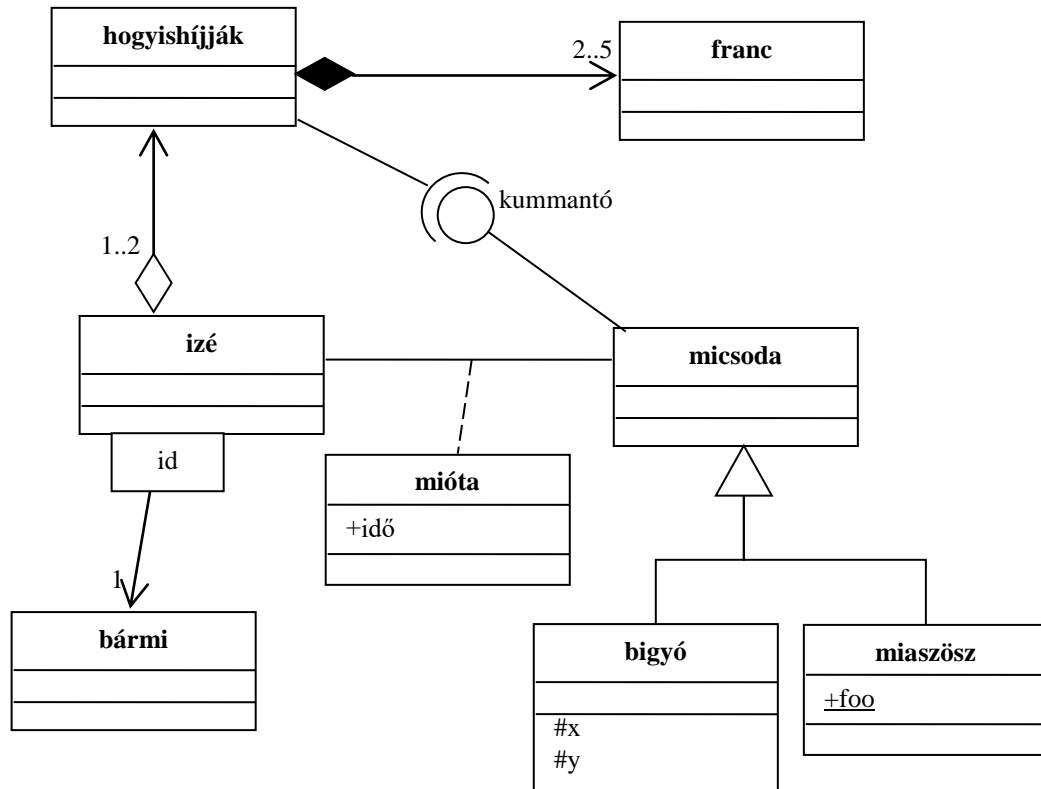
Készítsen UML 2 osztálydiagramot (class diagram) az alábbi leírás alapján! Használja a kövéren szedett kifejezéseket! Ahol lehet, adja meg a paraméterek, attribútumok, stb típusát is!

Az OOniX operációs rendszerben **folyamatok**, **fájlok** és hálózati kapcsolatok (**socket**) vannak. A folyamatoknak van azonosítója (**pid**), tulajdonosa (**uid**). Egy folyamatból a **fork()** metódussal lehet újat létrehozni. Minden folyamat ismeri a közvetlen **őst** és a **gyerekeit**. A fájloknak van egy senki más által nem látható **inode** száma, és lekérdezhető a **méretük**. A fájlok többfélék lehetnek: **könyvtárak**, amelyek más fájlokat tartalmazhatnak (a **nevük** alapján), **reguláris** fájlok, amiknek van **típusa**, stb. Minden fájl egy könyvtár része. A folyamatok egyformán kezelhetnek socketet és fájlt is, de csak egy közös interfészt (**stream**) látnak belőlük, amiken bájtokat lehet **olvasni** és **írni**. Az ilyen objektumokról a folyamatnak van egy listája, aminek az elemeit fájlleíróval (**fd**) azonosítja. A folyamatok létrehozásakor egy (a folyamatok számára közös) számláló (**lastpid**) növekszik, ez lesz az újonnan létrehozott folyamat azonosítója. A folyamatoknak más folyamatok tudnak üzenni a **signal()** üzenet meghívásával (egy darab egész típusú paramétere van). A lastpid és a signal csak folyamatból (és esetleges leszármazottjából) látható.



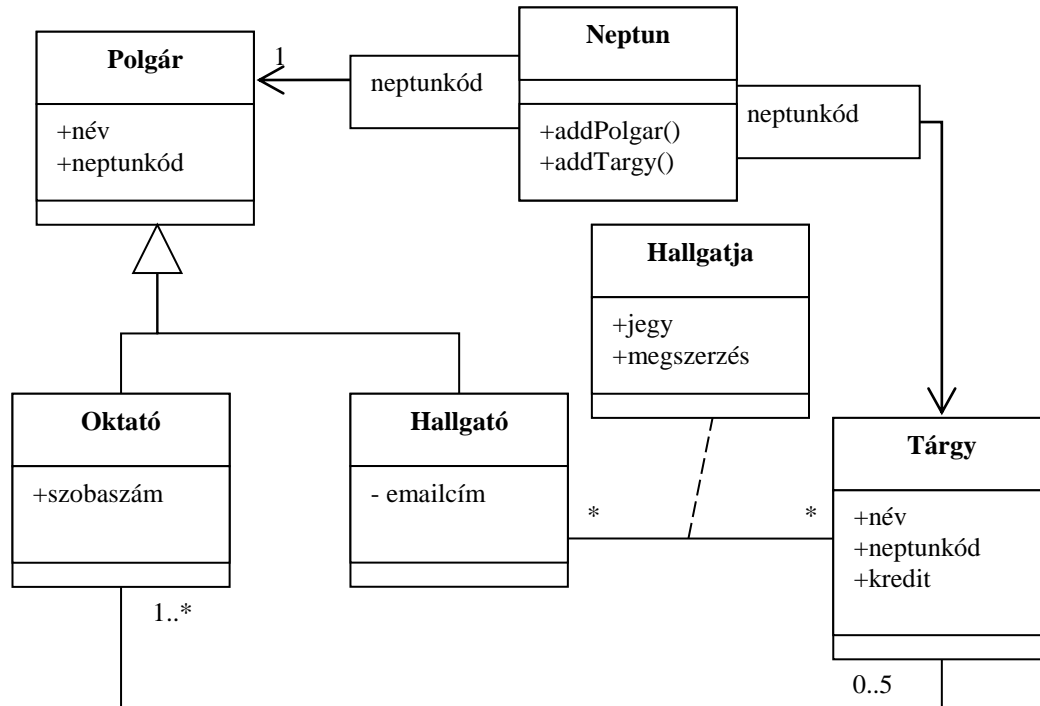
Rajzoljon UML 2 osztálydiagramot az alábbi leírás alapján!

A *hoguishíjják* legalább két, legfeljebb öt *francból* áll. A *francok* létrehozása és megsemmisítése a *hoguishíjják* feladata. A *hoguishíjják* mindig egy nagyobb rendszer, az *izé* része. Az *izének* legalább egy *hoguishíjjákja* kell legyen. Előfordul azonban, hogy egy *hoguishíjják* két *izéhez* is tartozik egyszerre. A *hoguishíjják* a *kummantó* interfészt várja el. Ilyen interfészt biztosít a *micsoda* és két leszármazottja, a *bigyó* és a *miaszösz*. A *bigyónak* két protected metódusa van: *x* és *y*. A *miaszösz* egy darab publikus és statikus attribútummal rendelkezik (*foo*). Az *izék* és a *micsodák* ismerik egymást, több-több kapcsolatban állnak. Nyilván tudjuk tartani, hogy egy *izé* egy adott *micsodával* *mióta* áll kapcsolatban. Végül az *izé* ismer egy csomó *bármit*, amiket egyedi azonosítóval (*id*) különböztet meg egymástól.



Rajzoljon UML2 osztálydiagramot az alábbi történet alapján! Jelölje a számosságokat is!

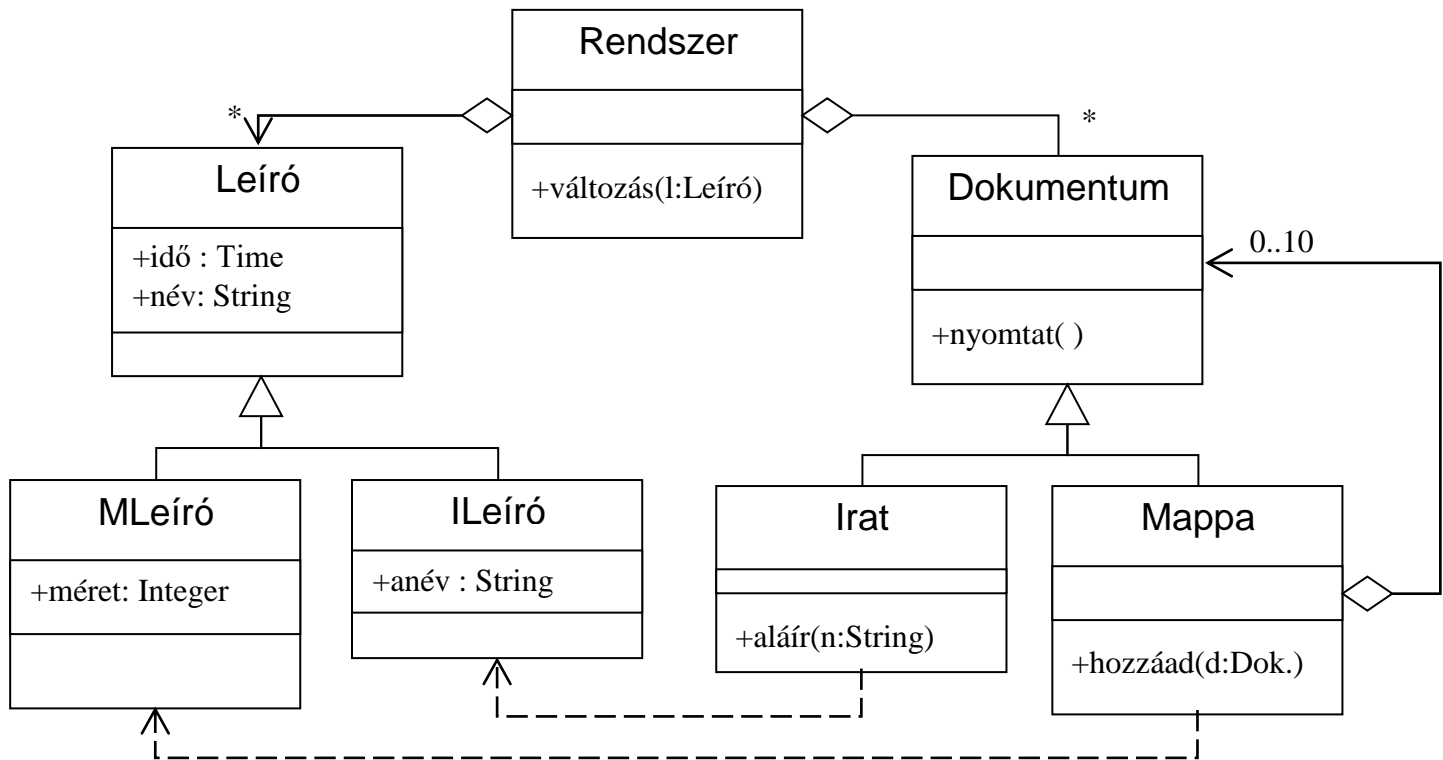
Az egyetemi polgárokat, akiknek nyilvános a neve és a neptunkódja, a Neptun rendszer tartja nyilván, mégpedig a polgáronként egyedi neptunkód alapján. Polgár az oktató és a hallgató is. Az oktatónak nyilvántartjuk a szobaszámát, a hallgatónak az emailcímét (privát adat). A Neptunban tároljuk a tárgyakat is egyedi neptunkóddal. A tárgyaknak szintén ismerjük a nevét és neptunkódját, valamint az értük kapható kreditek számát. Egy hallgató több tárgyat is felvehet (hallgatja), egy tárgyra több hallgató is járhat. Ezen kívül egy adott hallgató egy adott tárgyra kapott jegyét és a megszerzés évét is nyilvántartjuk. Egy tárgyat legalább egy oktató oktat, egy oktatónak pedig lehet több tárgya is (maximum 5), de van akinek egy sincs. A Neptunba fel tudunk venni új tárgyat és polgárt.





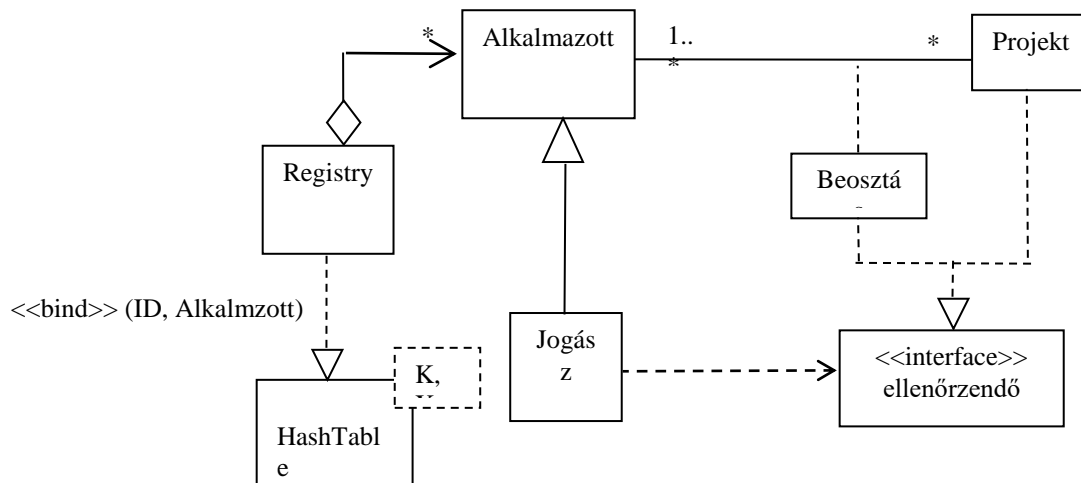
Készítsen UML2 **osztálydiagramot (class diagram)** az alábbi leírás alapján !

Egy dokumentumkezelő rendszerben heterogén kollektcióként névvel ellátott dokumentumokat tárolunk. A dokumentum lehet írat vagy mappa. A mappa további dokumentumokat tárolhat, de maximum 10-et. A rendszernek képesnek kell lennie arra, hogy később további dokumentumfajtákkal (fotó, hangszalag stb. bővítsük). A dokumentumokat ki tudjuk nyomtatni (*nyomtat*), az iratokat alá lehet írni (*aláír*), a mappákba újabb dokumentum helyezhető (*hozzáad*). A rendszer nyilvántartja a dokumentum-módosításokat is. Minden dokumentum a módosításról értesíti a rendszert (*változás* metódus), és átadja a módosítás adatait leíró objektumot. Ebben a módosuló dokumentum megadja a nevét és a módosítás idejét, de ezen kívül a dokumentum típusától függő egyedi adatok is szerepelhetnek (irat esetén az aláíró neve, mappa esetén a mappa mérete).

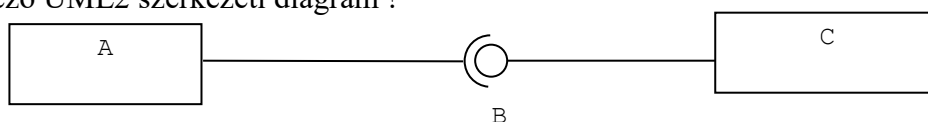


Rajzoljon UML 2 osztálydiagramot! A metódusokat és attribútumokat nem kell jelölje ! Csak a **vastagon** szedett classifier-ek szerepeljenek a diagramon !

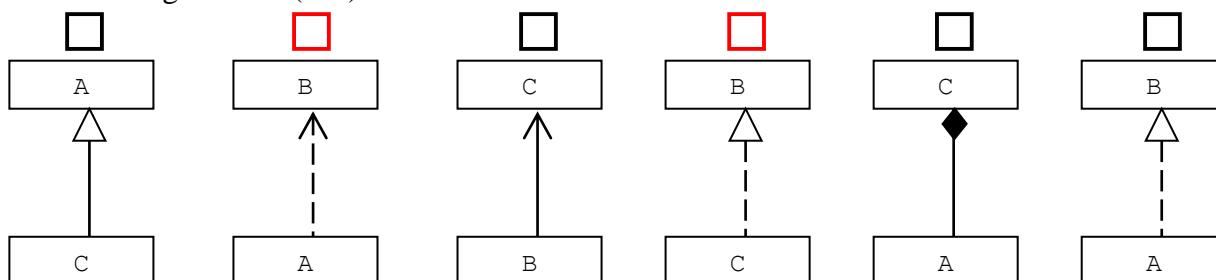
Egy cégnél nyilvántartják az **alkalmazottak** és a **projektek** adatait. Az alkalmazottak projektekhez vannak rendelve, minden projektnél különböző beosztásban. Minden projekten dolgozik legalább egy alkalmazott, de lehet olyan alkalmazott, aki éppen nincs projekthez rendelve. A **beosztás** határozza meg például, hogy mennyi bónuszt kap az alkalmazott az adott projekt sikere esetén. Mind a projekt, mind a beosztás jogilag **ellenőrzendő** (megvalósítják az **ellenőrzendő** interfészt). A **jogász** (aki persze a cég alkalmazottja is egyben) dolga, hogy az ellenőrzéseket elvégezze. A **registry**ben tárolják az alkalmazottak azonosítóinak és az alkalmazotti adatoknak az összerendelését. A registry az egyszerűség kedvéért a **HashTable** (K kulcs és X érték paraméterű) template osztályt példányosítja, ahol a kulcs az azonosító, az érték az alkalmazott adata.



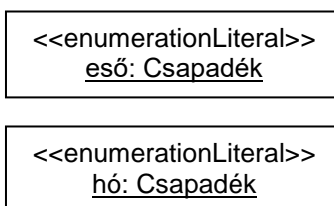
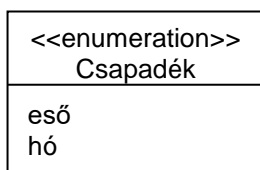
Legyen a következő UML2 szerkezeti diagram !



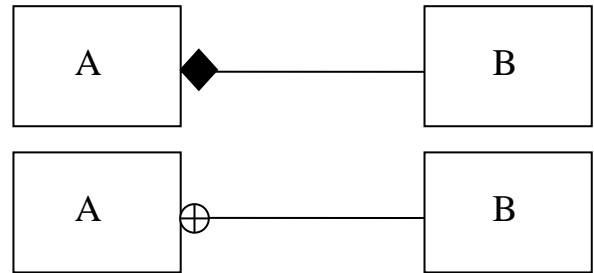
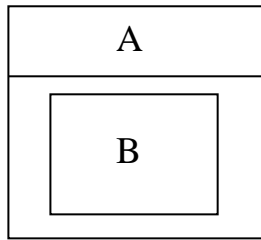
Feltételezve, hogy a fenti szerkezeti diagramon szereplő elemek között egyéb kapcsolat nincs, jelölje meg az alábbiak közül az igaz állítás(oka)t !



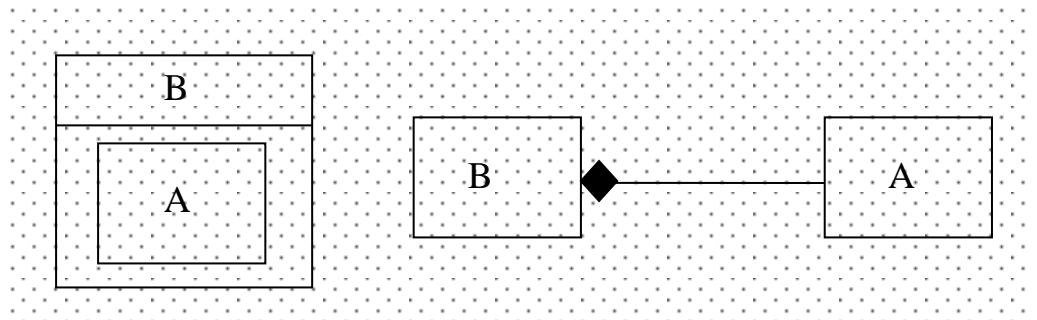
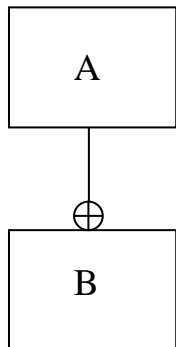
Definiálja az alábbi felsorolás konstansait UML2-ben!



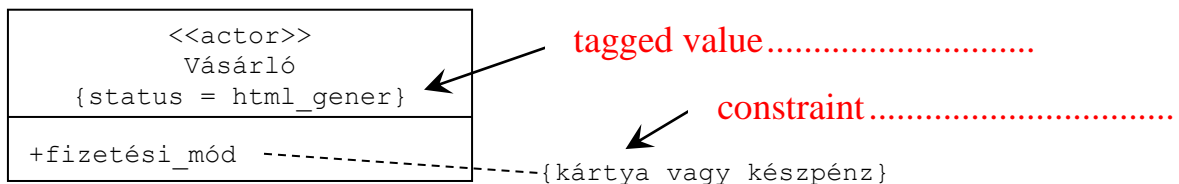
Adott az alábbi UML2 diagram. Rajzoljon egy olyan másik UML2 diagramot, amely szemantikailag ugyanazt fejezi ki, mint az adott diagram !



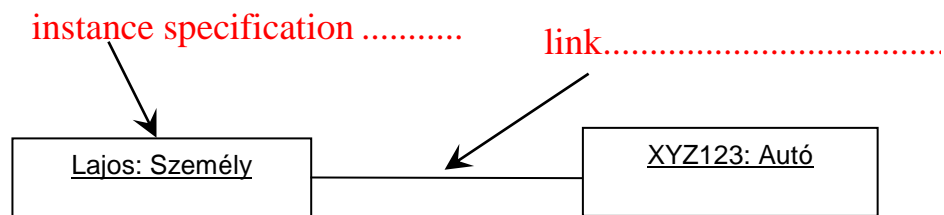
Adott az alábbi UML2 diagram. Rajzoljon két olyan másik UML2 diagramot, amely szemantikailag ugyanazt fejezi ki, mint az alábbi diagram !



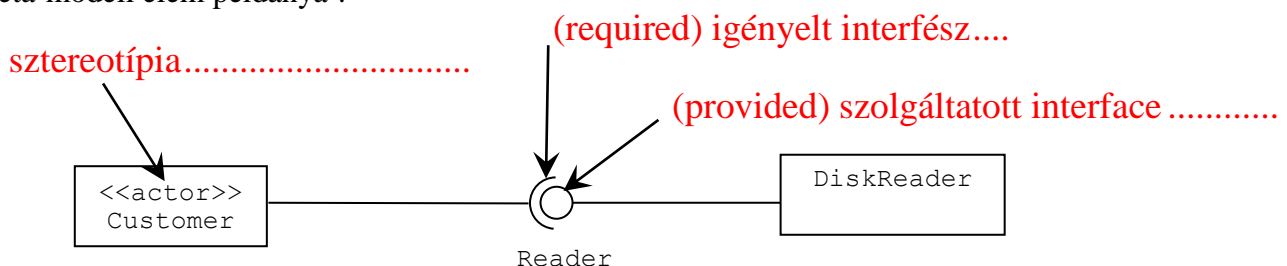
Adja meg, hogy a jelölt elemek melyik UML meta-modell elemek példányai !



Adja meg, hogy az alábbi object diagramon a megjelölt elemek mely UML2 meta-modell elem példányai !

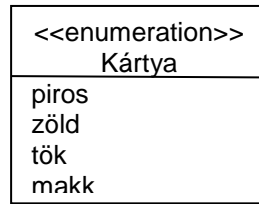


Az alábbi ábrán három UML2 modell elemet megjelöltünk. Adja meg elemenként, hogy az melyik UML2 meta-modell elem példányai !



Definiálja UML2-ben az alábbi felsorolást !

Kártya = [piros | zöld | tök | makk]

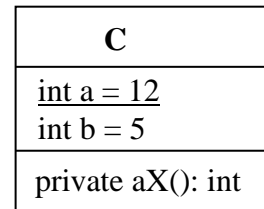


Elkészítjük az alábbi C osztály két példányát, c1-et és c2-t. Ezt követően végrehajtjuk a következő műveleteket:

```
c2.a = 8; c1.a = -2;
c1.b = c2.a + 4;
c2.b = c2.a + c1.b;
```

Mennyi lesz a változók értéke ?

c1.b = **2** ..... c2.b = **0** .....

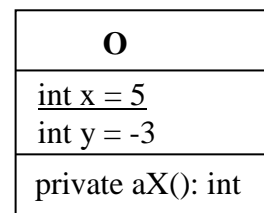


Elkészítjük az alábbi O osztály két példányát, o1-et és o2-t. Ezt követően végrehajtjuk a következő műveleteket:

```
o2.x = -3; o1.x = 4;
o1.y = o2.x + 4;
o2.y = o2.x + o1.y;
```

Mennyi lesz a változók értéke ?

o1.y = **8** ..... o2.y = **12** .....

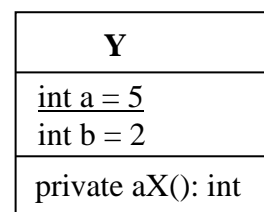


Elkészítjük az alábbi Y osztály két példányát, y1-et és y2-t. Ezt követően végrehajtjuk a következő műveleteket:

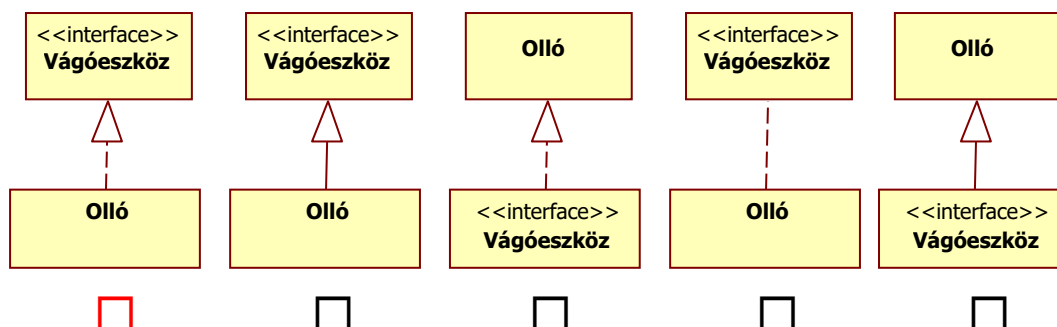
```
y2.a = 3; y1.a = -3;
y1.b = y2.a + 4;
y2.b = y2.a + y1.b;
```

Mennyi lesz a változók értéke ?

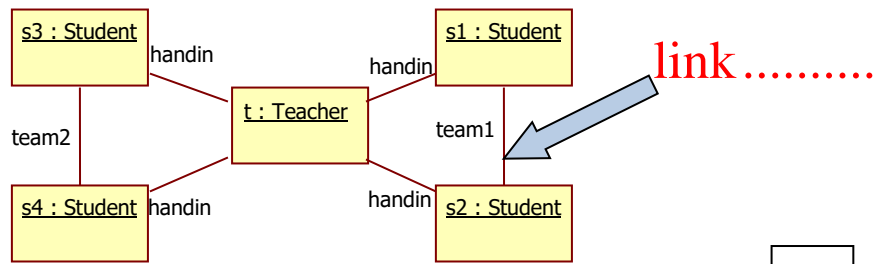
y1.b = **1** ..... y2.b = **-2** .....



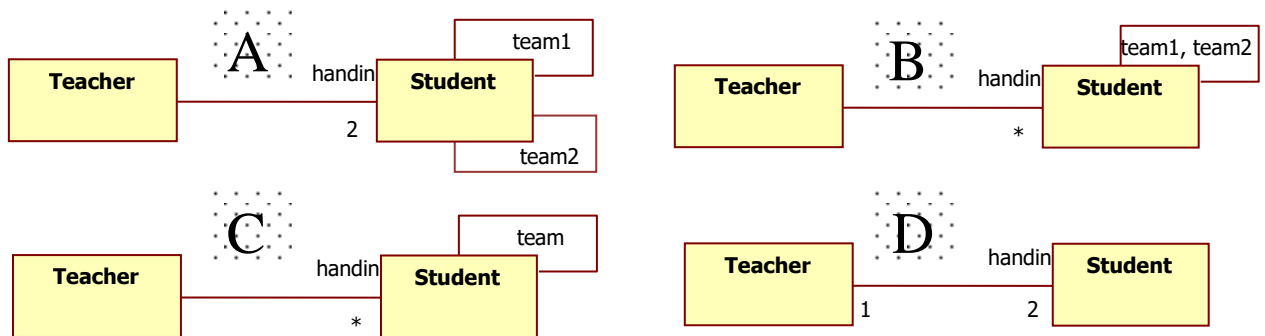
Jelölje meg az alábbi ábrák közül a szintaktikailag és szemantikailag helyese(ke)t!



Adja meg, hogy az alábbi ábrán megjelölt vonal melyik UML2 meta-modell elem példánya !



A fenti diagram, mely alatti osztálydiagramnak felel meg ?



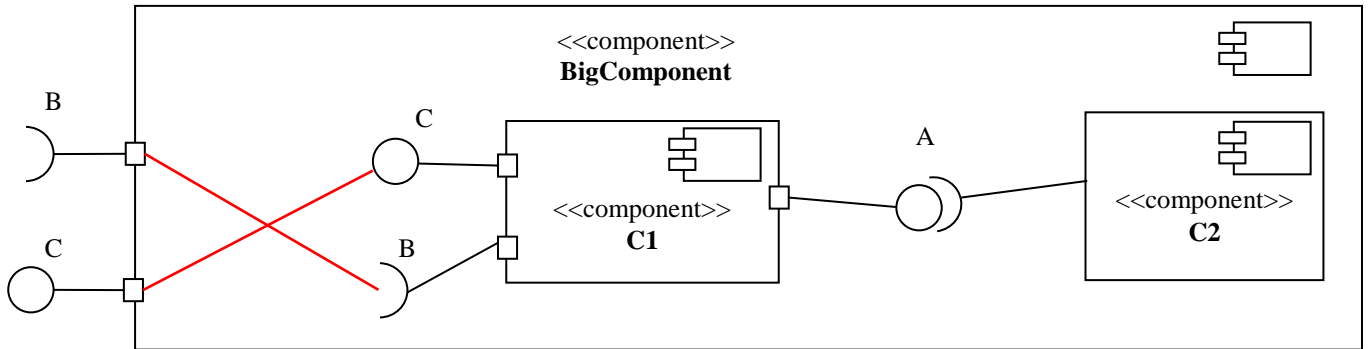
Az alábbi kulcs felhasználásával jellemezze az állításokban megfogalmazott relációkat !

<b>S</b>	specializálás (öröklés)
<b>A</b>	asszociáció
<b>D</b>	függőség (dependency)
<b>C</b>	kompozíció

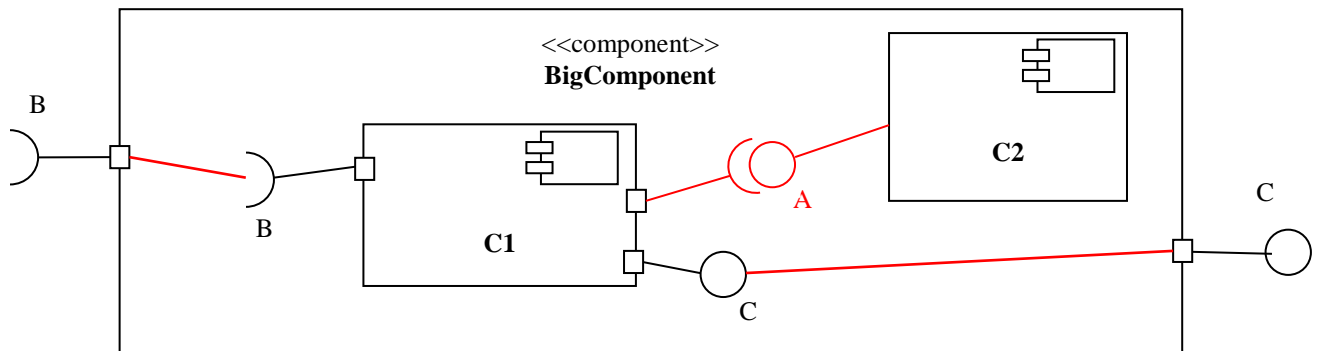
Egy ember baltával dolgozik.	<b>D</b>
Egy file rekordokból áll.	<b>C</b>
Egy sokszög a csúcspontok rendezett sorozata.	<b>C</b>
A függőség (dependency) a relációnak egy formája.	<b>S</b>

## 2 Komponensdiagram

Rajzolja be alábbi UML2 diagramba a hiányzó kapcsolatokat!

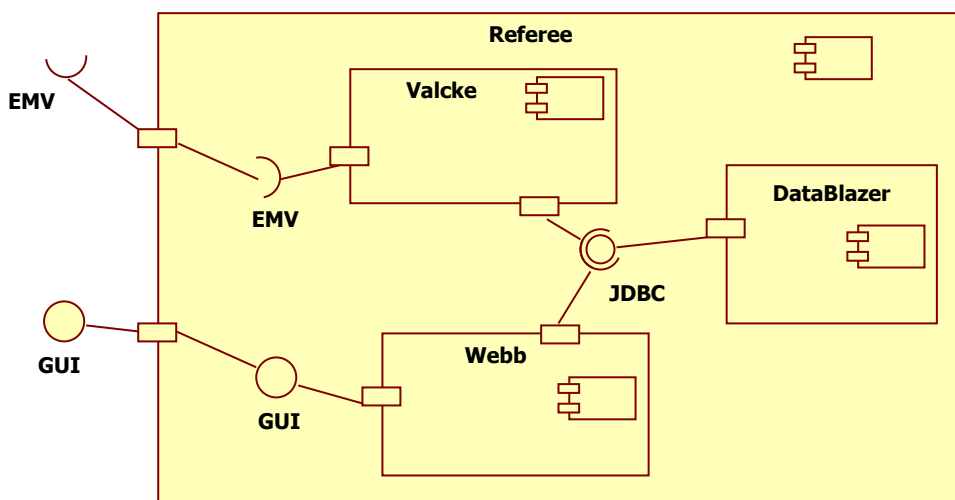


Egészítse ki az alábbi UML2 diagramot a hiányzó részletekkel, figyelemmel arra, hogy **C1** komponens használja a **C2** komponens által megvalósított **A** interfészt!

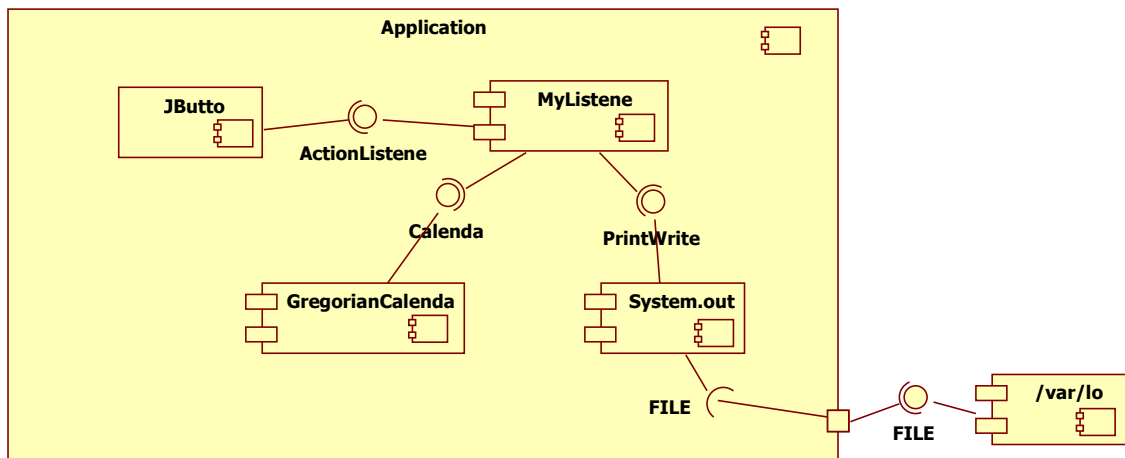


Készítsen UML2 komponens-diagramot az alábbi leírás alapján!

A FICA-nál (Fédération Internationale de Corruption Association) a pályázatok elbírálását egy speciális készülék (Referee) végzi. A készüléken található egy bankkártyaolvasó port (ami a szabványos EMV bankkártyák interfészét várja el), valamint egy pályázati-anyagkezelő webes felület (amit bárki szabadon elérhet). A készülék három komponensből áll: egy bankkártyakezelőből (Valcke, ez biztosítja az EMV csatlakozást), egy webszerverből (Webb, ez biztosítja a webes GUI-t), és egy adatbázisból (DataBlazer, ami a másik két komponens számára JDBC hozzáférést biztosít).

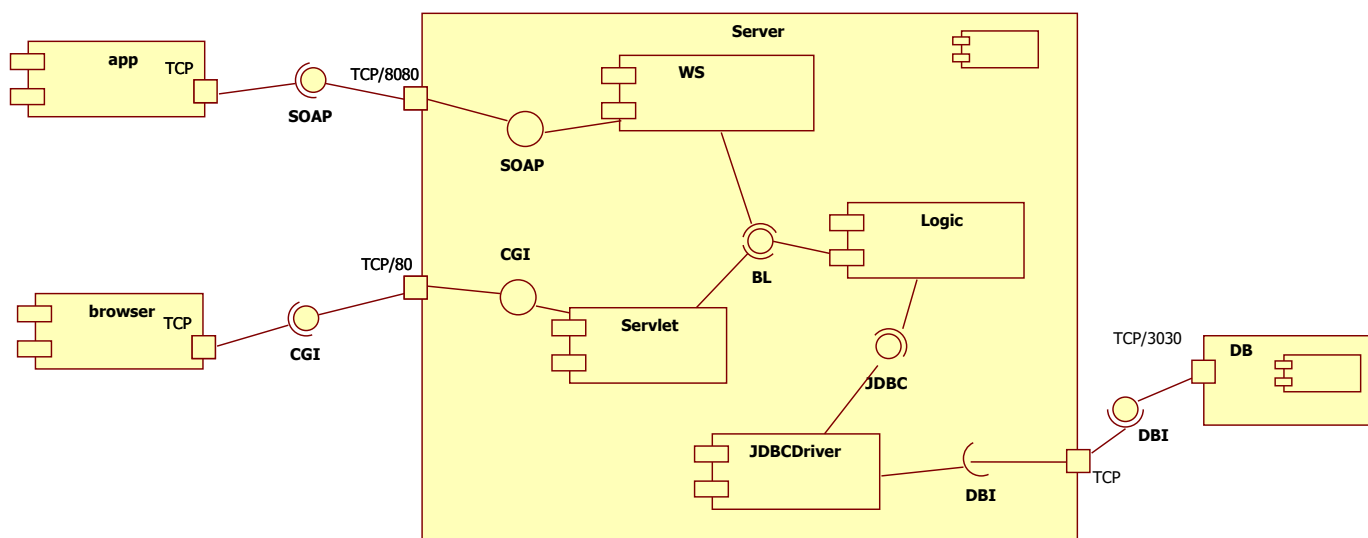


Készítsen UML2 komponens diagramot arra az esetre, amikor egy Swing-es alkalmazásban egy gomb megnyomására a MyListener osztály egy példánya a PrintWriter interfészű System.out-ra kiírja az aktuális dátumot, amit a Calendar interfészen keresztül kap meg egy GregorianCalendar objektumtól. A System.out megvalósítása C-ben történt, és az alkalmazásból ebben az esetben egy FILE\* típusú változón keresztül egy logfájlba (/var/log) van irányítva.



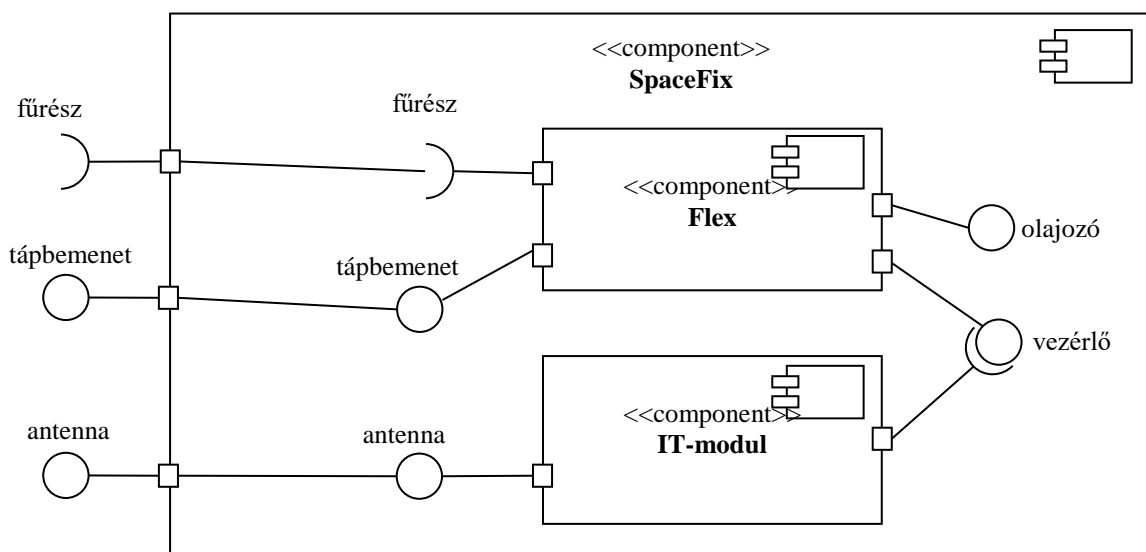
Készítsen UML2 komponens diagramot az alábbi történet alapján!

Egy szerverhez a 80-as TCP portján böngészők tudnak kapcsolódni. A böngészők ezen a porton a CGI interfészt használják. Ugyanehhez a szerverhez mobil alkalmazások is tudnak csatlakozni, ők a 8080-as TCP porton elérhető SOAP interfészt veszik igénybe. A szerver az adattároláshoz adatbáziskezelőt használ, amit az adatbáziskezelő speciális DBI interfészen keresztül ér el. Ezt az interfészt az adatbázis a 3030-as portján publikálja. A szerver négy komponensből áll. A SOAP interfészt a WebService komponens, a CGI interfészt a Szervlet komponens valósítja meg. Mindkettő a BusinessLogic API-t biztosító Logika komponens használja. Ez utóbbi egy JDBC felülettel rendelkező JDBCDriver komponensen keresztül éri el az adatbázist.

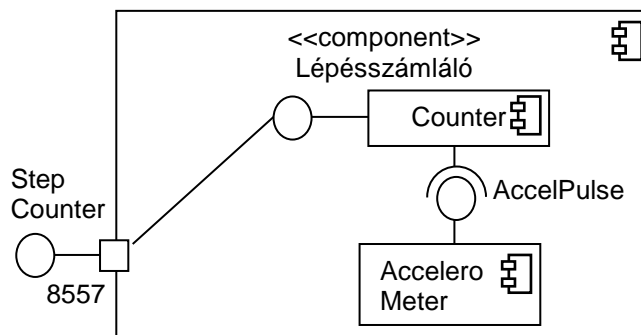


Készítsen UML2 komponens diagramot az alábbi leírás alapján!

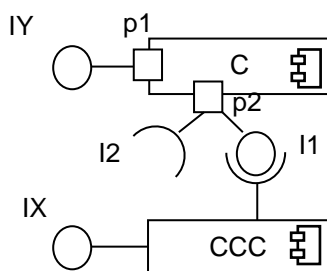
A SpaceFix magyar műholdkarbantartó-modul felületén van fűrés (ami elvárja, hogy valamit fűrészelhessen), antennabemenet és tápbemenet. Ha a modult szétszedjük, akkor azt látjuk, hogy valójában egy flexből és egy IT-modulból áll. A flex biztosítja a fűrészelést és kapja a tápbemenetet, míg az antennabemenet az IT-modulhoz kapcsolódik. A flex olajozónírással és vezérlőgombbal is rendelkezik, az IT-modul ebből a vezérlőgombot nyomkodja a flex irányításához.



A mobiltelefonba épített lépésszámláló komponens a 8557-es porton megvalósítja a StepCounter interfészt. Ez az interfész megegyezik a lépésszámláló felépítéséhez felhasznált Counter komponens interfészeivel. A Counter felhasználja az AcceleroMeter komponenst, azzal az AccelPulse interfészen keresztül kapcsolódik. Rajzoljon UML2 komponens diagramot !

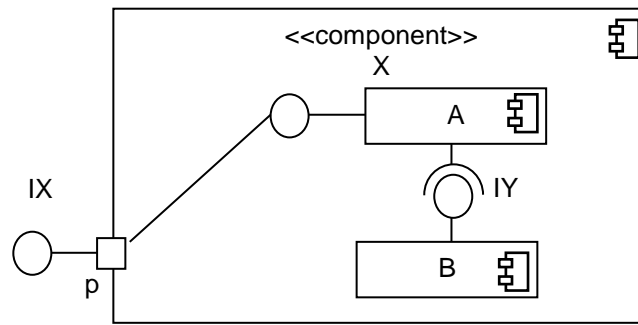


Legyen egy C komponensünk, amely a p1 portján megvalósítja az IY interfészt, a p2 portján megvalósítja az I1 interfészt és várja az I2 interfészt. Van egy CCC komponensünk is, amely az I1 interfészen keresztül kapcsolódik a C komponenshez. A CCC komponens megvalósítja az IX interfészt is. Rajzoljon UML2 komponens diagramot !



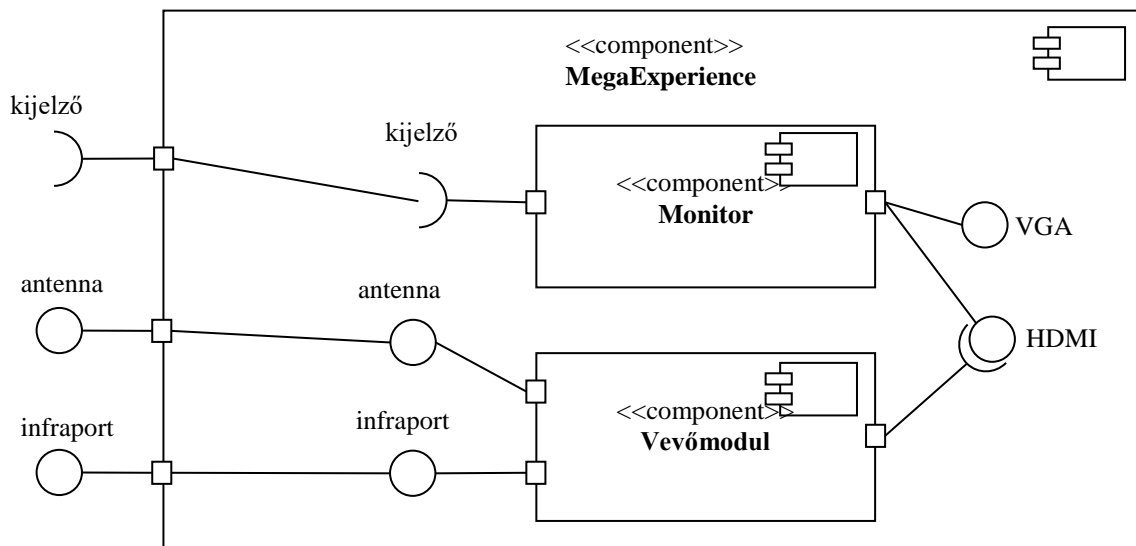


Legyen egy X komponensünk, amely p portján megvalósítja az IX interfészt. Ennek a komponensnek a felépítéséhez felhasználjuk az A komponenst, amely realizálja az IX interfészt, de szükséges felhasználnunk egy B komponenst is, amely megvalósítja az A által elvárt IY interfészt. Rajzoljon UML2 komponens diagramot !

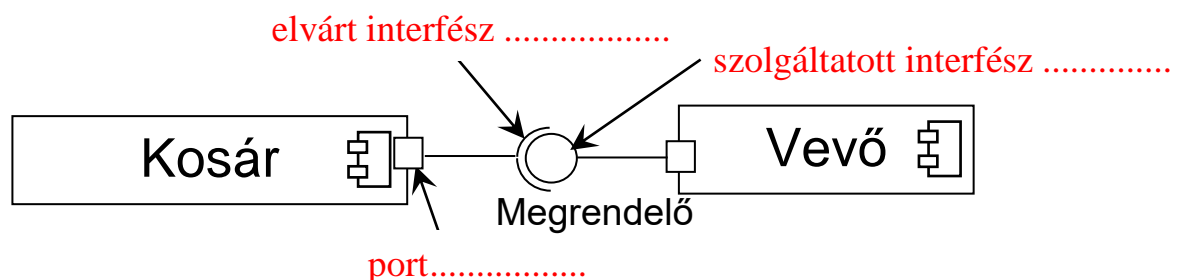


Készítsen UML2 komponens diagramot az alábbi leírás alapján!

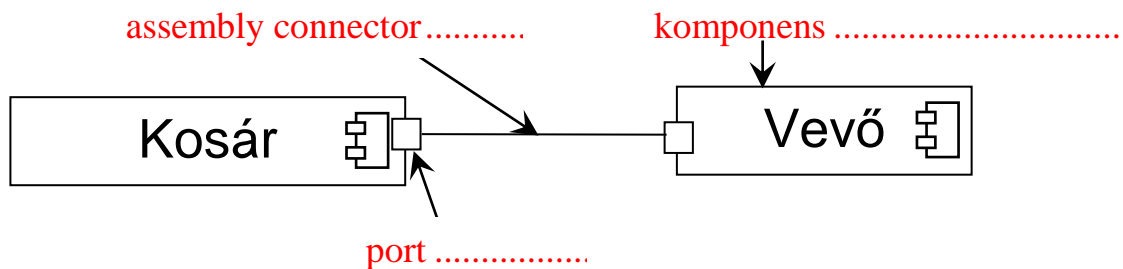
A MegaExperience televíziókészülék a szokásos felülettel rendelkezik: van kijelzője (ami elvárja, hogy valaki nézze), antennabemenete és infraportja. Ha a készüléket szétszedjük, akkor azt látjuk, hogy valójában egy monitorból és egy vevőmodulból áll. A monitor biztosítja a kijelzést, míg az antennabemenet és az infraport a vevőmodulhoz kapcsolódik. A monitor VGA és HDMI interfésszel is rendelkezik, a modul ebből a HDMI interfészt használja a kommunikációhoz.



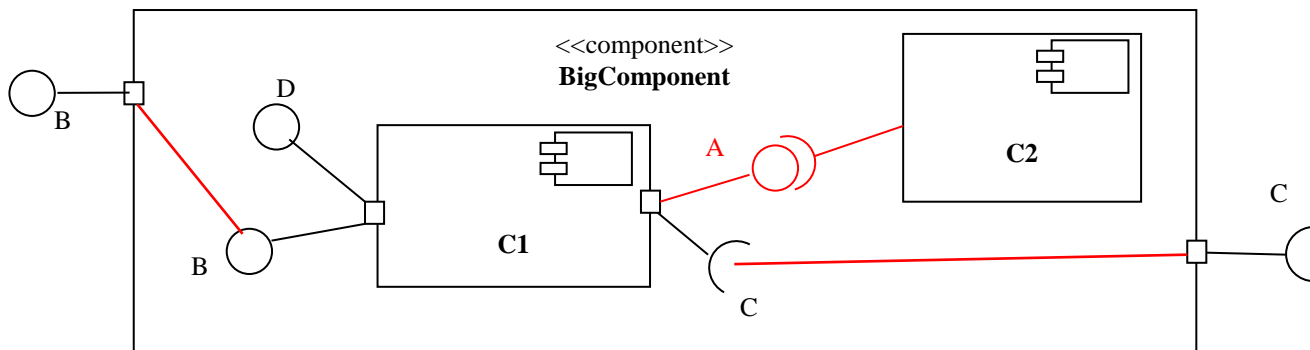
Az alábbi ábrán három UML2 modell elemet megjelöltünk. Adja meg elemenként, hogy az melyik UML2 meta-modell elem példánya !



Az alábbi ábrán három UML2 modell elemet megjelöltünk. Adja meg elemenként, hogy az melyik UML2 meta-modell elem példánya !

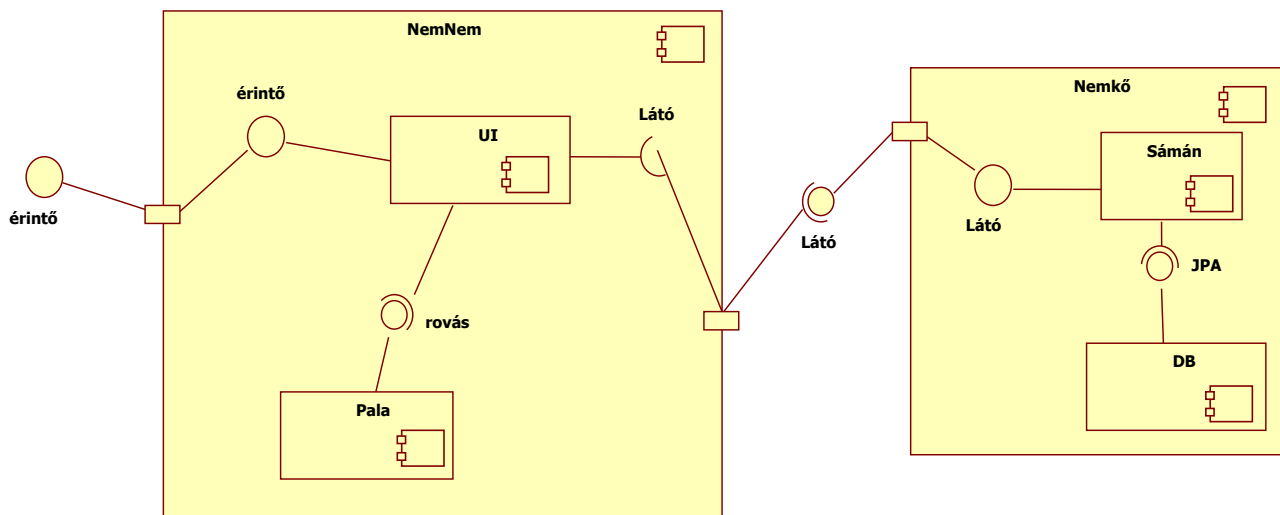


Egészítse ki az alábbi UML2 diagramot a hiányzó részletekkel, figyelemmel arra, hogy **C2** komponens használja a **C1** komponens által megvalósított **A** interfészt!



Készítsen UML2 komponens diagramot a következő leírás alapján!

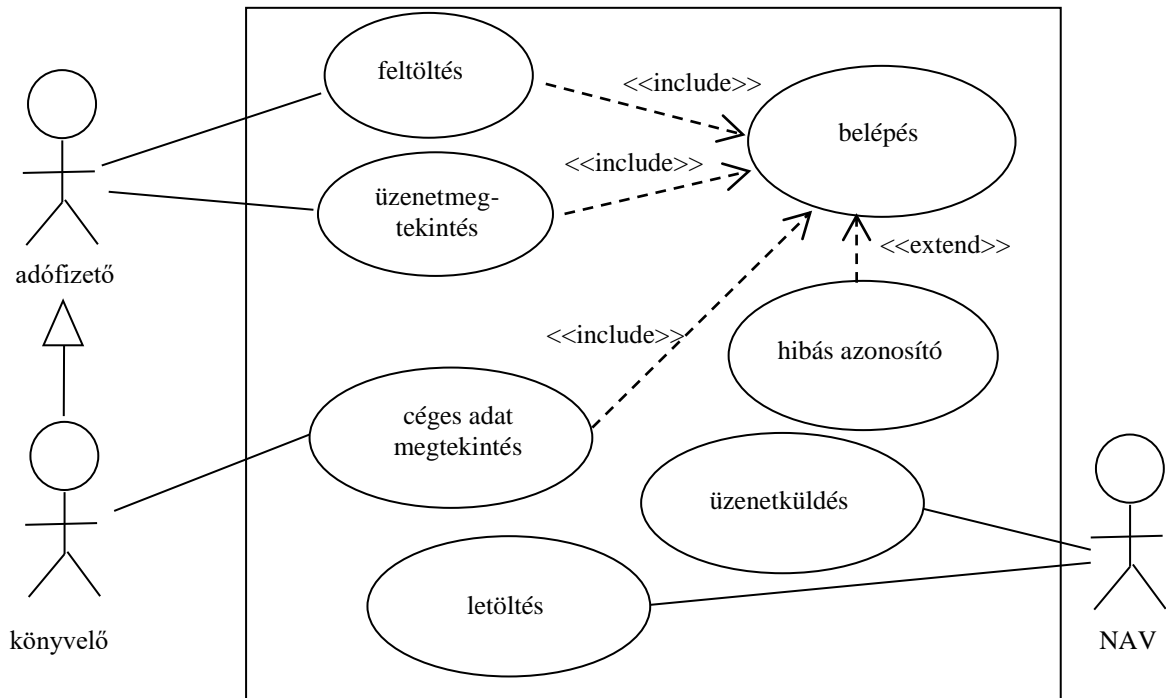
A Nemzeti Nemzető társkereső alkalmazás mobil appja (NemNem) érintőképernyő interfészen kapja a felhasználói inputot. Ezt az UI modul kezeli, amely a bemenetet egyfelől a Pala modulnak adja át (Rovás interfész használatával), másrészt internetes kapcsolaton keresztül a Nemzeti Nemzető Központnak (Nemkő) továbbítja, ahol a beérkező adatokat a Látó interfészt biztosító Sámán modul dolgozza fel. A Sámán JPA interfészű adatbázisban tárol mindent.



### 3 Use-case diagram

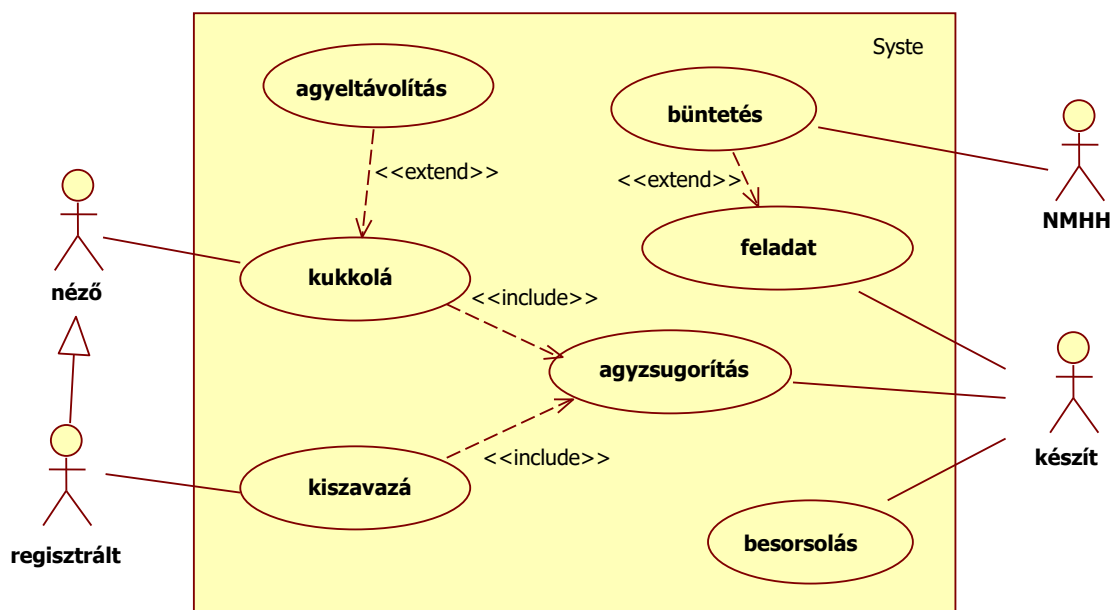
Rajzoljon UML2 use-case diagramot az alábbi történet alapján!

Az Ügyfélkapun az adófizető adóbevallást tud feltölteni, illetve hivatalos üzeneteit tudja megnézni. Mindkettőhöz be kell lépnie (név és jelszó megadásával). A könyvelő ezen kívül céges adatokat is meg tud nézni (ehhez szintén be kell lépnie). Ha belépéskor valaki rossz azonosítót ad meg, akkor hibaüzenetet kap. A Nemzeti Adó és Vámhivatal (NAV) le tudja tölteni az adóbevallásokat, és üzenetet tud küldeni.

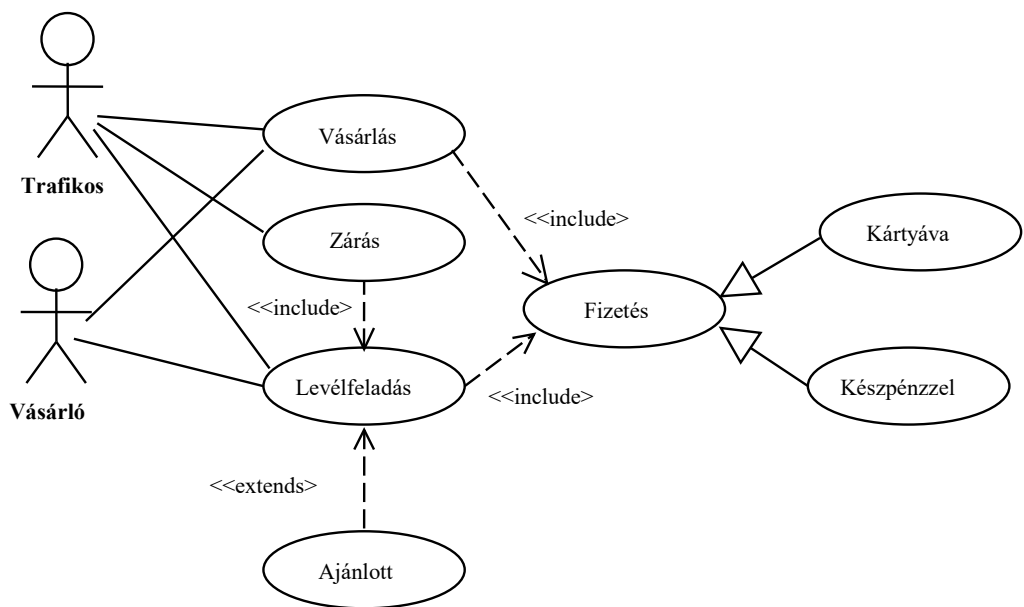


Az alábbi történet alapján készítsen UML2 használati eset (use-case) diagramot!!

A HelóVilág c. tévéműsor folyamán a nézők kukkolhatnak. Ez egyes esetekben agyeltávolításban végződik. A regisztrált nézők a kukkolás mellett kiszavazásban is részt vehetnek. Mind a kiszavazás, mind a kukkolás során a rendszer végrehajtja az agyzsugorítást. A rendszert a készítő is használják: egyfelől besorsolhatnak idiótákat, másfelől feladatokat adhatnak meg, harmadrészt pedig aktív résztvevői az agyzsugorításnak. Ha a feladat túl rücskösre sikerül, akkor büntetéssel jár, ennek mértékét az NMHH határozza meg.

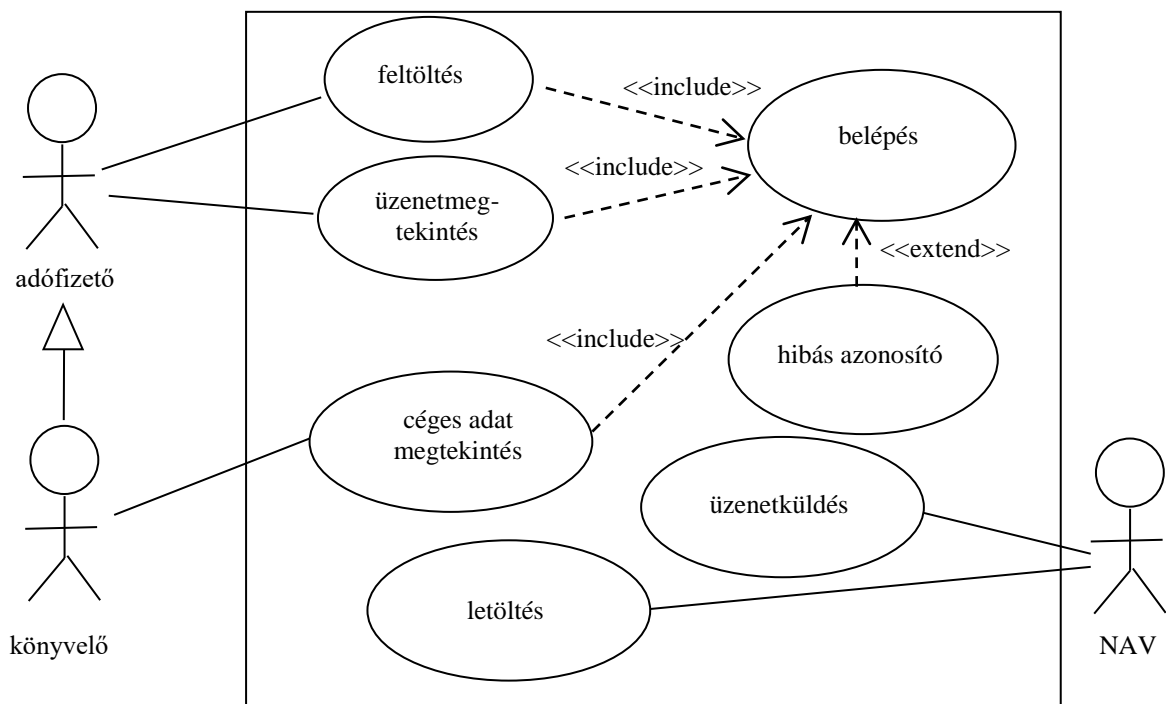


A nemzeti dohányárudában a trafikos közreműködésével lehet – cigarettát, alkoholt stb. – vásárolni és levelet földadni. Késszpénzzel vagy SZÉP-kártyával fizethet a vásárló. Egyes trafikokban lehetőség van ajánlott leveleket is földadni. Záráskor a napi forgalmi adatokat a trafikos levélben feladja a nemzeti adóhivatal részére. Rajzoljon UML2 use-case diagramot !



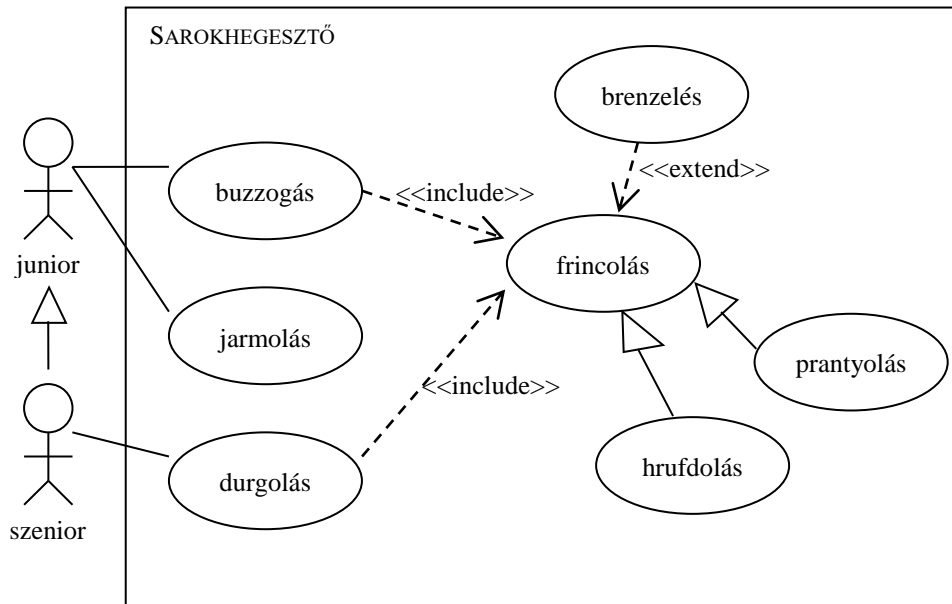
Rajzoljon UML2 use-case diagramot az alábbi történet alapján!

Az Ügyfélkapun az adófizető adóbevallást tud feltölteni, illetve hivatalos üzeneteit tudja megnézni. Mindkettőhöz be kell lépnie (név és jelszó megadásával). A könyvelő ezen kívül céges adatokat is meg tud nézni (ehhez szintén be kell lépnie). Ha belépéskor valaki rossz azonosítót ad meg, akkor hibaüzenetet kap. A Nemzeti Adó és Vámhivatal (NAV) le tudja tölteni az adóbevallásokat, és üzenetet tud küldeni.



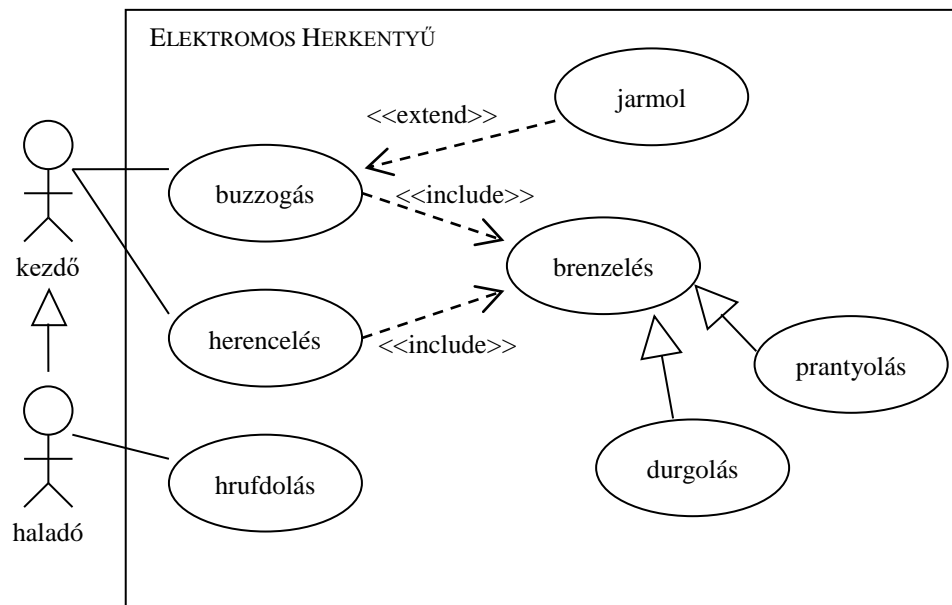
Készítsen UML2 use-case diagramot az alábbi leírás alapján!

A sarokhegesztővel a junior furga buzzogni, és jarmolni tud. A buzzogáshoz be kell kapcsolni a frincolást. Ennek két módja van: a prantolás és a hrufdolás. A herkentyű hibás beállítások esetén frincolás közben brenzel is kicsit. A szenior furga a fentiek mellett a durgolás funkcióhoz is hozzáfér, amihez szintén be kell kapcsolni a fenti frincolás funkciót.



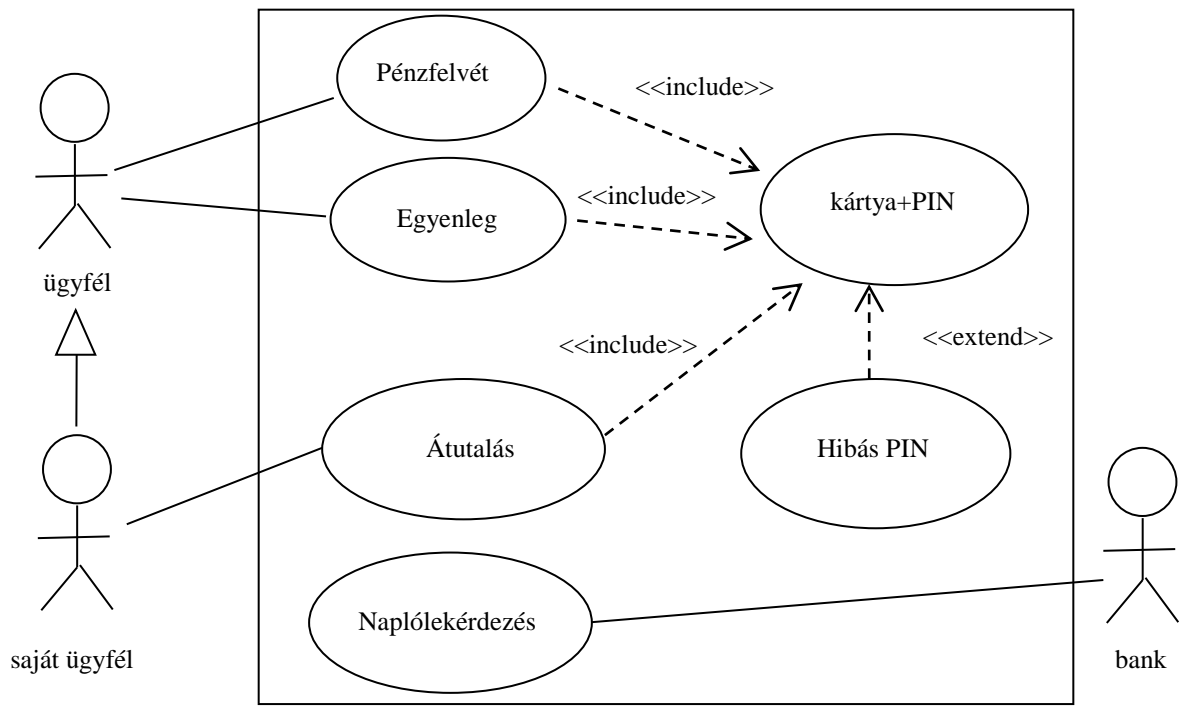
Készítsen UML2 use-case diagramot az alábbi leírás alapján!

Az elektromos herkentyűn a kezdő kukor buzzogni, és herencelni tud. Mindkettőhöz be kell kapcsolni a brenzelést. Ennek két módja van: prantolással vagy durgolással. A herkentyű hibás beállítások esetén buzzogás közben jarmol is kicsit. A haladó kukor a fentiek mellett a hrufdolás funkcióhoz is hozzáfér.



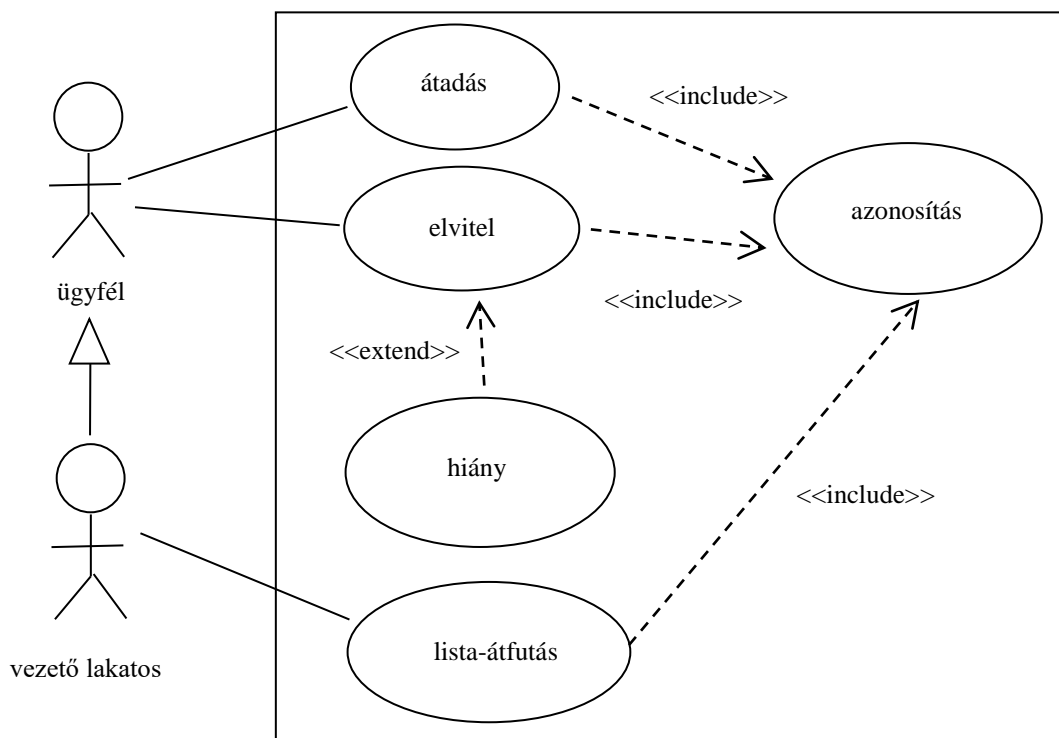
Rajzoljon UML2 use-case diagramot az alábbi történet alapján!

A Management Optimal Bonus (MOB) Bank automataival pénzt lehet felvenni, számlaegyenleget lehet lekérdezni, és a bank saját ügyfelei pénzt utalhatnak a bank vezetőségi bónusz programja számára. Mindezen funkciók eléréséhez be kell helyezni a kártyát és meg kell adni a 4 jegyű azonosítót (PIN). Ha ez háromszor egymás után nem sikerül, az automata a kártyával elérhető teljes összeget a bank jutalomkeretére utalja. Ezen kívül a bank lekérdezheti az automata naplófájlját.

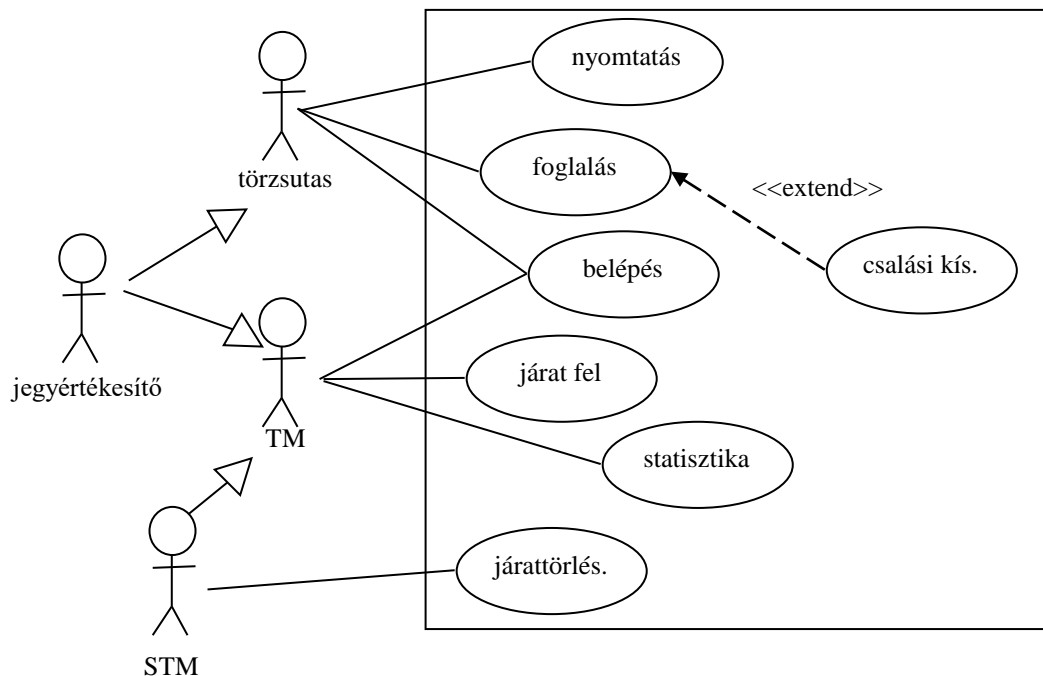


Rajzoljon UML 2.0 use-case diagramot az alábbi leírás alapján!

A Rezesbanda Kft. éjjel-nappali autóbontó telepet működtet, amit informatikai rendszerrel kíván támogatni. A regisztrált ügyfelek roncs autókat adnak át, és alkalmanként használt autóalkatrészt visznek el. Mindkét esetben jelszóval azonosítják magukat. Ha a kért autóalkatrész nincs raktáron, akkor a rendszer felírja a kérést a kívánságlistára. A telep vezető lakatosja időnként átfutja a kívánságlistát, hogy lássa, mire van szükség, ekkor ő is jelszóval azonosítja magát. A vezető lakatos ügyfélként is viselkedhet.

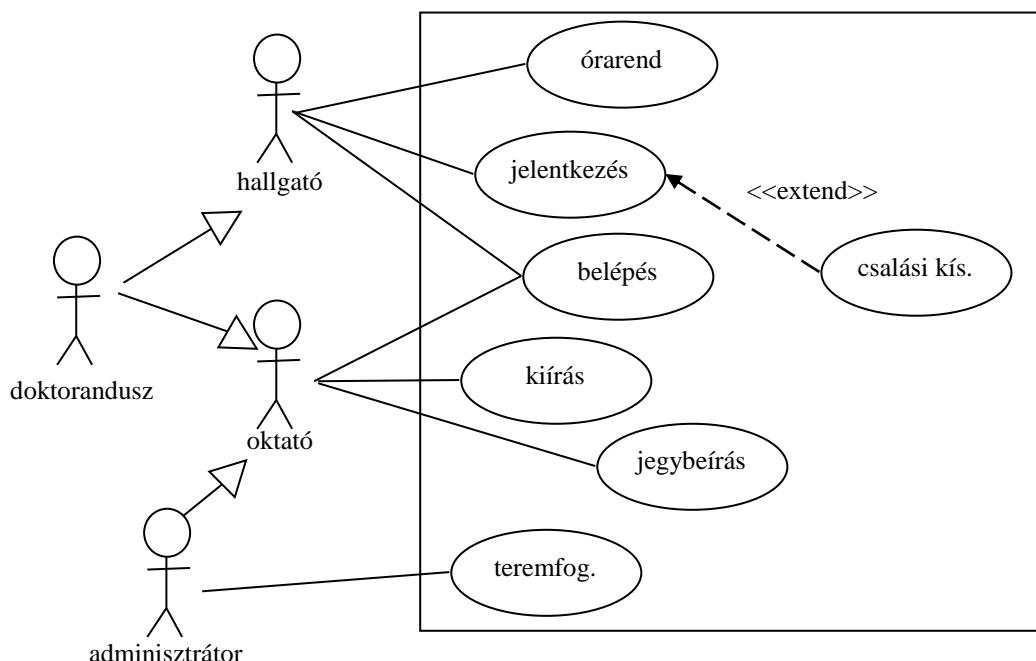


A Lushfanta légitársaság jegyfoglaló rendszerét törzsutasok és ticket-managerek (TM) használhatják a repülőjegyfoglalások rögzítésére. A törzsutasok bejelentkezhettek, jegyet foglalhatnak és beszállókártyát nyomtathatnak. Ha jegyfoglalás közben kiderül, hogy módosultak a session-adatok, akkor a rendszer rögzíti a csalási kísérletet. A TM-ek is belépnek, járatok adatait vihetik fel, illetve jegyvásárlási statisztikákat kérhetnek le. A senior ticket-managerek (STM) az egyszerű TM lehetőségein túl még járatot is törölhetnek. A jegyértékesítők, mivel járatok adatait is kezelniük kell, mind TM-ként, mint törzsutasként használhatják a rendszert.



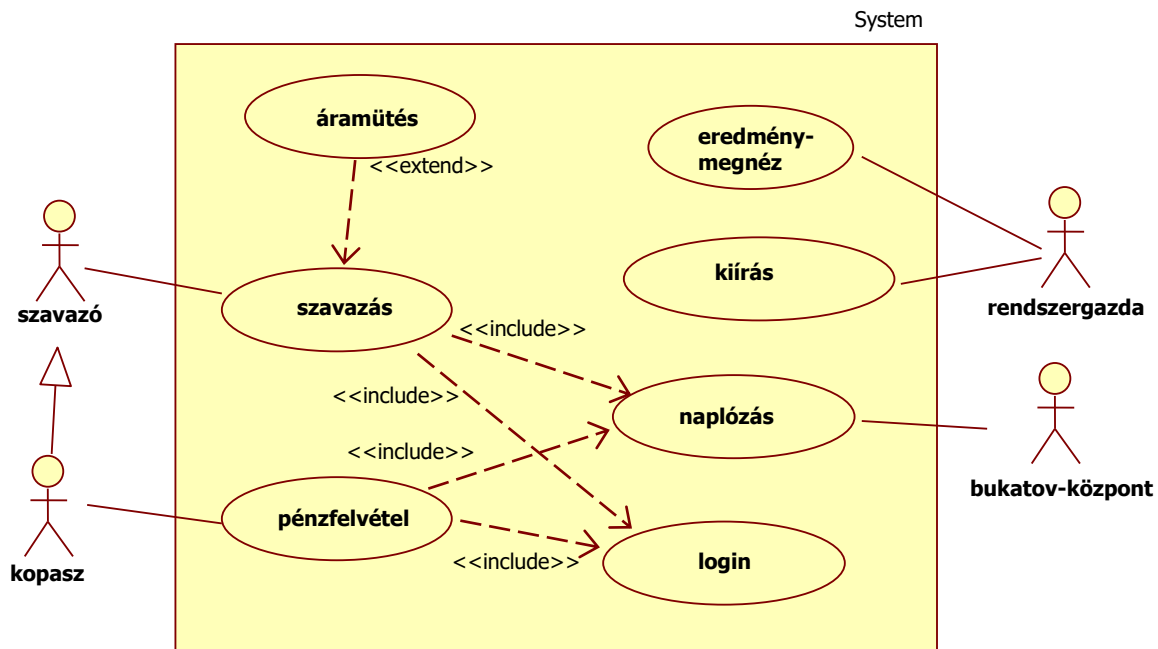
Rajzoljon UML 2 use-case diagramot az alábbi leírás alapján!

A Poseidon rendszert egyetemi oktatók és hallgatók használhatják a tanulmányi adatok rögzítésére. A hallgatók bejelentkezhettek, vizsgára jelentkezhetnek és órarendet nyomtathatnak. Ha vizsgára jelentkezés közben kiderül, hogy valamely előfeltétel nem teljesül, akkor a rendszer rögzíti a csalási kísérletet. Az oktatók is belépnek, vizsgát írhatnak ki, illetve vizsgaeredményeket írhatnak be. Az adminisztrátorok az oktatók lehetőségein túl még termet is foglalhatnak. A doktoranduszok, mivel oktatniuk és tanulniuk is kell, mind oktatóként, mind hallgatóként használhatják a rendszert.



Az alábbi történet alapján készítsen UML2 használati eset (use-case) diagramot!

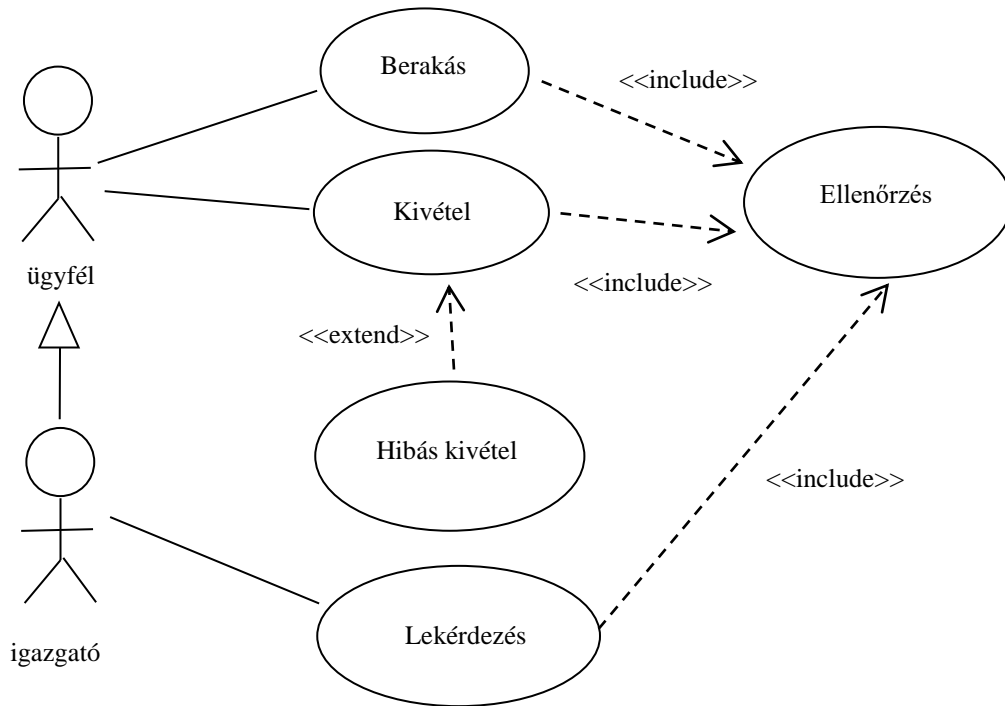
Bergengóciában elektronikus szavazógépet készítenek. A szavazógépet szavazók használják szavazásra. Ha a szavazó rosszul szavaz, megüti az áram. A szavazók egy része ú.n. "kopasz" szavazó, ők pénzt is fel tudnak venni a gépből. A szavazás és a pénzfelvétel mind részletes naplózásra kerül a nemzetbiztonság erősítése végett. A naplóbejegyzéseket a gép azonnal továbbítja a Bukatov-központba. A szavazáshoz és a pénzfelvételhez be kell lépni a rendszerbe. Az aranykoszorús rendszergazdák ki tudnak írni szavazást és meg tudják nézni a végeredményt.





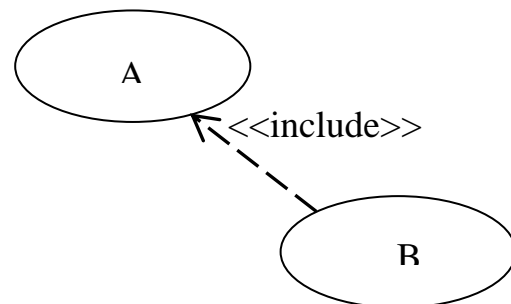
Rajzoljon UML2 use-case diagramot az alábbi történet alapján!

A PickPack Raktárszolgáltató Kft bivalypasnádi telepén hatalmas polcrendszerekben tárolják a csomagokat. Ha egy regisztrált ügyfél csomagot hoz tárolási célból, ellenőrzik az azonosítóját, majd robotok egy üres helyre teszik a csomagot. Mikor az ügyfél elvinné egyik csomagját, ismét ellenőrzik az azonosítóját, majd előhozzák a kívánt csomagot. Előfordulhat, hogy a csomag nincs a helyén, mert már korábban elvitték. Ebben az esetben a rendszer figyelmezteti az ügyfelet a hibára. A raktár igazgatója mindazt meg tudja tenni, amit egy egyszerű ügyfél, de ő lekérdezheti a telepen tárolt csomagok számát is. Ehhez természetesen az ő azonosítóját is ellenőrzik.

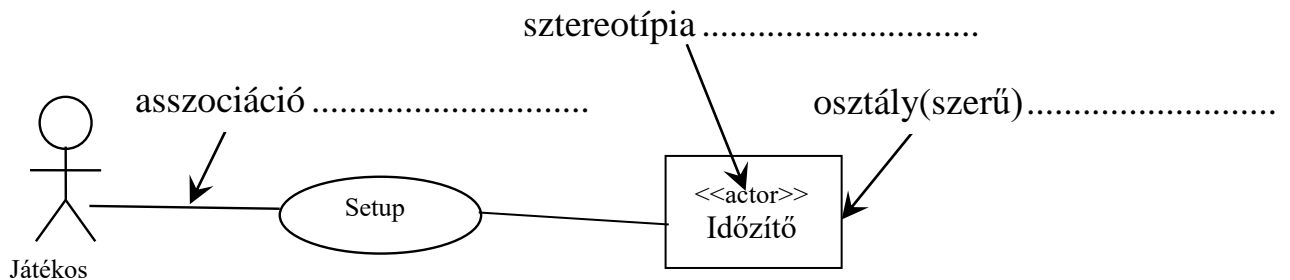


Adott az alábbi UML2 use-case (használati eset) diagram részlet. Jelölje be a felsoroltak közül az igaz állítást!

- ☐ A kötelezően tartalmazza B-t
- ☐ A esetleg tartalmazza B-t
- ☒ B kötelezően tartalmazza A-t
- ☐ B esetleg tartalmazza A-t



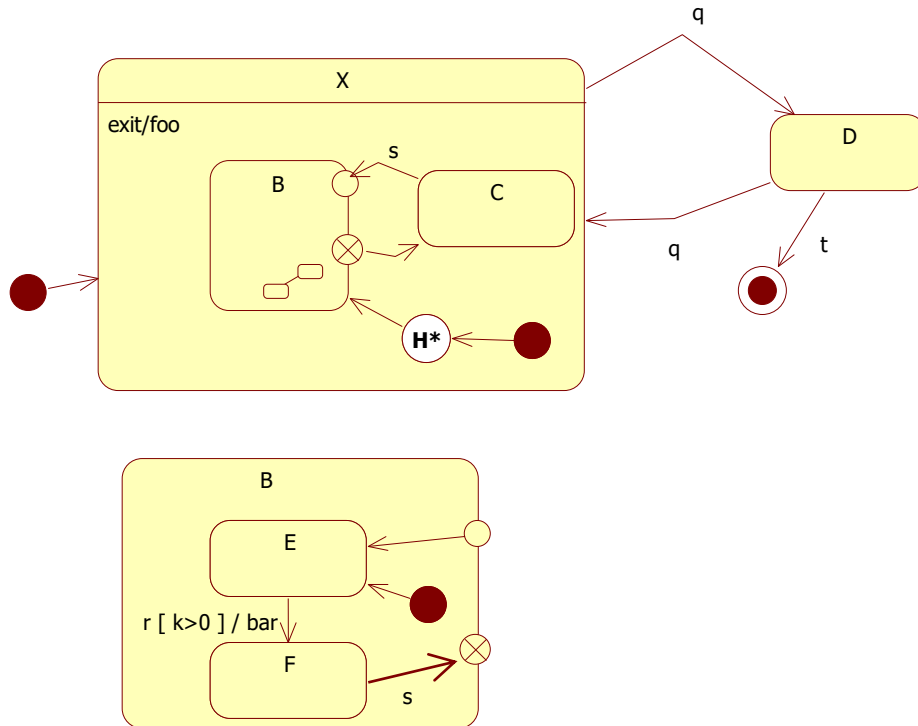
Az alábbi ábrán három UML2 modell elemet megjelöltünk. Adja meg elemenként, hogy az melyik UML2 meta-modell elem példánya !



## 4 Állapotátmenet diagram

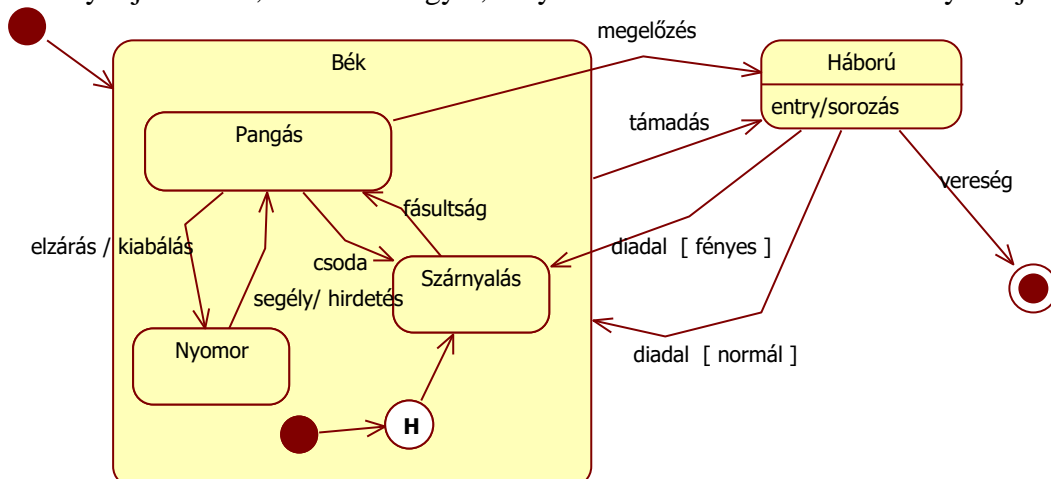
Készítsen UML állapotátmeneti diagramot (state chart) az alábbi specifikáció alapján!

Egy állapotgép **X**, **B**, **C**, **D**, **E** és **F** állapotokkal rendelkezik. Az **X**-nek alállapota **B** és **C**, a **B**-nek alállapota **E** és **F**. A **B** állapot *belső* működését ábrázolja az **X** dobozán kívül! A kezdőállapot az **E**. Az **X** állapot elhagyásakor mindig lezajlik a **foo** akció. A **C** állapotból **E**-be az **s** esemény bekövetkeztekor jutunk. **F** állapotból **C**-be szintén **s** esetén jutunk. Az **X** és a **D** közötti átmenethez (mindkét irányban) a **q** eseménynek kell bekövetkeznie. Amikor **X**-be visszatérünk, mindig ugyanabba a belső állapotba kerülünk, mint amit **X**-ből kilépve elhagytunk (tetszőleges mélységig). **E**-ből **F**-be az **r** esemény hatására akkor juthatunk át, ha **k** értéke pozitív, ekkor mindig lefut a **bar** akció is. A **D** állapotban **t** esemény bekövetkezésekor az állapotgép megáll.



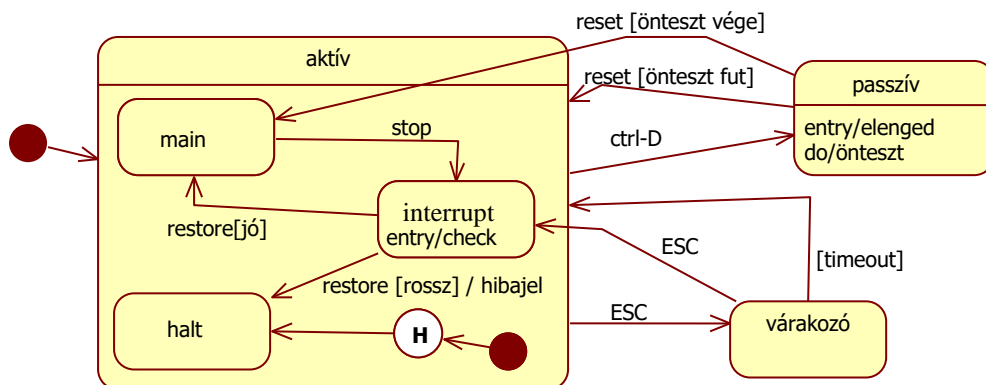
Rajzoljon UML2 állapotdiagramot az alábbi leírás alapján!

Kanyarország vagy békében él, vagy háborúzik. Béke idején nyomor, pangás és szárnyalás lehet. Az ország megalakulásakor szárnyalt. Ha beüt a fásultság, akkor jön a pangás, ahonnan a csoda újra szárnyaláshoz vezet. Pangásból, a segélyek elzárása esetén kiabálnak, és jön a nyomor, ha újra van segély, akkor kihirdetik, hogy "ez nekünk jár!", és az ország visszajut a pangásba. Pangás esetén, megelőző jelleggel háborúba lehet lépni. Békéből támadás esetén automatikusan háborúba lép az ország. A háború minden esetben (itt nem részletezettekben is) sorozással kezdődik. Ha a háborút elveszti, az országnak vége. Ha diadalt arat, akkor normál esetben ott folytatja a békét, ahol abbagyta, fényes diadal esetén azonban szárnyalás jön.



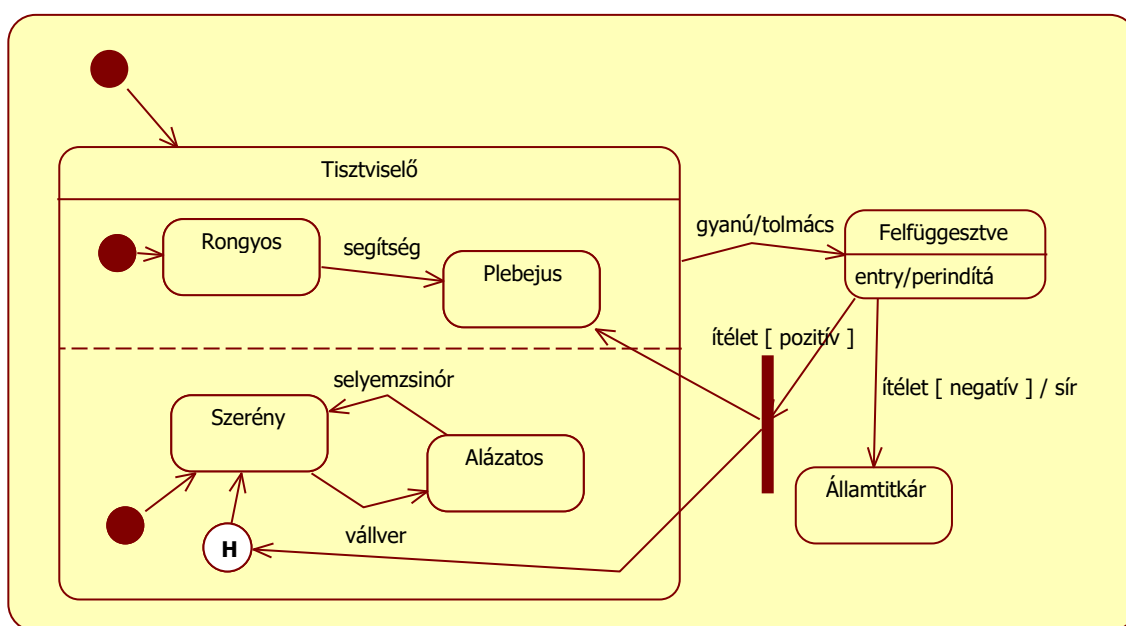
Rajzoljon UML2 állapot diagramot!

A Dowswin program futása során aktív, passzív és várakozó lehet. Aktivitás közben lehet main, interrupt és halt helyzetben. Main-ből a stop hatására interrupt-ba megy. Az interrupt-ba belépéskor lefuttat egy memory check-et. Az interrupt-ból a restore hatására lép ki. Ha a belépéskor végrehajtott memory check jó volt, akkor main-be lép, ha nem, akkor halt-ba. Halt-ba lépés közben hibajelzést küld a konzolra. Aktivívból ctrl-D hatására passzíválódik. Passzív állapotba lépéskor elengedi az erőforrásokat, majd egy hosszantartó öntesztelést végez. Reset hatására újra aktivizálódik. Ha az öntesztelés befejeződött, akkor main-be lép, ha nem, akkor a ctrl-D érkezése előtti helyzetbe tér vissza. Aktivitásból várakozásba megy át, ha ESC érkezik. A várakozás végetér egy újabb ESC előfordulásakor. Ekkor a működést interrupt-ban folytatja. A várakozás egy timeout letelte után szintén befejeződik. Ez esetben a várakozást megelőző helyzetben folytatódik a működés. A program halt helyzetből indul.



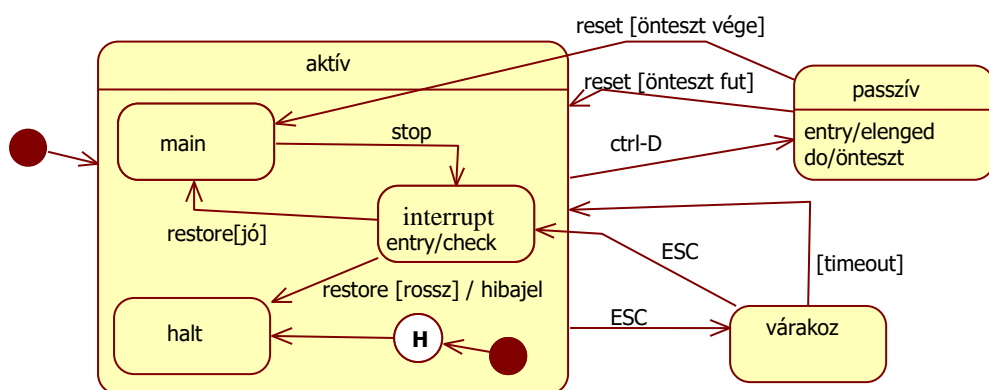
Készítsen UML2 állapotdiagramot (statechart) az alábbi történet alapján!

Lilliputban a tisztségviselők rongyosan kezdik tevékenységüket, ekkor még szerények is. Ha családi segítséget kapnak, akkor plebejussá válnak. A szerény tisztségviselő, ha vállon veregetik, alázatos lesz. Ha alázatos, és selyemzsinórt kap, újra szerénnyé válik. Ha a tisztségviselőt gyanú éri, akkor tolmácsot kér és felfüggesztett állapotba kerül. Felfüggesztéskor pert indít. A bírósági ítélet alapján (pozitív esetben) újra tisztségviselő lesz (mindig plebejus, valamint vagy szerény, vagy alázatos, attól függően, hogy a felfüggesztés előtt mi volt). Ha a bírósági ítélet negatív, akkor elsírja magát, és államtitkárrá léptetik elő.

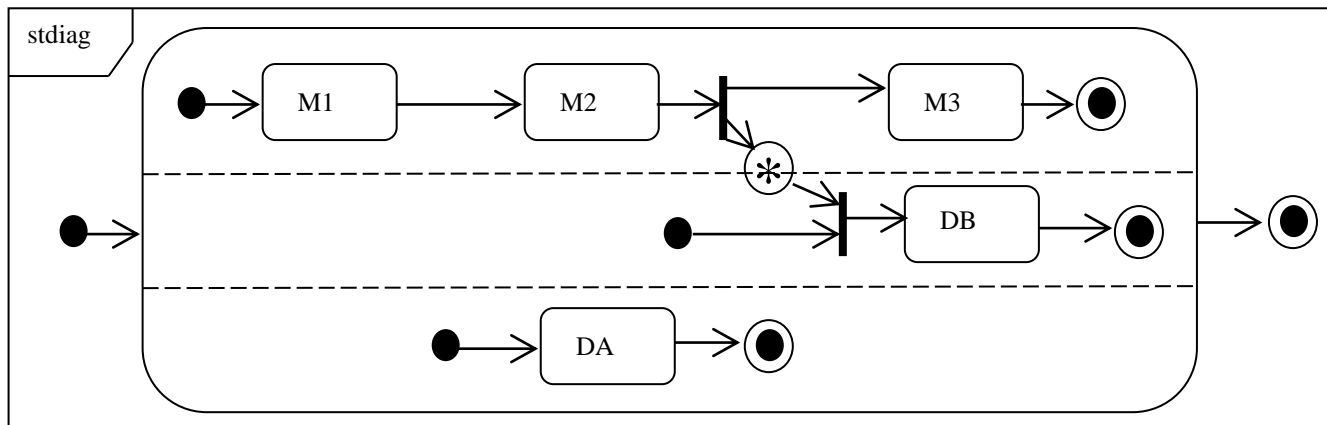


Rajzoljon UML2 állapot diagramot!

A Dowswin program futása során aktív, passzív és várakozó lehet. Aktivitás közben lehet main, interrupt és halt helyzetben. Main-ből a stop hatására interrupt-ba megy. Az interrupt-ba belépéskor lefuttat egy memory check-et. Az interrupt-ból a restore hatására lép ki. Ha a belépéskor végrehajtott memory check jó volt, akkor main-be lép, ha nem, akkor halt-ba. Halt-ba lépés közben hibajelzést küld a konzolra. Aktívból ctrl-D hatására passzíválódik. Passzív állapotba lépéskor elengedi az erőforrásokat, majd egy hosszantartó öntesztelést végez. Reset hatására újra aktivizálódik. Ha az öntesztelés befejeződött, akkor main-be lép, ha nem, akkor a ctrl-D érkezése előtti helyzetbe tér vissza. Aktivitásból várakozásba megy át, ha ESC érkezik. A várakozás végetér egy újabb ESC előfordulásakor. Ekkor a működést interrupt-ban folytatja. A várakozás egy timeout letelte után szintén befejeződik. Ez esetben a várakozást megelőző helyzetben folytatódik a működés. A program halt helyzetből indul.

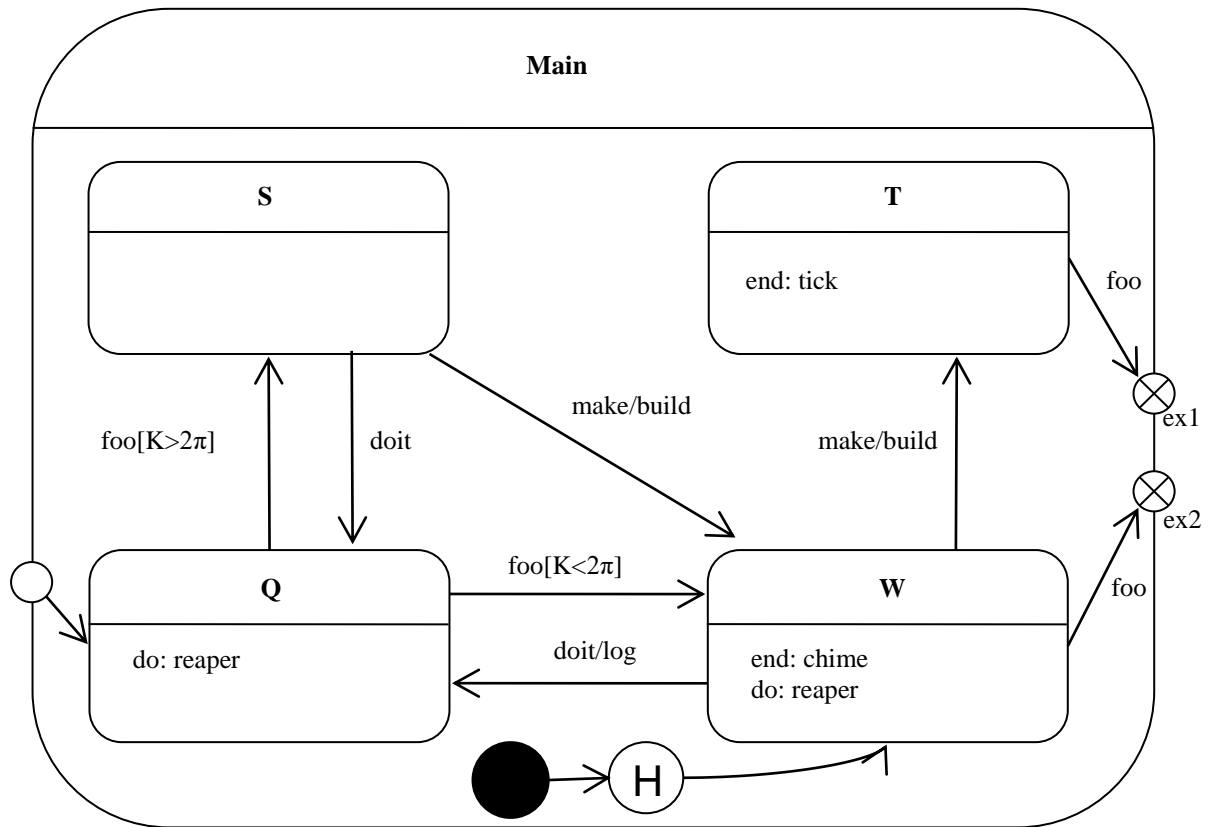


Egy tantárgy teljesítéséhez három mérést (M1, M2, M3) kell elvégezni, és két házi dolgozatot kell beadni (DA, DB). A méréseket szigorúan szám szerinti sorrendben kell elvégezni, a két dolgozat tetszőleges sorrendben készíthető el. A DB dolgozat csak az M2 mérés elvégzését követően készíthető el. Rajzoljon UML2 állapot modellt!



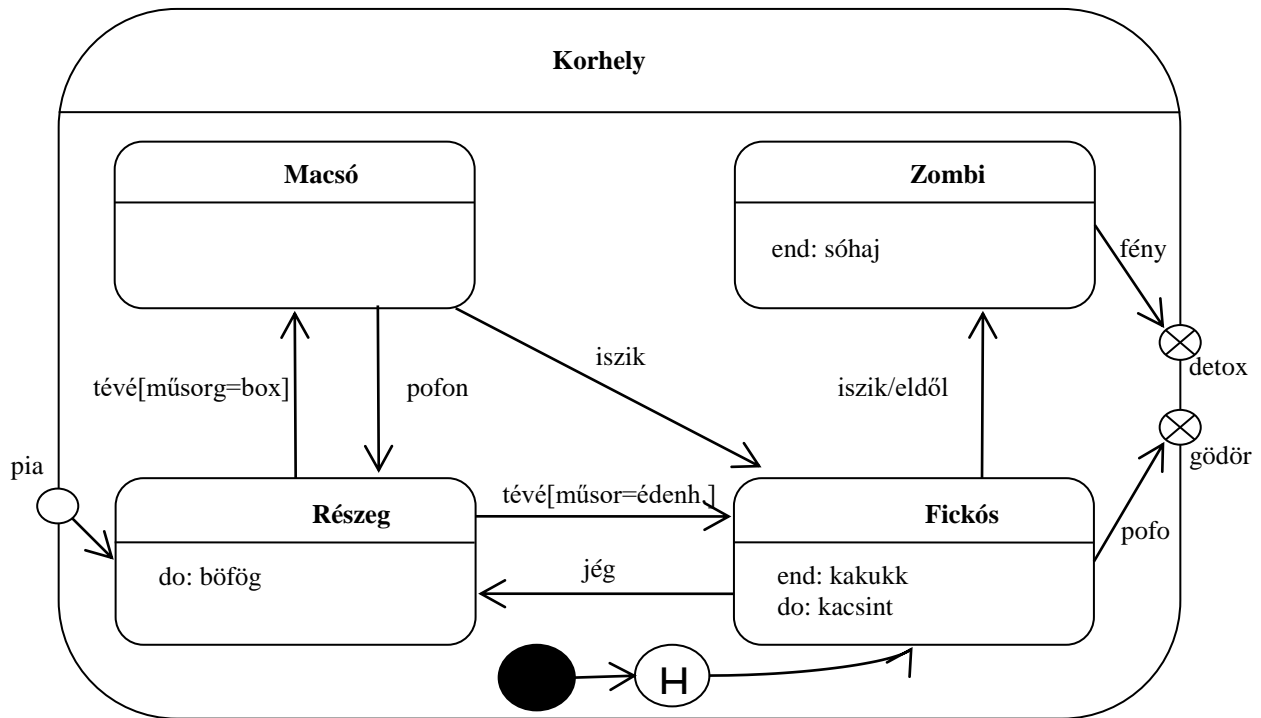
Egészítse ki az alábbi UML 2 állapotdiagramot (state chart) a következő leírás alapján!

Egy objektum **Main** állapotában 4 állapottal találhatók (**S**, **T**, **Q**, **W**). A **Main** állapotba egy belépési ponton (entry point) léphetünk be (**en1**), és két kilépési ponton (exit point) hagyhatjuk el (**ex1**, **ex2**). Ha nem **en1**-en lépünk be, akkor abba az állapotba kerülünk, amelyikben utoljára voltunk. Ha nem volt ilyen, akkor a **W**-be. Az **en1**-ből a **Q**-ba kerülünk. Ha **Q**-ban **foo** esemény éri, akkor attól függően, hogy **K** értéke kisebb, mint  $2\pi$  vagy sem, rendre a **W** vagy az **S** állapotba kerül. Mindkét állapot a **doit** és az **make** események hatására hagyható el. Előbbi esemény esetén visszatér **Q**-ba, utóbbinál pedig (lefuttatva a **build** metódust) **W**-ből **T**-be, **S**-ből **W**-be kerül. **T**-ből kilépéskor mindig lefut a **tick** metódus. **W**-ből a **doit** eseményre történő állapotváltás során a **log** metódus hívódik meg. **Q** állapotban a **reaper** metódus fut. A **foo** esemény hatására **T**-ből **ex1**-be, **W**-ből **ex2**-be lépünk.



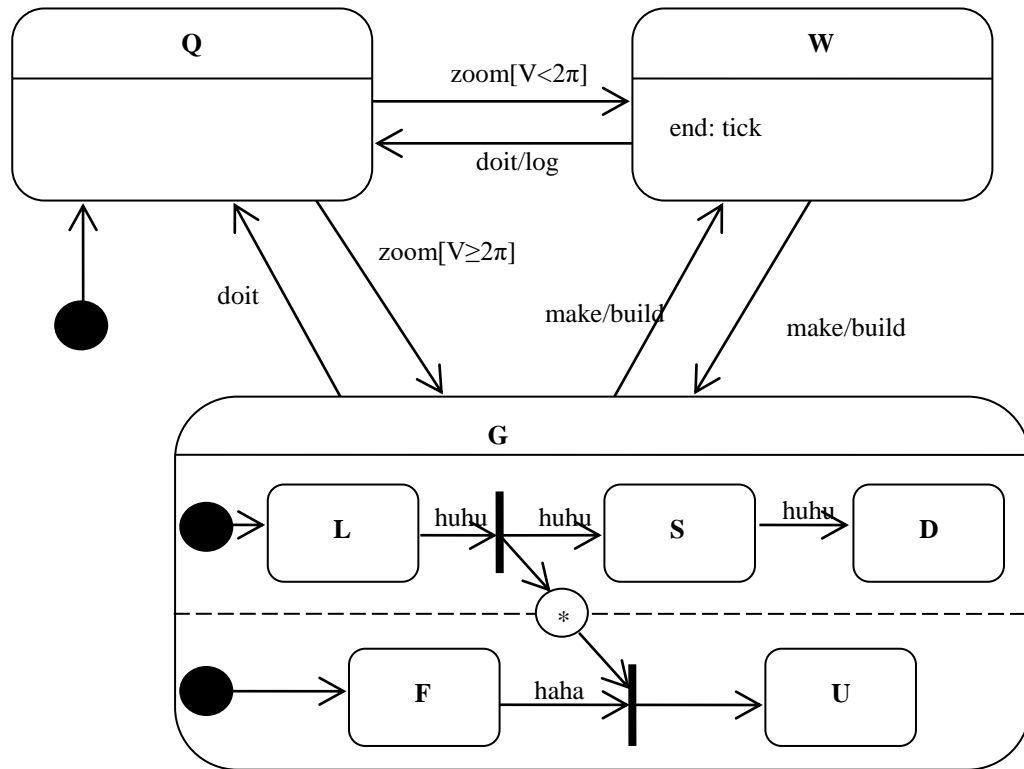
Egészítse ki az alábbi UML 2 állapotdiagramot (state chart) a következő leírás alapján!

Egy objektum **Korhely** állapotában 4 alállapot található (**részeg**, **fickós**, **macsó**, **zombi**). A **korhely** állapotba egy belépési ponton (entry point) léphetünk be (**pia**), és két kilépési ponton (exit point) hagyhatjuk el (**detox**, **gödör**). Ha nem pia-n lépünk be, akkor abba az állapotba kerülünk, amelyikben utoljára voltunk. Ha nem volt ilyen, akkor a fickósba. A pia-ból a részegbe kerülünk. Ha részeg, folyton böfög. Ilyenkor, ha tévét néz, és box megy, akkor macsó lesz, ha az éden hotel megy, akkor meg fickós. Fickósan mindig kacsint. Fickósból részeg jég hatására, macsóból részeg pofon hatására lesz. Ha macsó és iszik, akkor fickós lesz. Bármikor, amikor abbamarad a fickósság, akkor kakukkol. Ha fickós és iszik, akkor eldől és zombi lesz. Ha fickós és pofont kap, akkor a gödör kilépési ponthoz jut. Zombi állapotból kilépve sóhaj. Zombiból fény hatására a detox-ba kerül.



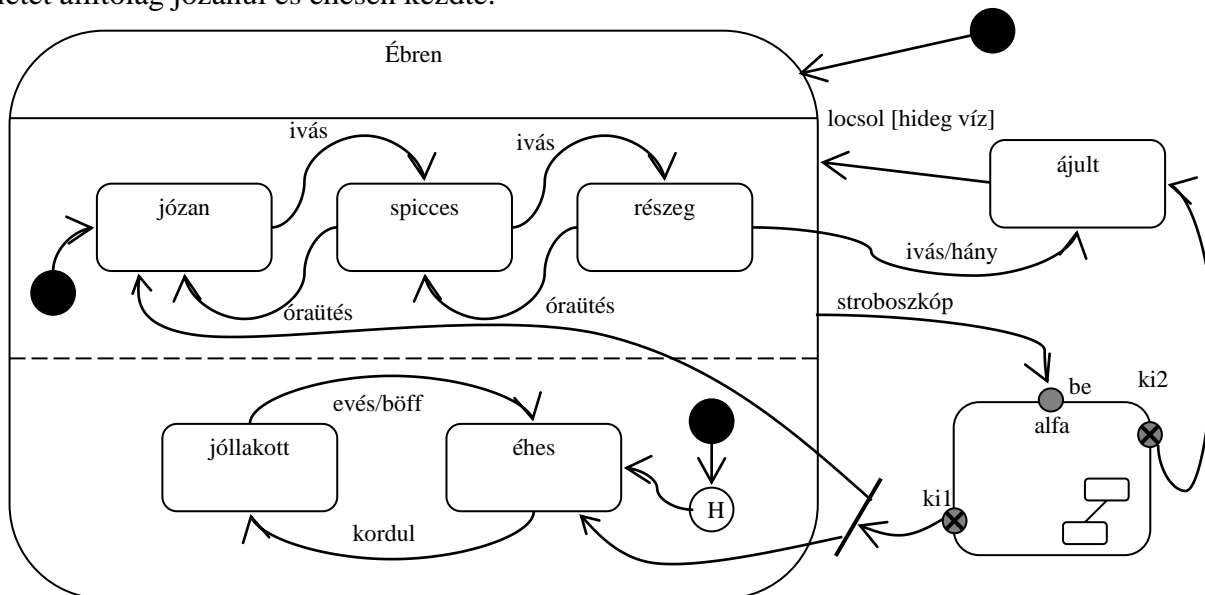
Egészítse ki az alábbi UML 2 állapotdiagramot (state chart) a következő leírás alapján!

Egy objektum három fő állapottal (**Q**, **W**, **G**) rendelkezik. A kezdőállapot a **Q**. Ha **Q**-ban **zoom** esemény éri, akkor attól függően, hogy **V** értéke kisebb, mint  $2\pi$  vagy sem, rendre a **W** vagy a **G** állapotba kerül. Mindkét állapot a **doit** és az **make** események hatására hagyható el. Előbbi esemény esetén visszatér **Q**-ba, utóbbinál pedig (lefuttatva a **build** metódust) **W**-ből **G**-be, **G**-ből **W**-be kerül. **W**-t elhagyva a **tick** metódus hívódik meg. **W**-ből a **doit** eseményre történő állapotváltás során a **log** metódus hívódik meg. **G** állapotban öt alállapot van, amelyek két, független csoportba oszthatók (**L**, **S**, **D**, és **F**, **U**, a csoportok első tagjában kezdünk). **L**-ből **S**-be, **S**-ből **D**-be jutunk a **huhu** esemény hatására. **F**-ből **U**-ba kerülhetünk a **haha** eseményre. Kapcsolat annyiban van köztük, hogy **U**-ba csak akkor kerülhetünk, ha már elhagytuk **L**-t.



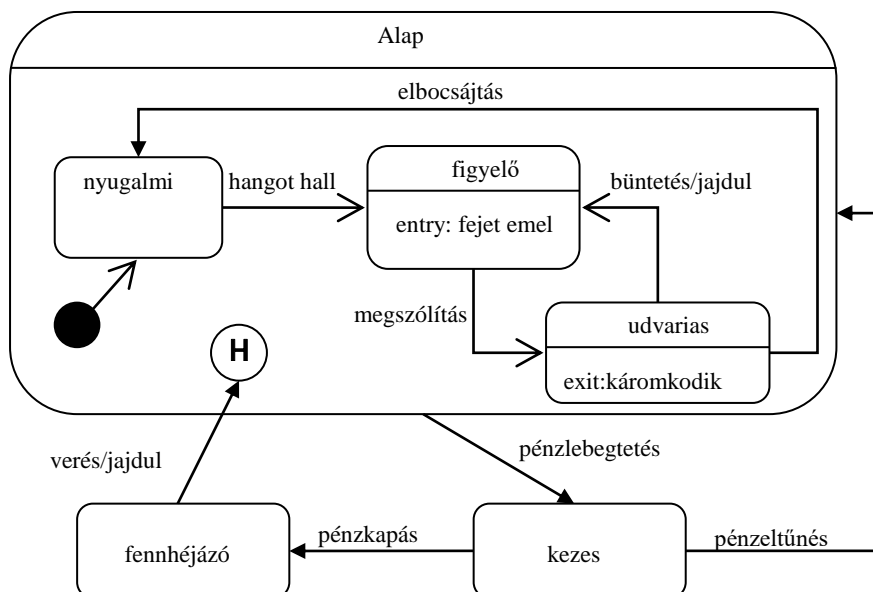
Rajzoljon UML 2.0 állapotábrát (state chart) az alábbi történet alapján!

Stux ébren háromféle hangulatban lehet: józan, spicces, részeg. Ezen kívül (szintén ébren) lehet éhes vagy jóllakott. Értelemszerűen, ha józan és iszik, akkor spicces, ha megint iszik, részeg lesz. Óraütésre visszafelé változik. Ha éhes és eszik, akkor elbőffenti magát és jóllakott lesz. Akkor éhez meg, mikor megkordul a gyomra. Ha részeg, és még iszik, akkor elhányja magát és elájul, amely állapotban sem az éhséget, sem a jóllakottságot nem érzi. Ha ájult, akkor addig marad így, amíg le nem locsolják, de csak hideg vízre reagál. Ájultából kelve mindig józan lesz, de az éhsége vagy jóllakottsága nem változik. Mindezekon kívül le tud menni alfába. Ennek az állapotnak a belsejéről annyit tudunk, hogy összetett, de többet nem. Egy entry (be) és két exit pontja (ki1, ki2) van. Éber állapotból kerülhet ide, ha stroboszkópba néz. Hogy hogyan jön ki belőle, arról csak annyi bizonyos, hogy a ki1 pontból józanul és éhesen jön elő, a ki2-n keresztül pedig elájul. Az életét állítólag józanul és éhesen kezdte.



Az alábbi történet alapján rajzoljon a diagramba UML 2 állapotábrát (state chart)!

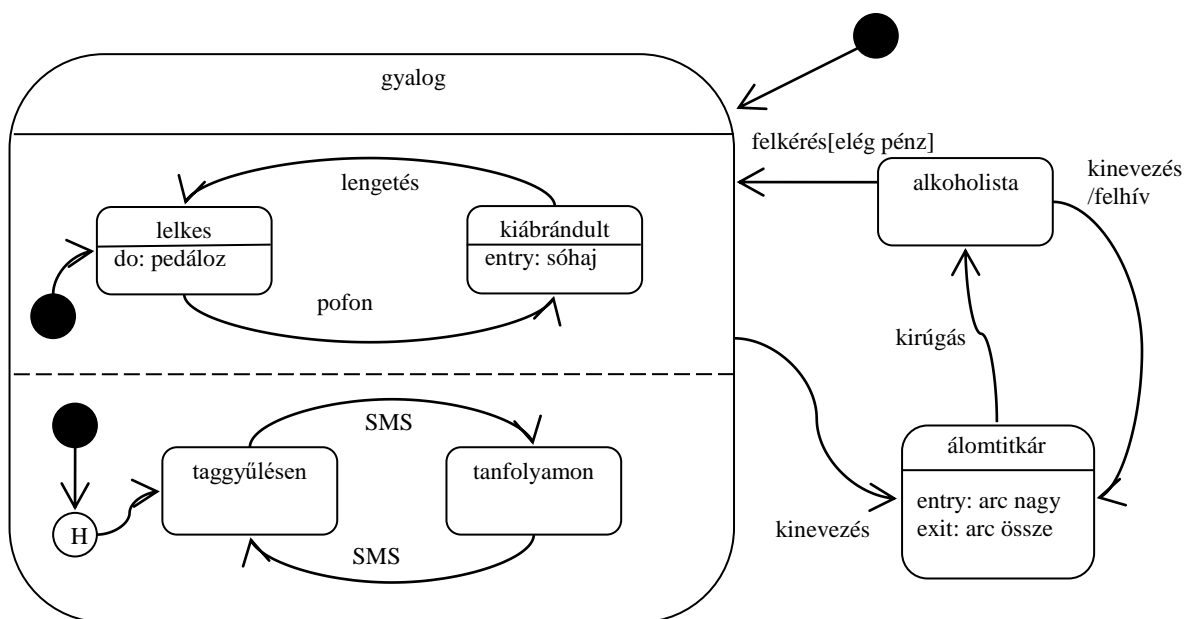
A Titanon létezik egy Lowyir-nek keresztelt parazita életforma. A Lowyir életét nyugalmi állapotban kezdi. Ha hangot hall, figyelő állásba lép (figyelő állás kezdetén mindig felemeli a fejét). Ha ekkor megszólítják, akkor udvarias lesz. Udvariasságából két módon lehet kimozdítani. Elbocsájtással, amire ismét nyugalmi állapotba kerül, vagy büntetéssel, ekkor megint figyelő állásba lép, de előtte feljajdul. Az udvarias állapotból való kizökkentéskor mindig elkáromkodja magát. Bármely fenti állapotban is volt, ha pénzt lebegtetnek meg előtte, akkor kezessé válik, ha a pénz eltűnik, a nyugalmi helyzetét veszi fel. Ha kezés és a pénzből kap, akkor fennhéjázó lesz. Fennhéjázása csak akkor szűnik meg, ha megverik, ekkor feljajdul, de aztán ott folytatja, ahol a pénz meglebegtetése előtt abbahagyta.





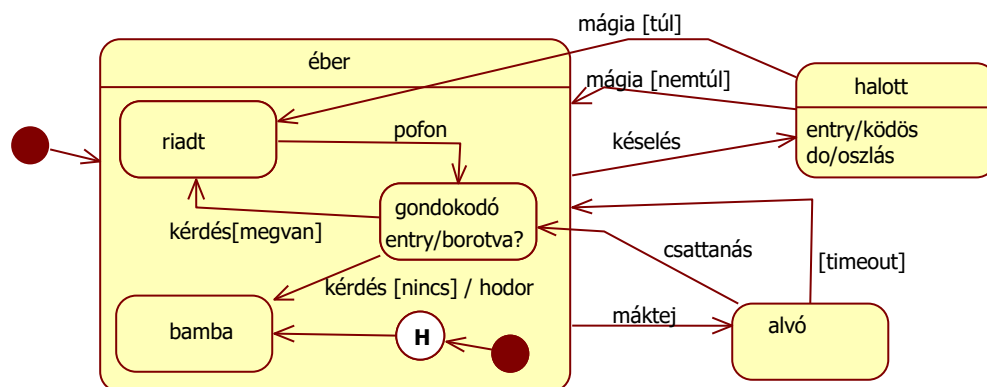
Rajzoljon UML 2.0 állapotábrát (state chart) az alábbi történet alapján!

A Stupiditas nevű szervezet tagja *gyalogként* (másként *paraszt*) kezdi pályafutását. Először nagyon *lelkes*, ilyenkor folyton pedálozik. Ha nagy pofont kap, *kiábrándul* (és elsóhajtja magát). Némi állami támogatás belengetésével ismét lelkes lesz (és pedálozik). Mindeközben (vagyis hogy éppen lelkes vagy kiábrándult), csak két összejegyzetelen lehet megtalálni: *taggyűlésen* és *tanfolyamon* (a taggyűlés az első). Ha az egyiken SMS-t kap, átmegy a másikra. Mikor kinevezik *álomtitkárnak*, akkor maga mögött hagyja a gyalogos életet (hiszen nagy fekete autót is kap). Álomtitkárként először az arca lesz nagy. Amikor kirúgják „állásából”, az arca összemegy és *alkoholista* lesz. Ekkor felkérésre, ha elég pénzt kap (az alkoholizmust levetkőzve) ismét gyalog lesz. Itt mindenképpen lelkesen azon az összejegyzetelen folytatja, ahol utoljára gyalogként megfordult. Az alkoholizmusból egy újabb álomtitkári kinevezés is kigyógyítja. Ilyenkor felhívja anyukáját.



Rajzoljon UML2 állapot diagramot!

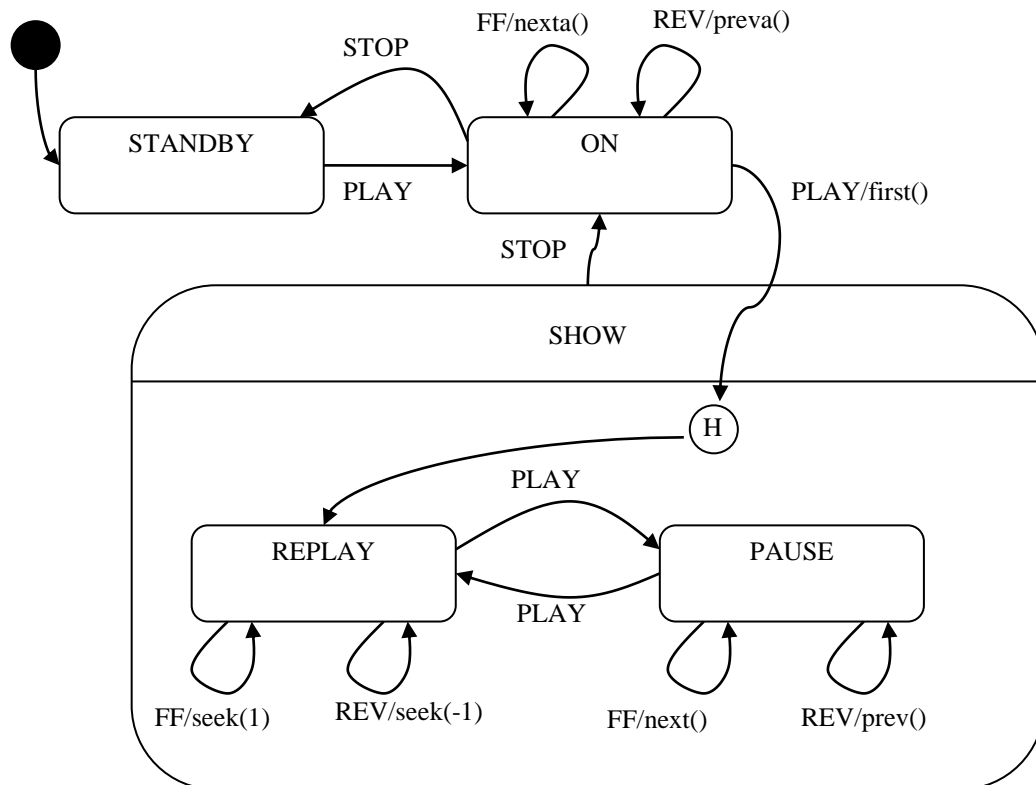
Jon's Now létezése során *éber*, *halott* és *alvó* lehet. Ébren lehet *riadt*, *gondokodó* és *bamba* helyzetben. Riadtból pofon hatására gondolkodóba megy. A gondolkodó állapotba kerüléskor megnézi, hogy nála van-e a borotvája. A gondolkodásból kérdés zökkenti ki. Ha a borotvája megvolt, akkor riadt lesz, ha nem, akkor bamba. Bambává váláskor azt mondja, "hodor". Éber állapotban késelés hatására halott lesz. Amikor halott lesz, ködös lesz a tekintete, és lassan elkezd oszlan. Mágia hatására újra éber lesz. Ha az oszlás nem túl előrehaladott, akkor a halál előtti helyzetbe tér vissza, egyébként riadt lesz. Éberből alvóba megy át, ha máktejet iszik. Az alvás véget ér, amikor csattanást hall. Ekkor gondolkodó állapotba kerül. Az alvás egy idő letelte után szintén befejeződik. Ez esetben az alvást megelőző helyzetbe kerül. Létezését bamba helyzetből kezdi.



Rajzoljon UML2 state-chartot (állapot-diagram) az alábbi történet alapján !

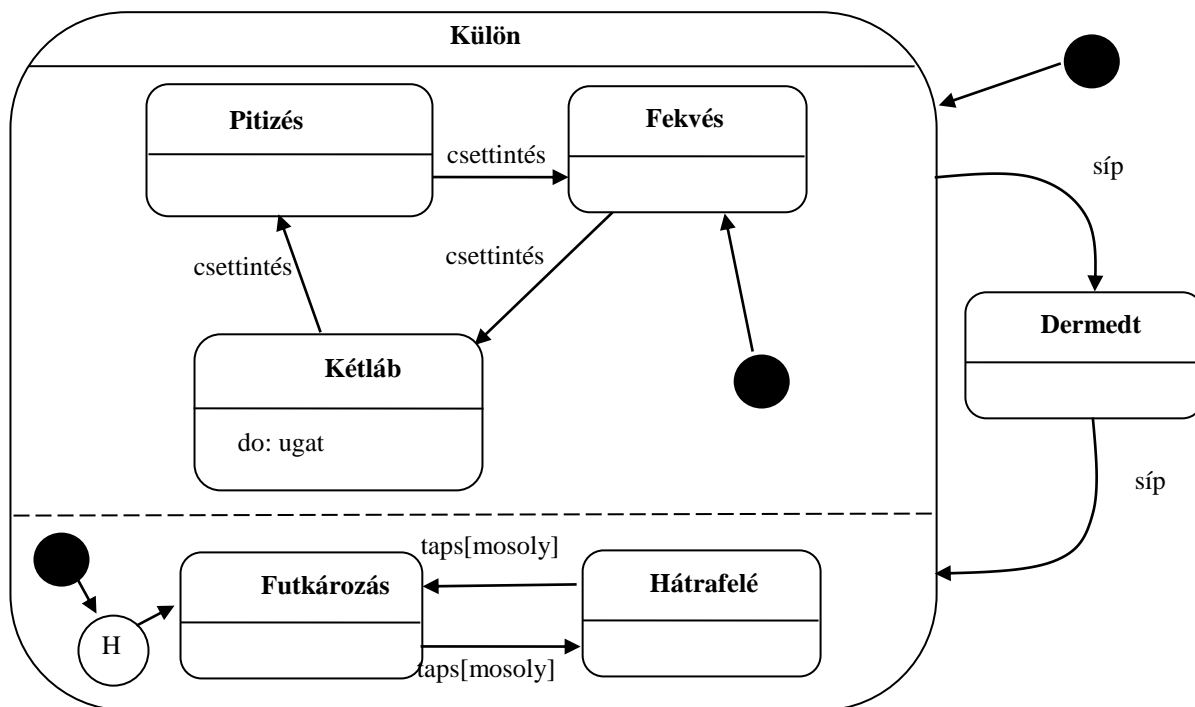
A Dárembéz MP3 lejátszón 4 gomb van: PLAY, STOP, FF, REV. Amikor elemet teszünk bele, akkor STANDBY állapotba kerül. PLAY hatására kapcsol be (ON). Ekkor az FF és a REV gombokkal lehet előre- és hátralépni az albumok között. PLAY megnyomására lejátszó (SHOW) módba kerül, amikor vagy az aktuális album első számát kezdi lejátszani (REPLAY), vagy szünetelteti a lejátszást (PAUSE). Hogy melyiket csinálja, az attól függ, hogy utoljára melyiket végezte SHOW módban (ha még egyiket sem, akkor REPLAY az alap). Ha REPLAY alatt nyomkodjuk az FF és a REV gombokat, a számon belül tekerünk előre vagy hátra 1 mp-nyit. Ha PAUSE módban nyomkodjuk őket, akkor a számok között ugrálhatunk. STOP hatására ismét ON-ba kerülünk, újabb STOP-ra STANDBY-ba. SHOW állapotban a PLAY gombbal lehet megállítani (PAUSE) és újraindítani (REPLAY) az aktuálisan játszott számot.

A lejátszó mp3-API-ja a következő függvényeket ismeri: *seek(int x)*: x mp-et előre megy; *next()*, *prev()*: következő, előző számot választja; *nexta()*, *preva()*: következő/előző album; *first()*: album első számára áll.



Rajzoljon UML 2 **állapotdiagramot** (state chart) az a következő leírás alapján a kutyapárról!

A Csinnbumm Cirkusz szerződtette Lali Bohócot. A bohócnak van egy idomított kutyapárja (Csahos és Rühös). Mindkét kutya külön-külön mutatványt tud, de néha ugyanazt csinálják. Csahos csettintésre pitizik, újabb csettintésre fekszik, újabb csettintésre két lábon járva ugat, végül újabb csettintésre ismét pitizik. A műsort a fekvéssel kezdi. Rühös tapsra vált ha, közben a bohóc mosolyog: először futkározik (mindig ezzel kezd), aztán hátrafele megy, majd megint futkározik. Amikor a bohóc megfújja a sípját, a két kutya megdermed, és addig így maradnak, amíg újabb sípszó nem érkezik. Rühös az okosabb, ő ott folytatja, ahol abbahagyta, Csahos mindig fekvéssel kezd.

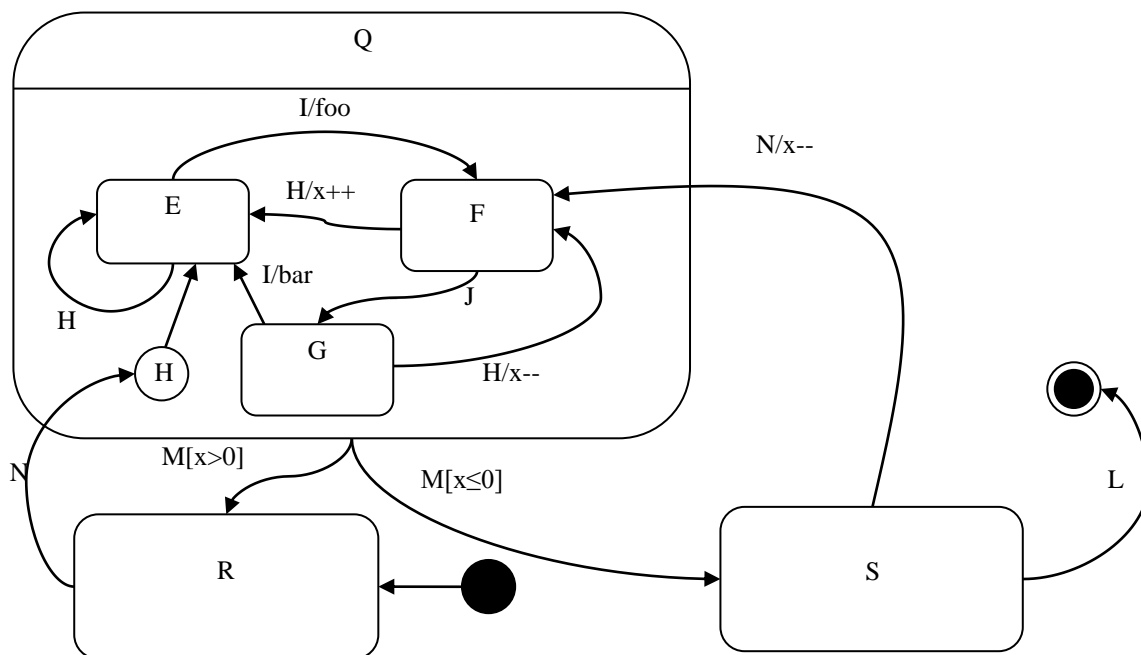


Rajzoljon UML 2 **állapotdiagramot** (state chart) az alábbi leírás alapján!

Egy állapotgép **Q**, **R** és **S** állapotokat tud felvenni. A **Q** állapot alállapotait az alábbi táblázat írja le:

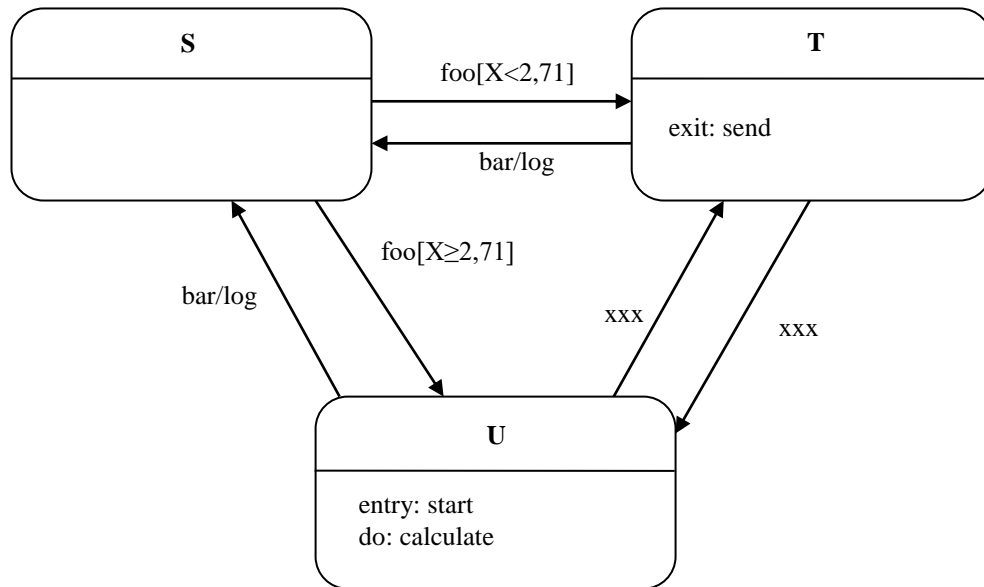
	<b>H</b>	<b>I</b>	<b>J</b>
<b>E</b>	E	F/foo	-
<b>F</b>	E/x++	-	G
<b>G</b>	F/x--	E/bar	-

A **Q**, **R**, **S** átmenetei a következők: **Q**-ból **M** esemény és pozitív **x** hatására **R**-be, nempozitív **x**-re **S**-be kerülünk. **R**-ből **Q**-ba az **N** esemény hatására lépünk, mégpedig abba az alállapotba, amelyik **Q**-ban utoljára aktív volt. Ha ilyen még nem volt, akkor az **E**-be. Az **N** esemény **S**-ből is **Q**-ba visz, de mindig az **F** alállapotba, és ilyenkor **x** értéke is eggyel csökken. Az **L** esemény **S** állapotban az állapotgép leállítását eredményezi. A kezdőállapot az **R**.

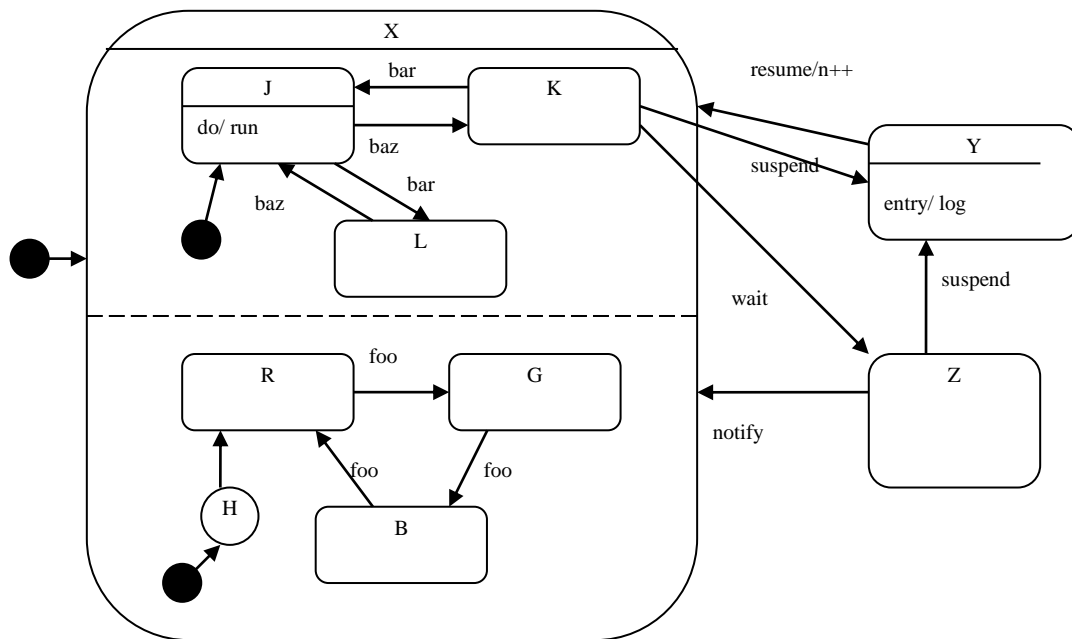


Egészítse ki az alábbi UML 2 állapotdiagramot (state chart) a következő leírás alapján!

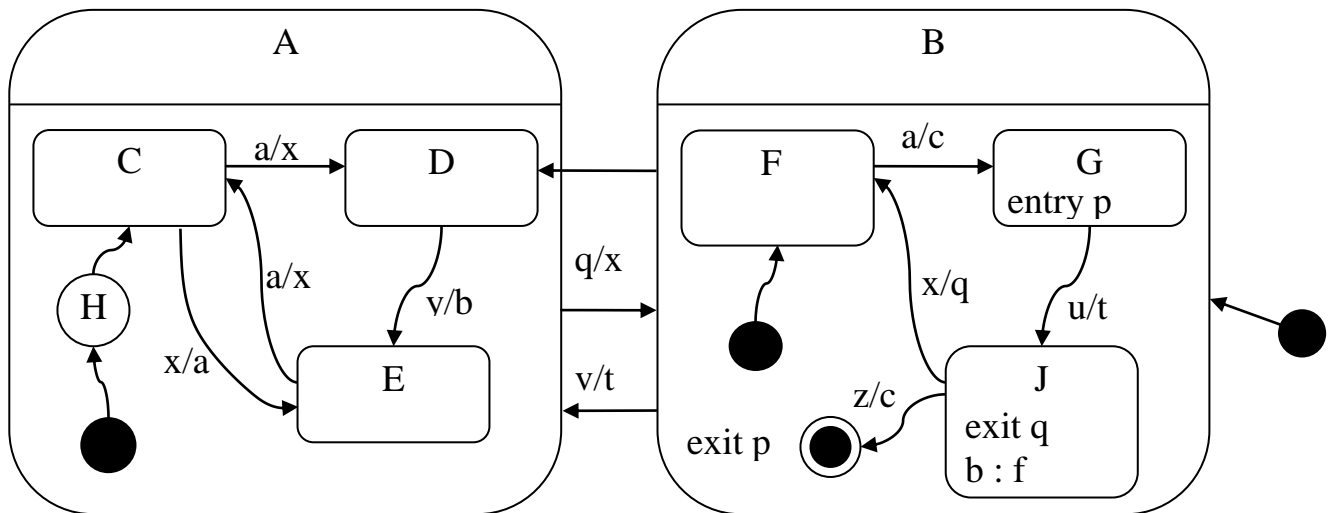
Egy objektum három fő állapottal (**S**, **T**, **U**) rendelkezik. A kezdőállapot az **S**. Ha **S**-ben **foo** esemény éri, akkor attól függően, hogy **X** értéke kisebb, mint 2,71 vagy sem, rendre a **T** vagy az **U** állapotba kerül. Mindkét állapot a **bar** és az **xxx** események hatására hagyható el. Előbbi esemény esetén visszatér **S**-be, utóbbinál pedig **T**-ből **U**-ba, **U**-ból **T**-be kerül. **U**-ba lépéskor mindig lefut a **start** metódus, míg **T**-t elhagyva a **send**. A **bar** eseményre történő állapotváltás során a **log** metódus hívódik meg. Az **U** állapotban tartózkodás közben a **calculate** metódus fut.



Az **O** objektumnak három fő állapota van: **X**, **Y** és **Z**. A kezdő, **X** állapotban felveheti a **J**, **K** vagy **L** alállapot valamelyikét. Ezek közül **J** a kezdő, és ebben az alállapotban folyamatosan fut a **run** metódusa. **J**-ből **K**-ba a **baz**, **L**-be a **bar** eseményre kerül; **L**-ből **J**-be a **baz**, **K**-ból **J**-be a **bar** esemény viszi. Az **X** állapotban a **J-K-L** alállapotoktól függetlenül felveheti az **R**, **G** és **B** állapotok valamelyikét is. A három közül **R**-rel kezd, innen **foo**-ra a **G**-be, **G**-ből **foo**-ra a **B**-be, innen pedig **foo**-ra ismét az **R**-be kerül. Ha a **K** alállapotban van, a **suspend** és a **wait** eseményekre hagyja el az **X** főállapotot, előbbire az **Y**-ba, utóbbira a **Z**-be kerül. **Z**-ből vissza az **X**-be a **notify**, az **Y**-ba a **suspend** viszi. Az **Y** állapotba való minden belépéskor meghívódik a **log** metódus. Az **Y**-ból a **resume** viszi az **X**-be, ekkor az **n** változó értéke eggyel megnő. **X**-be minden belépéskor a **J-K-L** állapotok közül a **J**-vel kezd, az **R-G-B** alállapotok közül azzal, amelyikben az **X**-ből való legutolsó kilépésekor volt. Rajzoljon UML2 állapot diagramot !



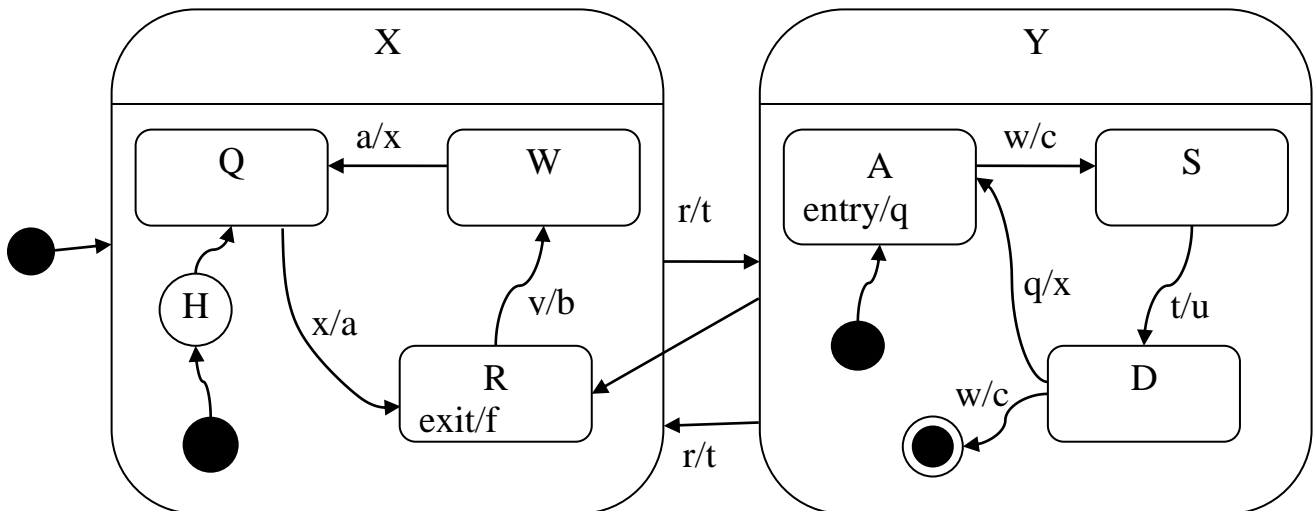
A következő UML2 állapotdiagram alapján minősítse az állításokat!



A kezdés után az **a, u, b, z, a, v** esemény-szekvencia hatására

Igaz	Hamis	Állítás
<input type="checkbox"/>	<input type="checkbox"/>	összesen 5 különböző állapotot érintünk
<input type="checkbox"/>	<input type="checkbox"/>	„b” tevékenységet nem végzünk
<input type="checkbox"/>	<input type="checkbox"/>	C lesz a végállapot
<input type="checkbox"/>	<input type="checkbox"/>	van olyan esemény, amelynek hatására nem végzünk tevékenységet
<input type="checkbox"/>	<input type="checkbox"/>	az E állapotot kétszer érintjük
<input type="checkbox"/>	<input type="checkbox"/>	egyszer lefut az „a” tevékenység

A következő UML2 állapotdiagram alapján minősítse az állításokat!

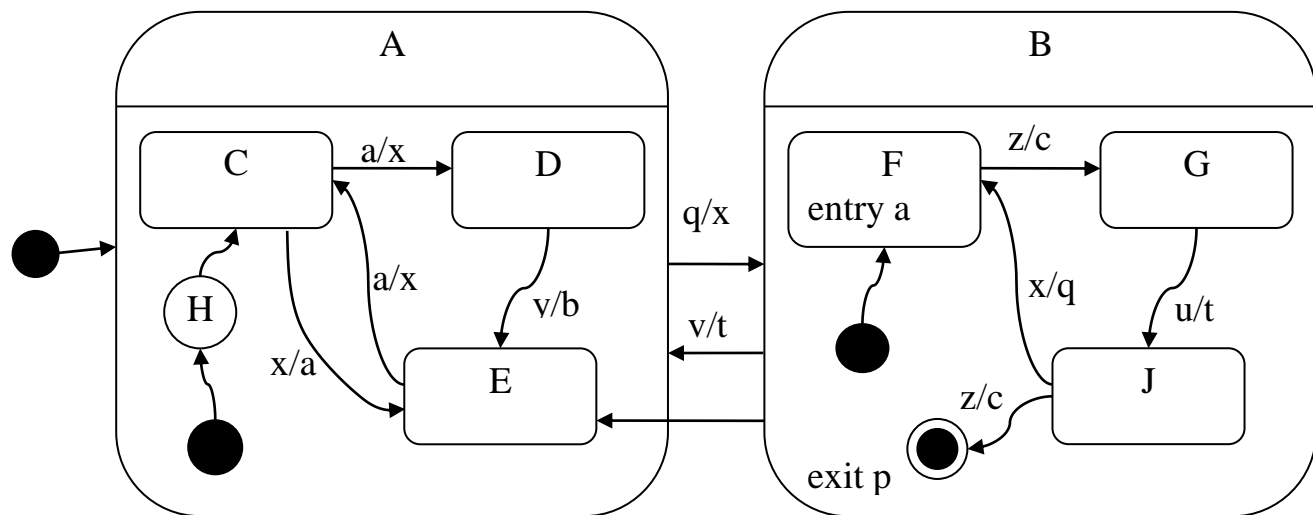


Igaz	Hamis	Állítás
<input type="checkbox"/>	<input type="checkbox"/>	R állapotból 2 lépésben nem lehet visszaérni R-be
<input type="checkbox"/>	<input type="checkbox"/>	A állapotból egyetlen esemény hatására W állapot következhet
<input type="checkbox"/>	<input type="checkbox"/>	S állapotból „r” esemény hatására H állapotba kerül
<input type="checkbox"/>	<input type="checkbox"/>	X-ből Y-ba való váltáskor végrehajtható az „f” tevékenység

A kezdés után az **x, r, w, r** esemény-szekvencia hatására

Igaz	Hamis	Állítás
<input type="checkbox"/>	<input type="checkbox"/>	Kétszer lefut a „t” tevékenység
<input type="checkbox"/>	<input type="checkbox"/>	Q állapotba kerülünk
<input type="checkbox"/>	<input type="checkbox"/>	összesen 6 tevékenység fut le.

A következő UML2 állapotdiagram alapján minősítse az állításokat!



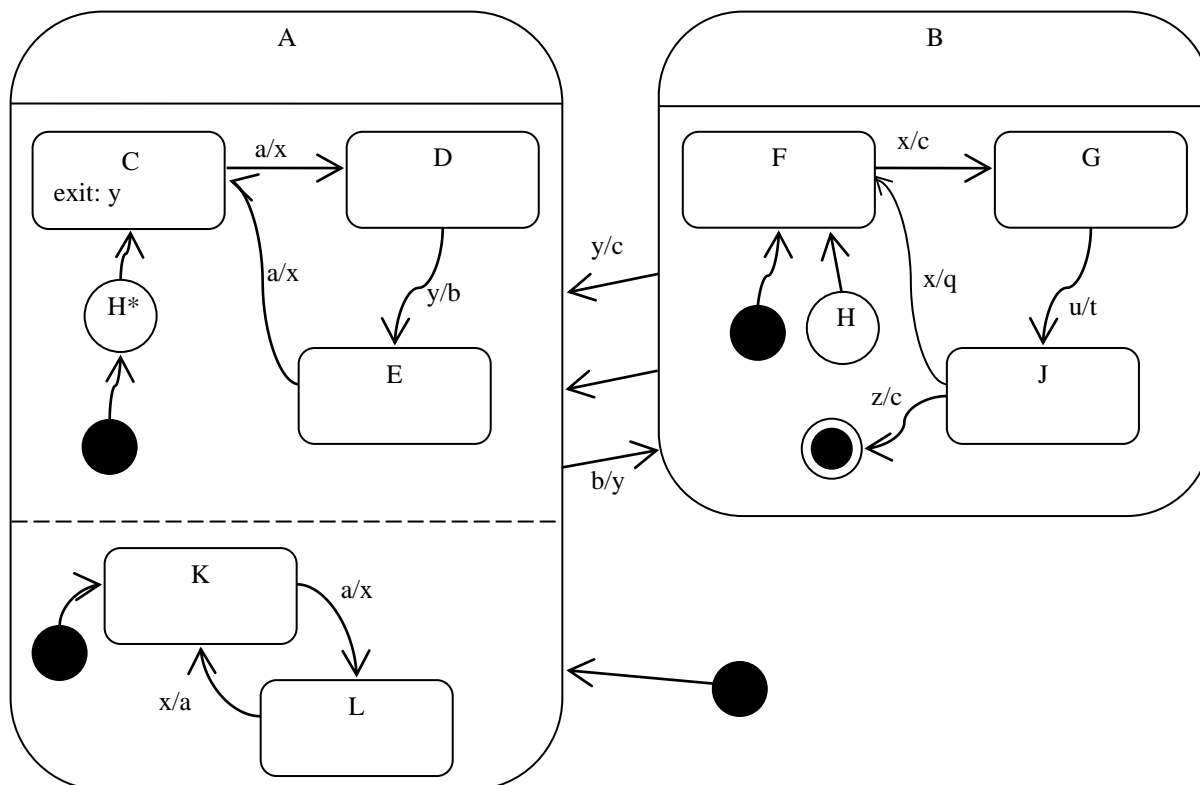
Igaz	Hamis	Állítás
<input type="checkbox"/>	<input type="checkbox"/>	D állapotból 2 lépésben visszaérhet D-be
<input type="checkbox"/>	<input type="checkbox"/>	J állapotból „p” esemény hatására E állapotba kerül
<input type="checkbox"/>	<input type="checkbox"/>	F állapotból „v” esemény hatására H állapotba kerül
<input type="checkbox"/>	<input type="checkbox"/>	A-ból B-be való váltáskor mindig végrehajtódik az „a” tevékenység

A kezdés után az **x, q, z, u, z** esemény-szekvencia hatására

Igaz	Hamis	Állítás
<input type="checkbox"/>	<input type="checkbox"/>	Egyszer lefut a „p” tevékenység
<input type="checkbox"/>	<input type="checkbox"/>	C állapotba kerülünk
<input type="checkbox"/>	<input type="checkbox"/>	Az E állapotot kétszer érintjük
<input type="checkbox"/>	<input type="checkbox"/>	Kétszer lefut az „a” tevékenység



A következő UML állapotdiagram alapján minősítse az állításokat! Csak a rubrikába tett jelzést vesszük figyelembe!

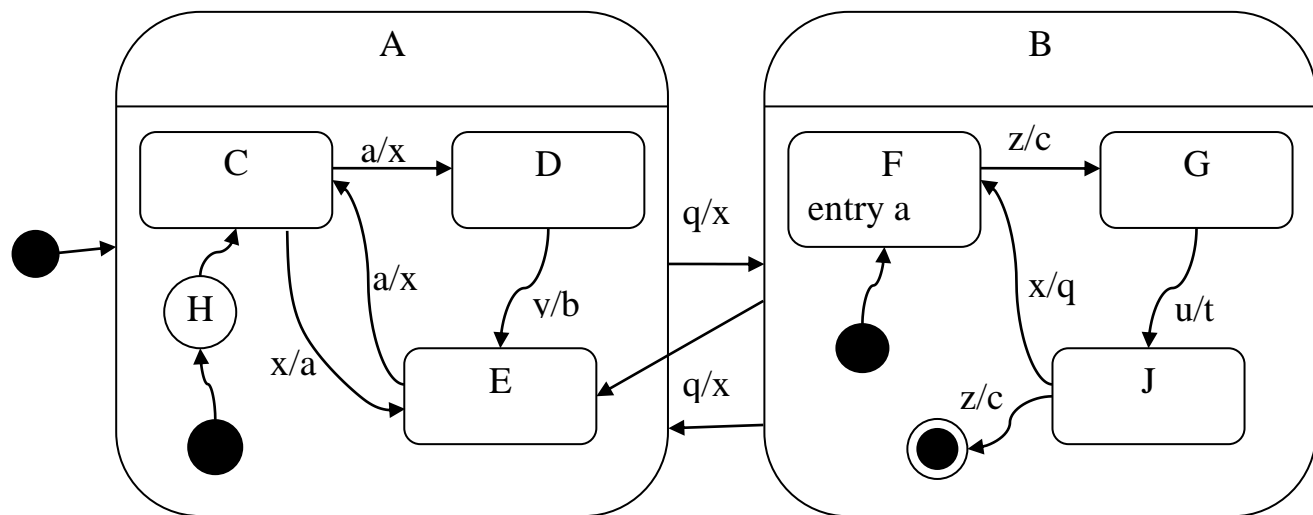


Igaz	Hamis	Állítás
<input type="checkbox"/>	<input type="checkbox"/>	H állapotból bármely esemény bekövetkeztekor F állapotba jutunk.
<input type="checkbox"/>	<input type="checkbox"/>	B állapot elhagyásakor a 'c' tevékenység pontosan egyszer hajtódik végre.
<input type="checkbox"/>	<input type="checkbox"/>	G-ből két lépésben eljuthatunk L-be.
<input type="checkbox"/>	<input type="checkbox"/>	A J állapottal egyidőben lehetünk K-ban is.

A kezdés után a következő esemény-szekvencia hatására: **a, b, x, y, a, b**

Igaz	Hamis	Állítás
<input type="checkbox"/>	<input type="checkbox"/>	pontosan kétszer fut le az 'x' tevékenység.
<input type="checkbox"/>	<input type="checkbox"/>	a végén G állapotba kerülünk.
<input type="checkbox"/>	<input type="checkbox"/>	pontosan kétszer fut le a 'c' tevékenység.
<input type="checkbox"/>	<input type="checkbox"/>	pontosan kétszer érintettük az L állapotot.

A következő UML2 állapotdiagram alapján minősítse az állításokat!

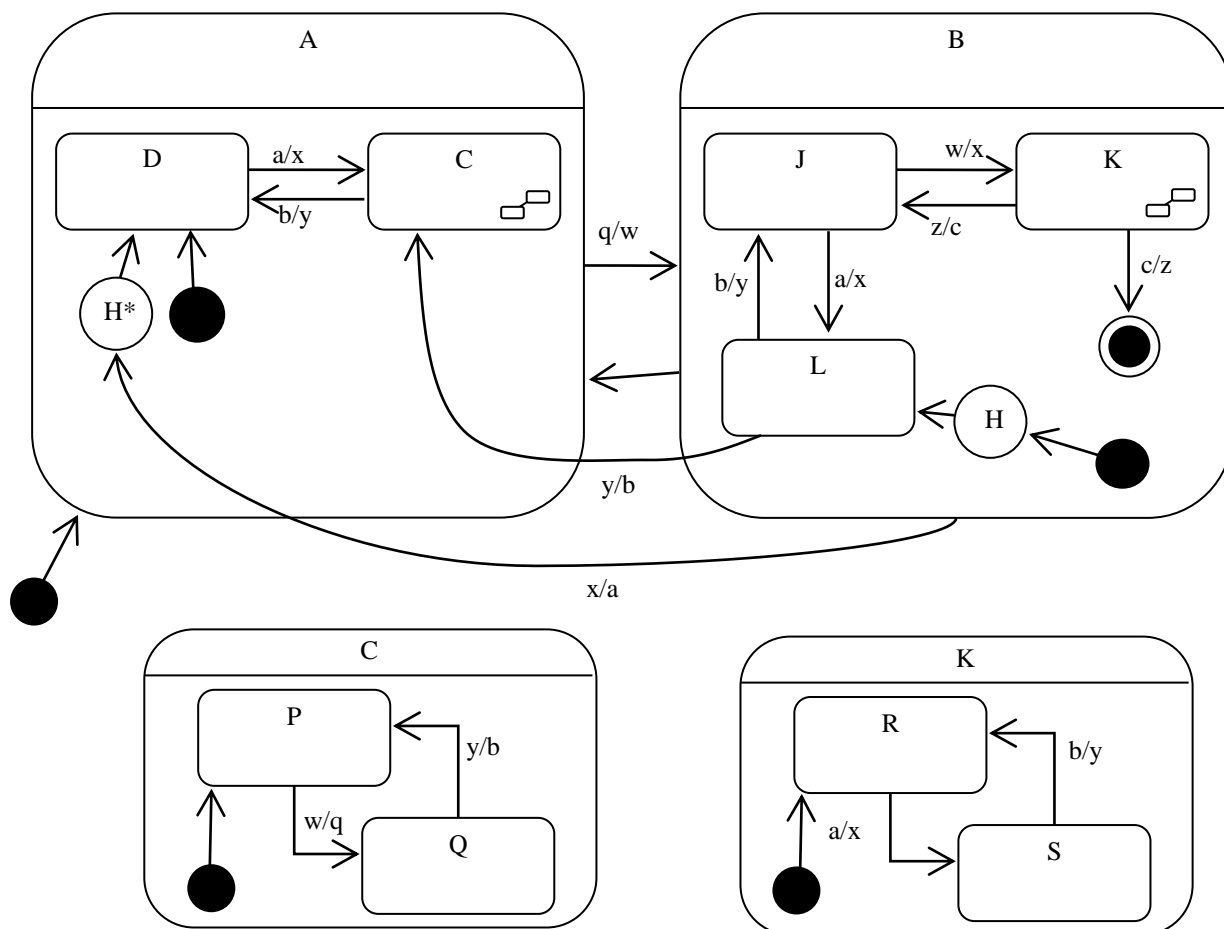


Igaz	Hamis	Állítás
<input type="checkbox"/>	<input type="checkbox"/>	D állapotból 2 lépésben visszaérhet D-be
<input type="checkbox"/>	<input type="checkbox"/>	J állapotból egyetlen esemény hatására D állapot következhet
<input type="checkbox"/>	<input type="checkbox"/>	F állapotból „q” esemény hatására H állapotba kerül
<input type="checkbox"/>	<input type="checkbox"/>	B-ből A-ba való váltáskor végrehajthat az „a” tevékenység

A kezdés után az **x, q, z, q** esemény-szekvencia hatására

Igaz	Hamis	Állítás
<input type="checkbox"/>	<input type="checkbox"/>	Kétszer lefut az „x” tevékenység
<input type="checkbox"/>	<input type="checkbox"/>	E állapotba kerülünk
<input type="checkbox"/>	<input type="checkbox"/>	Kétszer lefut az „a” tevékenység

A következő UML állapotdiagram alapján minősítse az állításokat! Csak a rubrikába tett jelzést vesszük figyelembe!

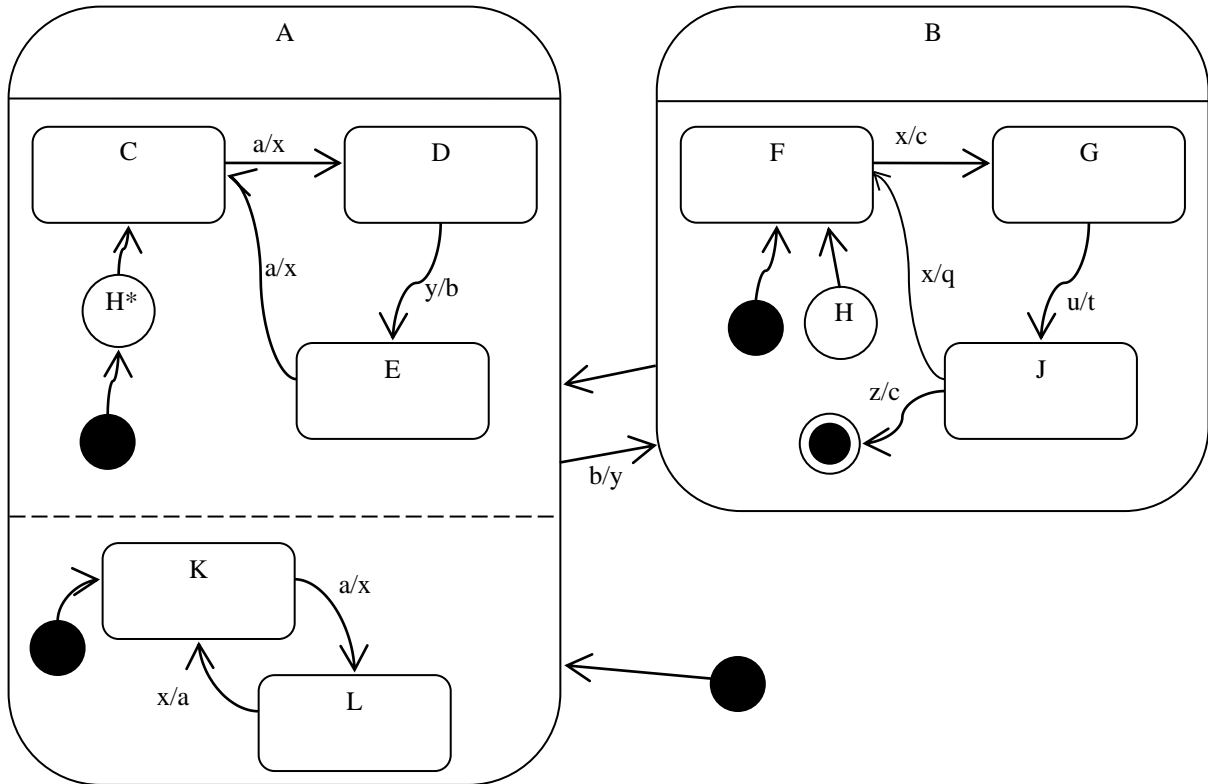


Igaz	Hamis	Állítás
<input type="checkbox"/>	<input type="checkbox"/>	L állapot után közvetlenül következhet Q állapot
<input type="checkbox"/>	<input type="checkbox"/>	C állapotból elérhető egy lépésben S
<input type="checkbox"/>	<input type="checkbox"/>	K állapotból csak „c” és „x” esemény hatására léphetünk át A állapotba
<input type="checkbox"/>	<input type="checkbox"/>	Q állapotból „y” esemény hatására átlépünk D-be
<input type="checkbox"/>	<input type="checkbox"/>	L állapot után csak C és J következhet egy lépésben

A kezdés után az **a, w, q, b, x** esemény-szekvencia hatására

Igaz	Hamis	Állítás
<input type="checkbox"/>	<input type="checkbox"/>	pontosan kétszer fut le a „q” tevékenység
<input type="checkbox"/>	<input type="checkbox"/>	Q állapotba kerülünk
<input type="checkbox"/>	<input type="checkbox"/>	érintettük az K állapotot

A következő UML állapotdiagram alapján minősítse az állításokat! Csak a rubrikába tett jelzést vesszük figyelembe!

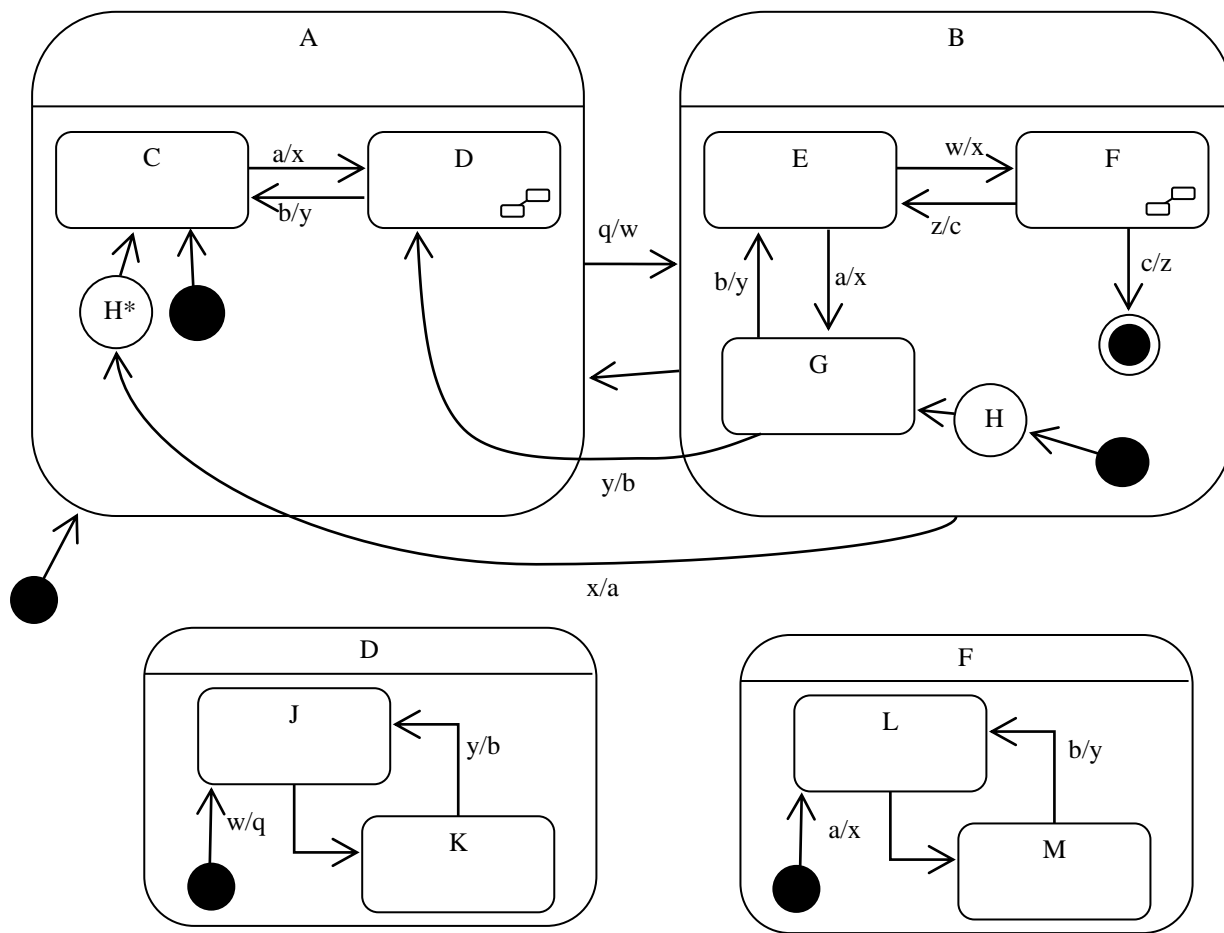


Igaz	Hamis	Állítás
<input type="checkbox"/>	<input type="checkbox"/>	D állapotból B érintésével csak pontosan 5 lépésben juthatunk E-be.
<input type="checkbox"/>	<input type="checkbox"/>	Az E állapotból egy lépésben elérhető állapotok: C, F, G, J.
<input type="checkbox"/>	<input type="checkbox"/>	J-ből egy lépésben nem juthatunk el L-be.
<input type="checkbox"/>	<input type="checkbox"/>	K-ból L-be csak akkor léphetünk, ha közvetlenül előtte C-ből D-be léptünk.
<input type="checkbox"/>	<input type="checkbox"/>	Az F állapottal egyidőben lehetünk L-ben is.

A kezdés után a következő esemény-szekvencia hatására: **a, b, x, u, z, b**

Igaz	Hamis	Állítás
<input type="checkbox"/>	<input type="checkbox"/>	pontosán kétszer fut le az 'x' tevékenység.
<input type="checkbox"/>	<input type="checkbox"/>	pontosán kétszer érintettük az L állapotot.
<input type="checkbox"/>	<input type="checkbox"/>	a végén J állapotba kerülünk.

A következő UML állapotdiagram alapján minősítse az állításokat! Csak a rubrikába tett jelzést vesszük figyelembe!

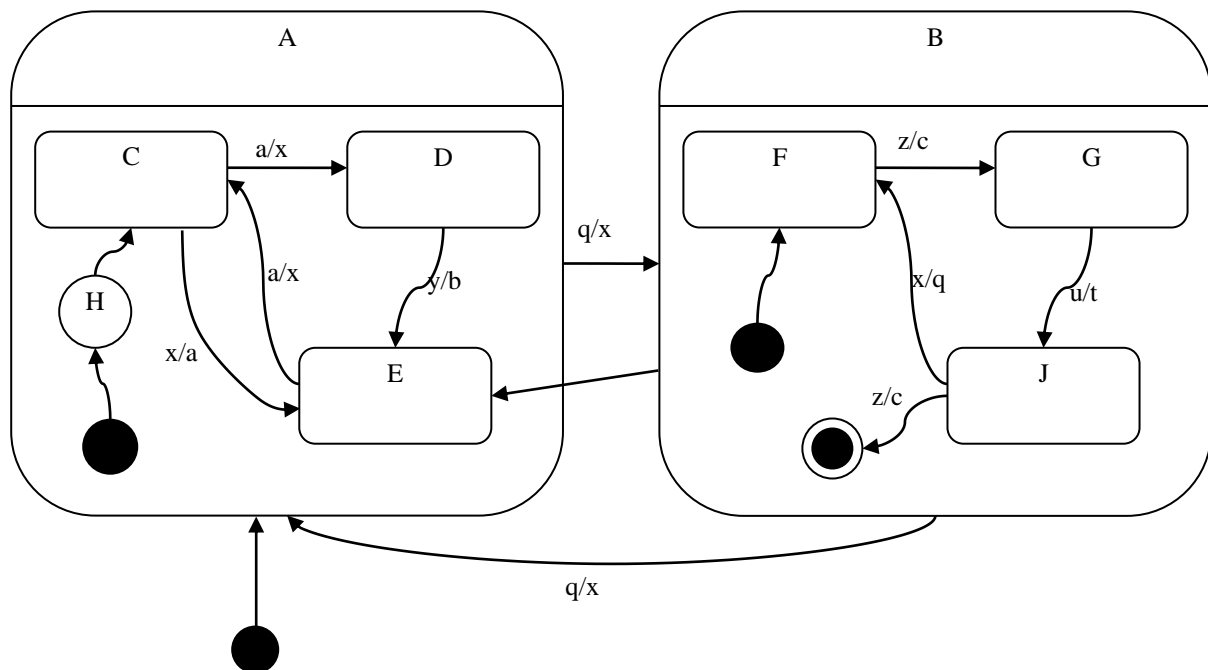


Igaz	Hamis	Állítás
<input type="checkbox"/>	<input type="checkbox"/>	G állapot után csak D és E következhet egy lépésben
<input type="checkbox"/>	<input type="checkbox"/>	D állapotból elérhető egy lépésben M
<input checked="" type="checkbox"/>	<input type="checkbox"/>	G állapot után közvetlenül következhet K állapot
<input type="checkbox"/>	<input type="checkbox"/>	K állapotból „y” esemény hatására átlépünk C-be
<input checked="" type="checkbox"/>	<input type="checkbox"/>	F állapotból csak „c” és „x” esemény hatására léphetünk át A állapotba

A kezdés után az **a, w, q, b, x** esemény-szekvencia hatására

Igaz	Hamis	Állítás
<input checked="" type="checkbox"/>	<input type="checkbox"/>	K állapotba kerülünk
<input type="checkbox"/>	<input type="checkbox"/>	érintettük az F állapotot
<input type="checkbox"/>	<input type="checkbox"/>	pontosan kétszer fut le a „q” tevékenység

A következő UML2 állapotdiagram alapján minősítse az állításokat!

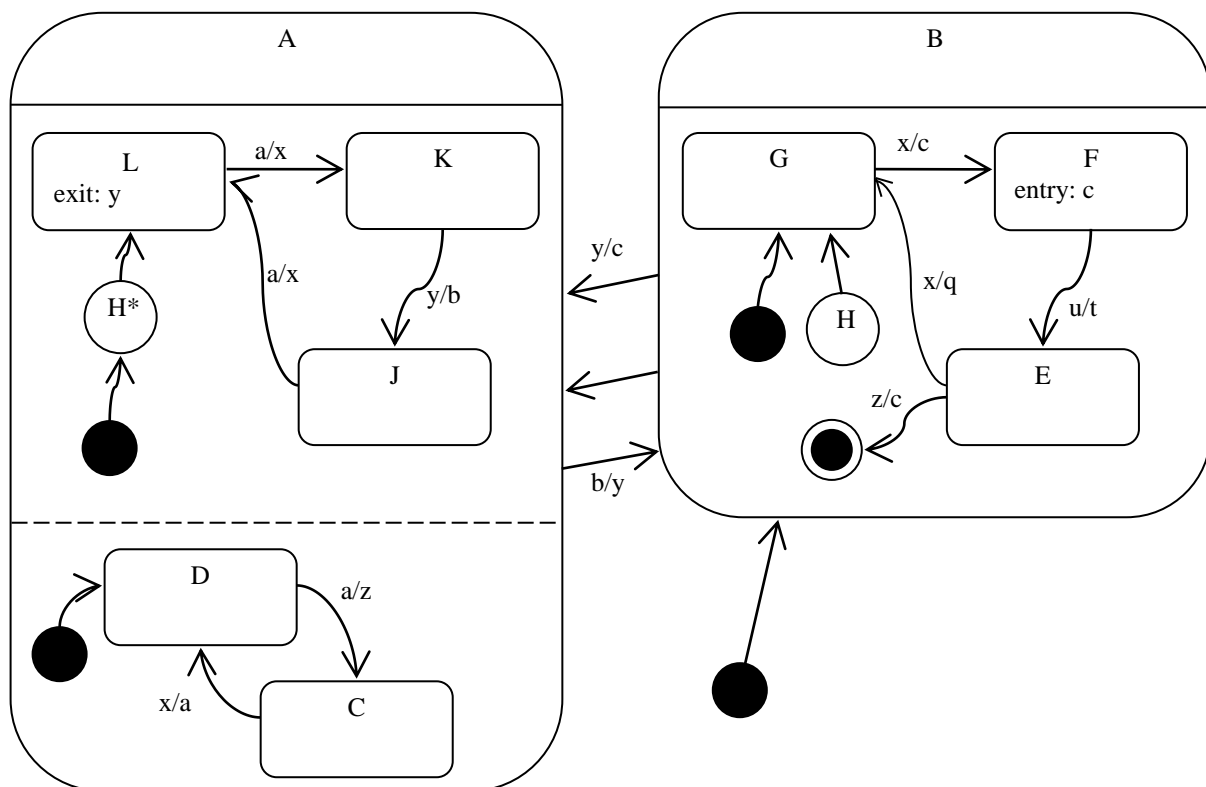


Igaz	Hamis	Állítás
<input type="checkbox"/>	<input type="checkbox"/>	D állapotból 2 lépésben visszaérhet D-be
<input type="checkbox"/>	<input type="checkbox"/>	F állapotból „q” esemény hatására H állapotba kerül
<input type="checkbox"/>	<input type="checkbox"/>	B-ből A-ba való váltáskor végrehajtható a „c” tevékenység
<input type="checkbox"/>	<input type="checkbox"/>	E állapotból egyetlen esemény hatására csak a C állapot következhet
<input type="checkbox"/>	<input type="checkbox"/>	J állapotból egyetlen esemény hatására E állapot következhet

A kezdés után az **x, q, z, q** esemény-szekvencia hatására

Igaz	Hamis	Állítás
<input type="checkbox"/>	<input type="checkbox"/>	C állapotba kerülünk
<input type="checkbox"/>	<input type="checkbox"/>	Kétszer lefut az „x” tevékenység
<input type="checkbox"/>	<input type="checkbox"/>	Érintjük a J állapotot

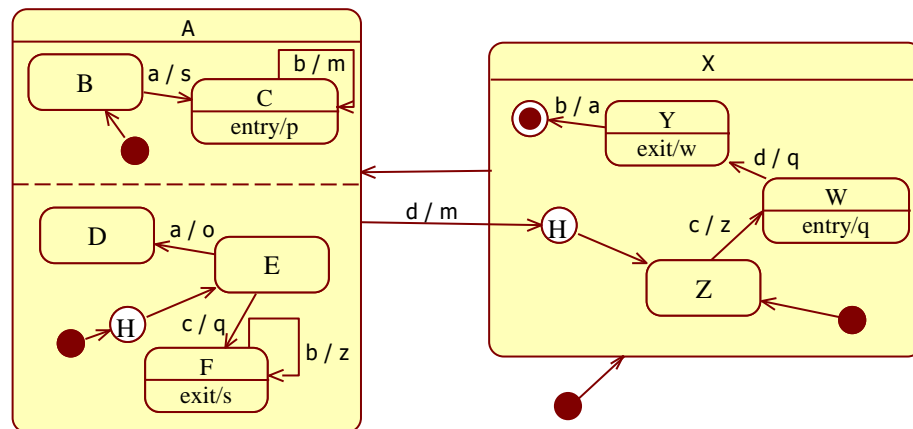
A következő UML állapotdiagram alapján minősítse az állításokat!



A kezdés után a következő esemény-szekvencia hatására: **x, y, a, b, x, u, z**

Igaz	Hamis	Állítás
<input type="checkbox"/>	<input type="checkbox"/>	K állapottal egyidőben voltunk D-ben is.
<input type="checkbox"/>	<input type="checkbox"/>	J állapottal egyidőben voltunk C-ben is.
<input type="checkbox"/>	<input type="checkbox"/>	valamennyi állapotot érintettük.
<input type="checkbox"/>	<input type="checkbox"/>	c tevékenység ötnél kevesebbyszer fut le.
<input type="checkbox"/>	<input type="checkbox"/>	van olyan átmenet, amelynek során lefut az x, az y és a z tevékenység is.
<input type="checkbox"/>	<input type="checkbox"/>	a végén E állapotba kerülünk.
<input type="checkbox"/>	<input type="checkbox"/>	a végén L/D állapotba kerülünk.
<input type="checkbox"/>	<input type="checkbox"/>	pontosan kétszer érintettük a K állapotot.

A következő UML2 állapotdiagram alapján minősítse az állításokat!

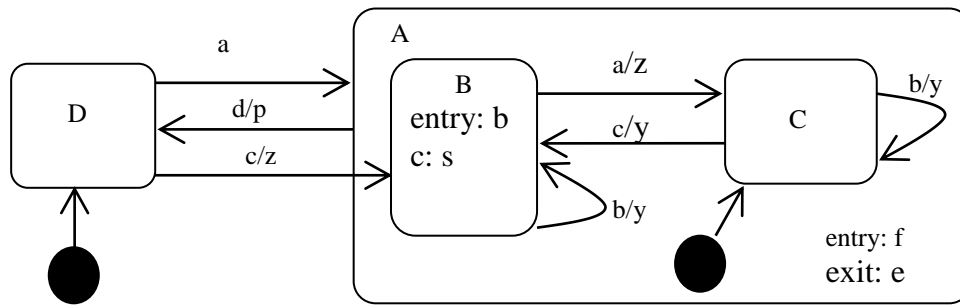


A kezdés után a következő esemény-szekvencia hatására: **c, d, b, c, a, b, d**

Igaz	Hamis	Állítás
<input type="checkbox"/>	<input checked="" type="checkbox"/>	pontosan kétszer érintettük a Z állapotot.
<input checked="" type="checkbox"/>	<input type="checkbox"/>	van olyan tevékenység, ami háromszor hajtódik végre.
<input checked="" type="checkbox"/>	<input type="checkbox"/>	van olyan tevékenység, ami nem hajtódik végre.
<input type="checkbox"/>	<input checked="" type="checkbox"/>	a végén Z állapotba kerülünk.
<input type="checkbox"/>	<input checked="" type="checkbox"/>	A összes alállapotát érintjük.
<input checked="" type="checkbox"/>	<input type="checkbox"/>	pontosan kétszer érintettük az Y állapotot.
<input checked="" type="checkbox"/>	<input type="checkbox"/>	van olyan átmenet, amikor 4 különböző tevékenység hajtódik végre.
<input checked="" type="checkbox"/>	<input type="checkbox"/>	X-nek van olyan alállapota, amit pontosan kétszer érintettünk.

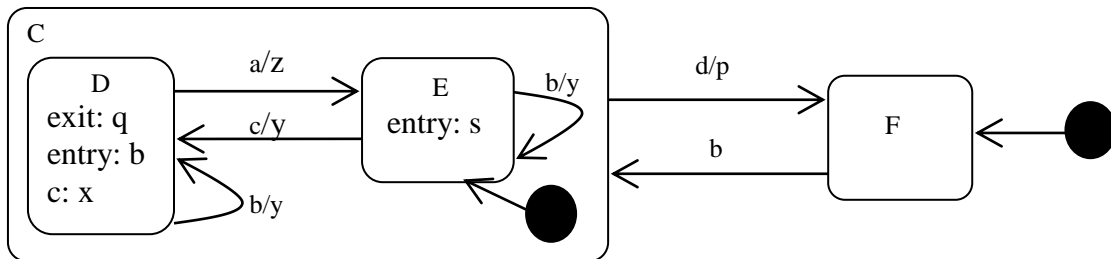


Rajzolja fel az alábbi UML2 state-chart-nak megfelelő állapototáblát !



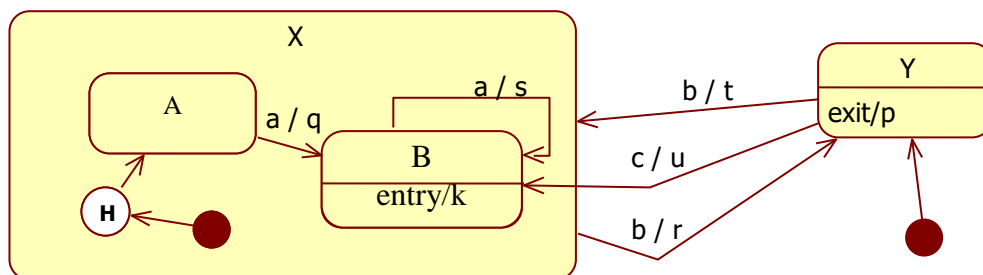
	a	b	c	d
D	C/ f		B/z, f, b	
C		C/y	B/y, b	D/e, p
B	C/ z	B/ y, b	B/s	D/e, p

Rajzolja fel az alábbi UML2 state-chart-nak megfelelő állapototáblát !



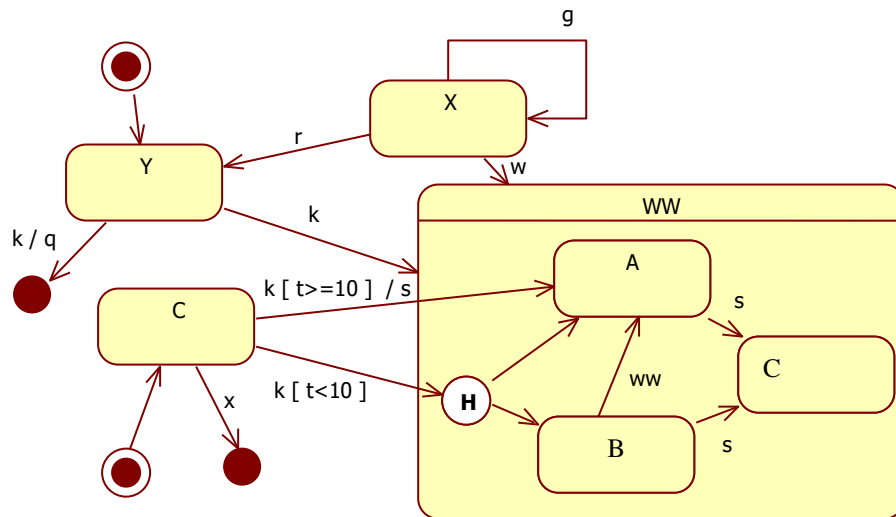
	a	b	c	d
D	E/q, z, s	D/q, y, b	D/x	F/q, p
E		E/y, s	D/y, b	F/p
F		E/s		

Rajzolja fel az alábbi UML2 state-chart-nak megfelelő állapototáblát! A nem specifikált blokkokat jelölje “---” jellel! A kezdő állapot legyen a táblázat első sora!



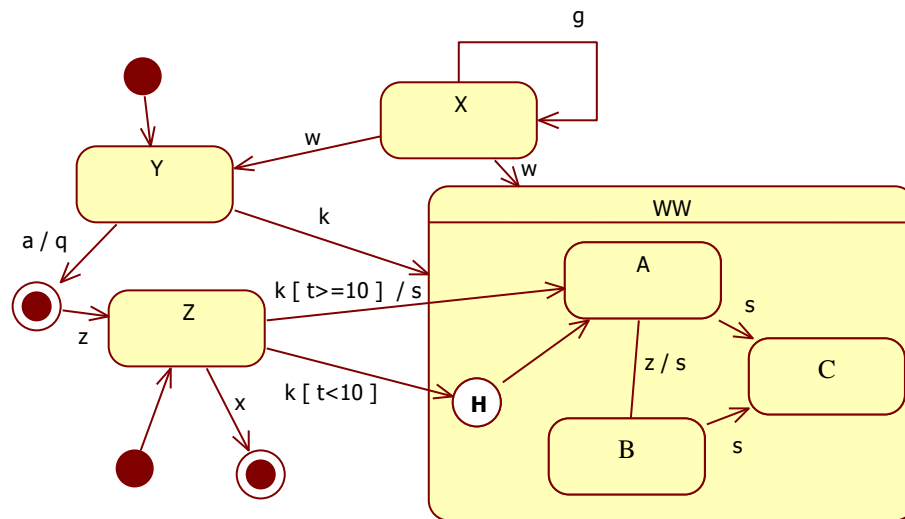
	a	b	c
YA	---	A/p,t	B/p,u,k
YB	---	B/p,t,k	B/p,u,k
A	B/q,k	YA/r	---
B	B/s,k	YB/r	---

Milyen szintaktikai és szemantikai hibák találhatók az alábbi UML2 állapot-diagramon (state-chart)?



- Végállapotból kilépünk
- Kezdőállapotba belépünk  
(kezdő és végállapotok fel vannak cserélve)
- Több kezdőállapot is van
- WW-ben nincs kezdőállapot
- WW-ből nem jutunk végállapothoz
- Y-ból két k kilépés is van
- WW-ben a historyből kétfele is megyünk
- X forrás (nincs belépő átmenet)

Milyen szintaktikai és szemantikai hibák találhatók az alábbi UML2 állapot-diagramon (state-chart)?



Két kezdőállapot

Ugyanarra az eseményre két állapotba is mehet

Végállapotból kilépünk

Nincs kezdőállapot

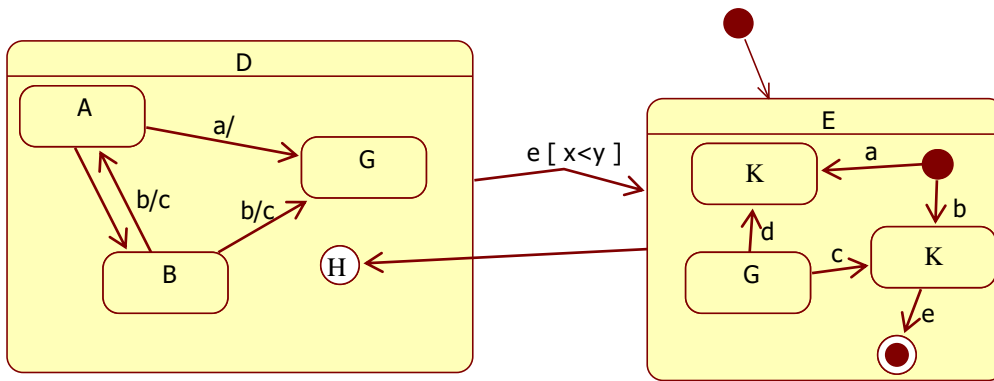
Ugyanolyan nevű állapotok

Nincs nyíl

Csak kimenő állapot

Csak bemenő állapot

Az alábbi UML2 állapotábrán (state chart) találhatóak szintaktikai (jelölésbeli) és szemantikai (értelmi) hibák. Sorolja fel őket!

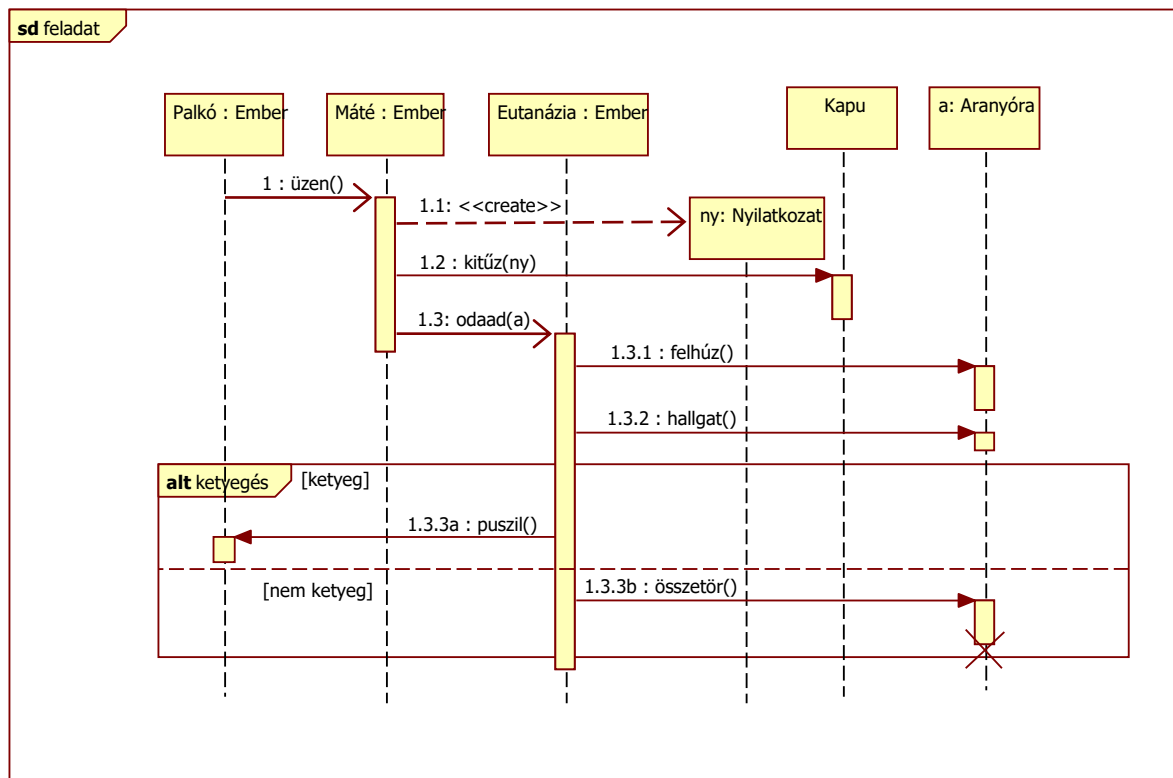


- E belső start állapota 2 helyre megy.....
- E-ben 2 K-van.....
- D history-jának nincs default célállapota.....
- G-ből csak kimenő átmenet van .....
- A-ból B-be átmeneten nincs jelölés .....
- E-ben startból átmenet eventtel .....

## 5 Szekvenciadiagram

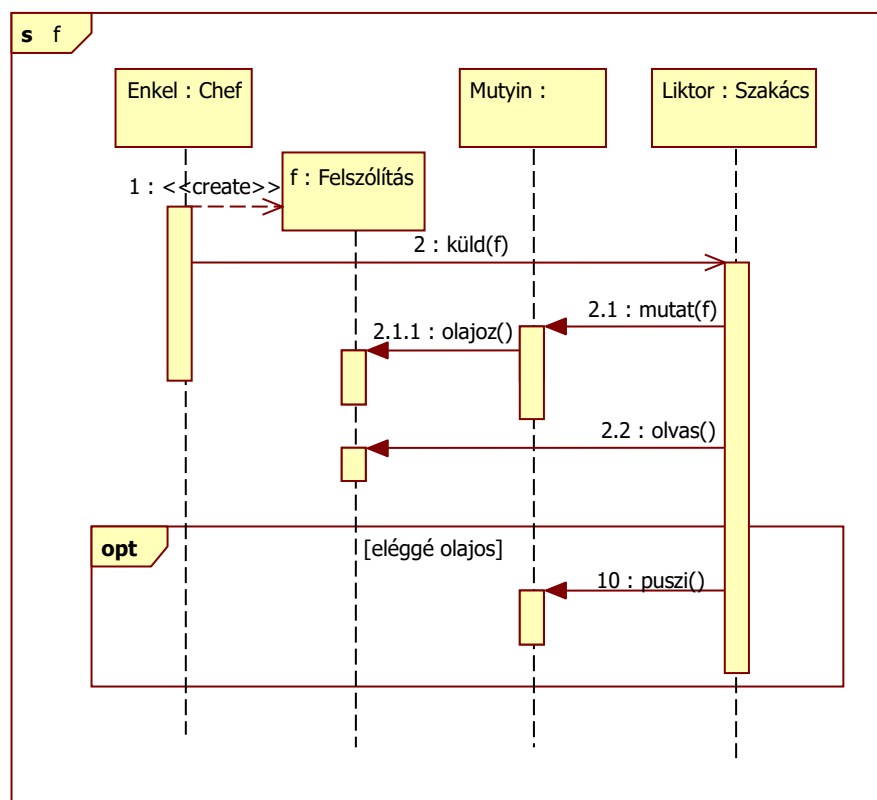
Készítsen UML 2 szekvenciadiagramot az alábbi történet alapján! Az üzeneteket hierarchikusan számozza!

Szállkovics Palkó megüzeni Boldog Máté Elemérnek, hogy adja oda aranyóráját Erkölcsfalvy Lorádné Tokács Eutanáziának. Máté ír egy nyilatkozatot, amit kitéűz a vármegye kapujára, majd az órát átadja Eutanáziának és elsiet. Eutanázia felhúzza az órát, majd meghallgatja, ketyeg-e. Ha az óra ketyeg, akkor Eutanázia Palkónak puszit ad; ha nem ketyeg, akkor az órát összetöri.



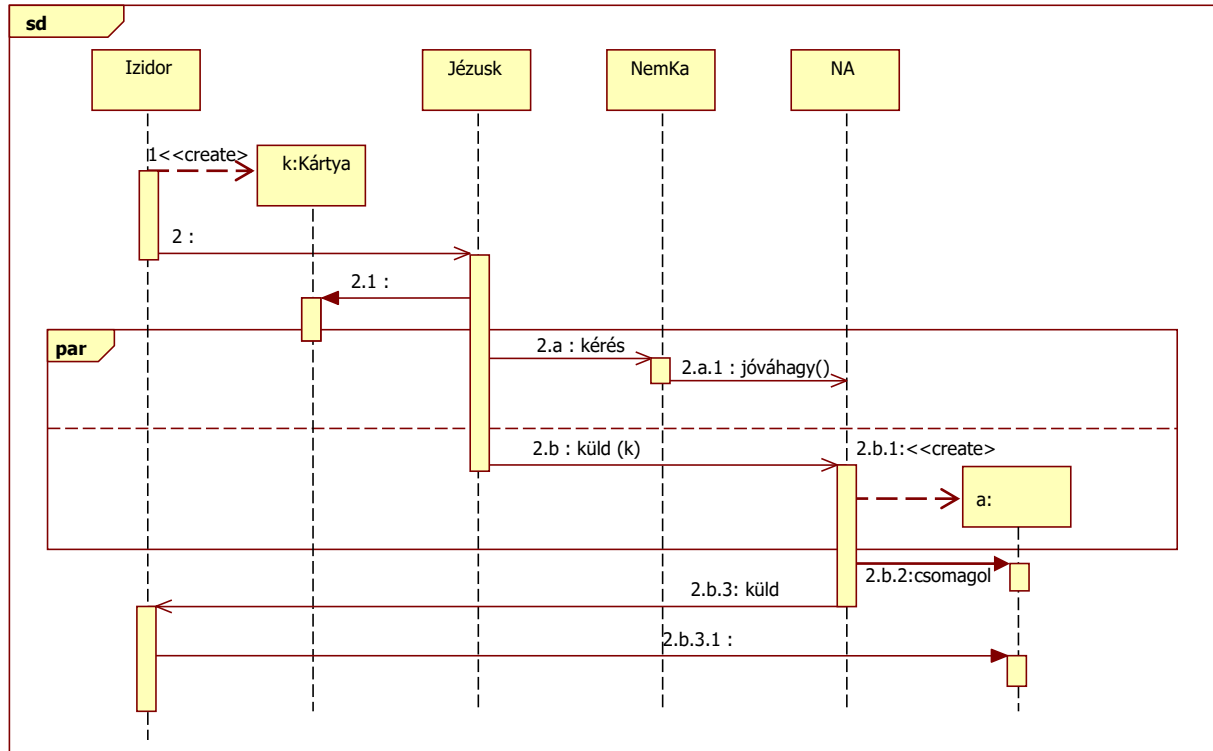
Készítsen UML2 szekvenciadiagramot az alábbi történet alapján!

Zsarátnok városában két étterem üzemel. Az egyiket Margela Enkel, a másikat Gagyinír Mutyin vezeti. Enkelnek nem tetszik, hogy egyik szakácsa, Hombár Liktör kokettál Mutyinnal, ezért ír egy felszólítást Liktornak, majd elküldi neki. Liktör a felszólítást megmutatja Mutyinnak, amit az azon nyomban megolajoz, majd Liktör átolvassa a felszólítást. Ha az eléggé olajos, akkor Mutyin orrára nyom egy puszt.



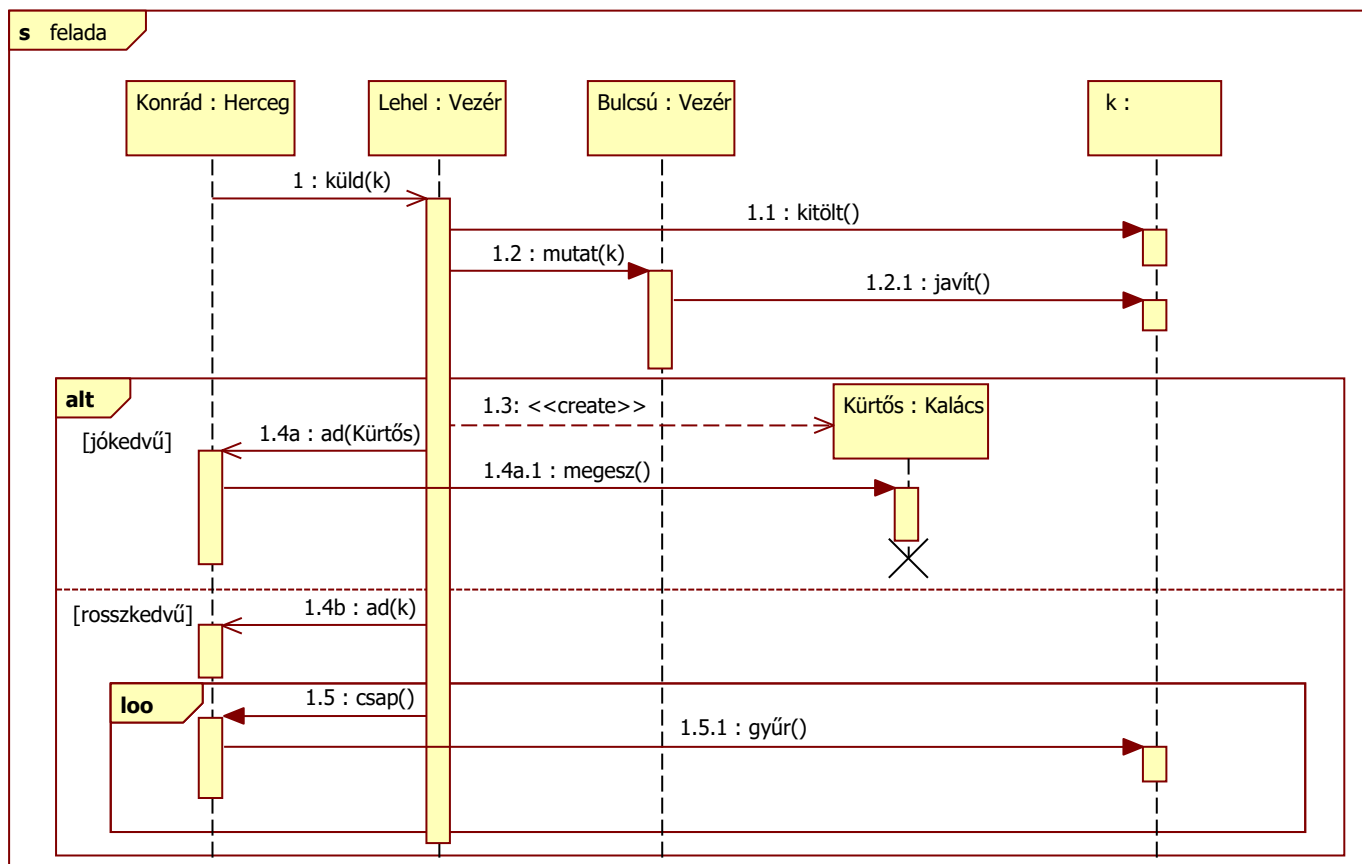
Az alábbi leírás alapján rajzoljon UML szekvencia-diagramot!

Izidor szeretne kapni egy életnagyságú legó stadiont, ezért készít egy fotóval ellátott kívánságkártyát, amit eljuttat a Jézuskának. Jézuska a kártyára ötkarikás szívet rajzol, majd ismeretlen sorrendben megküldi a kívánságot a Nemzeti Karácsony Partnerhez (NemKaP), a kártyát pedig a Nemzeti Ajándékgyártó Hivatalba (NAB). A NAB-ban elővarázsolják a kért ajándékot. Időközben a NemKaP jóváhagyó döntést hoz, és erről értesíti a NAB-ot. Amikor a NAB-hoz megérkezik a jóváhagyás, akkor becsomagolják az ajándékot és elküldik azt Izidornak. Izidor kicsomagolja az ajándékot.



Rajzoljon UML2 szekvenciadiagramot az alábbi történet alapján!

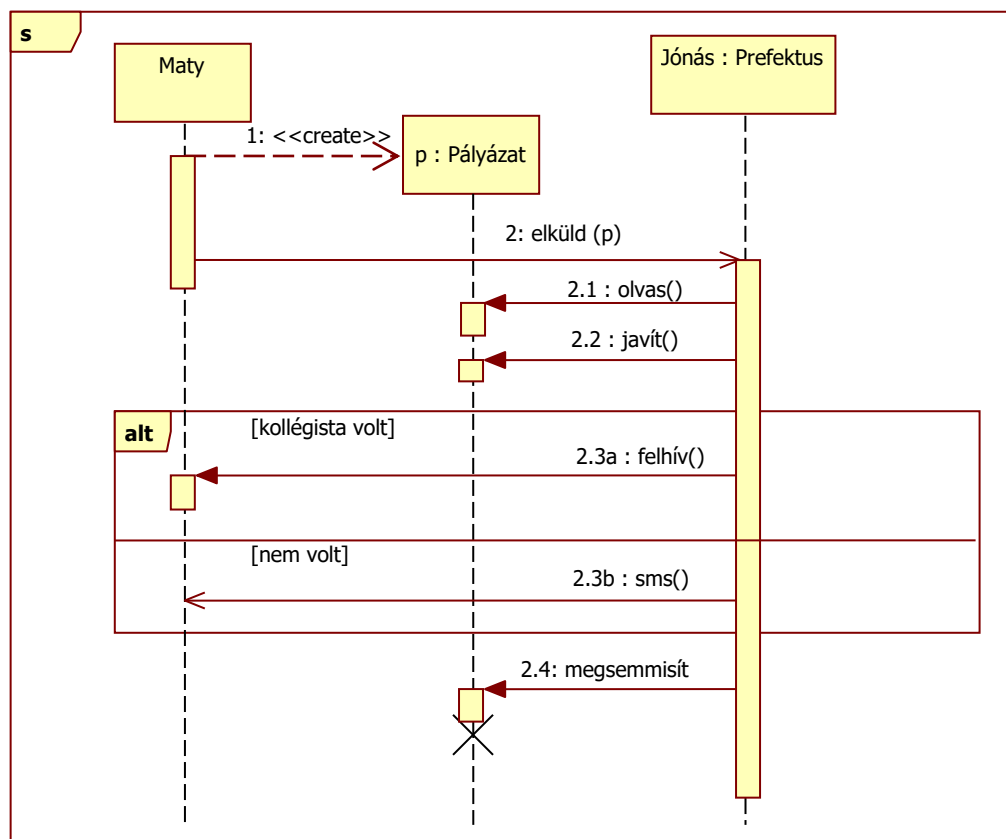
Konrád herceg konzultációs kérdőívet küld Lehel vezérnek. Lehel kitölti, majd átnyújtja Bulcsúnak, hogy ellenőrizze a helyesírást, az eredményt pedig megvárja. Bulcsú két rovást kijavít. Ha Lehel jókedvű lesz, akkor süt egy kürtőskalácsot, és odaadja Konrádnak, aki a kalácsot megeszi. Ha Lehel rosszkedvű, akkor visszaadja Konrádnak a kérdőívet, majd Lehel többször is fejbecsapja a herceget, aki minden csapásnál gyűr egyet az íven.





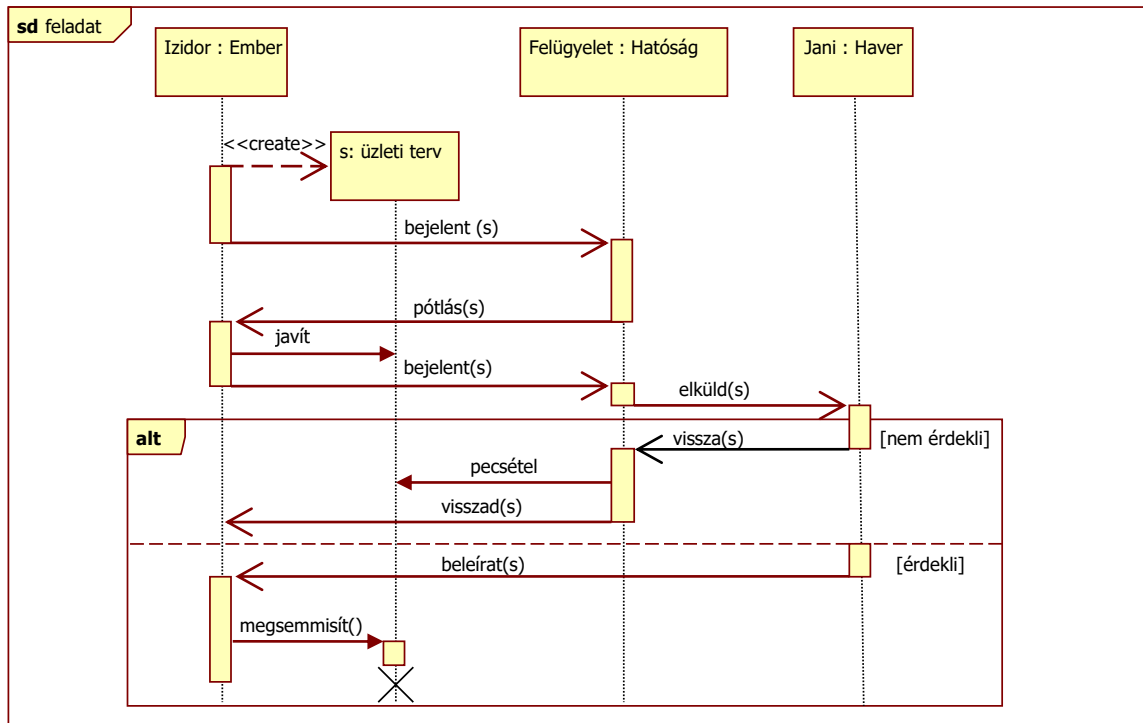
Készítsen UML2 szekvenciadiagramot az alábbi történet alapján!

Nernia egy eldugott szegletében, Lopószenttőkön Mutyi Matyi szeretne szotyolázót nyitni. Ezért készít egy pályázati anyagot, amit elküld Gázár Jónásnak, a prefektusnak. A prefektus elolvassa a pályázatot, majd kijavítja a helyesírási hibákat. Ha Matyi kollégista volt, akkor telefonon értesíti a sikerről, ha nem volt kollégista, akkor SMS-ben tájékoztatja. A prefektus az eljárás végén a pályázati anyagot (az adminisztráció egyszerűsítése céljából) megsemmisíti.



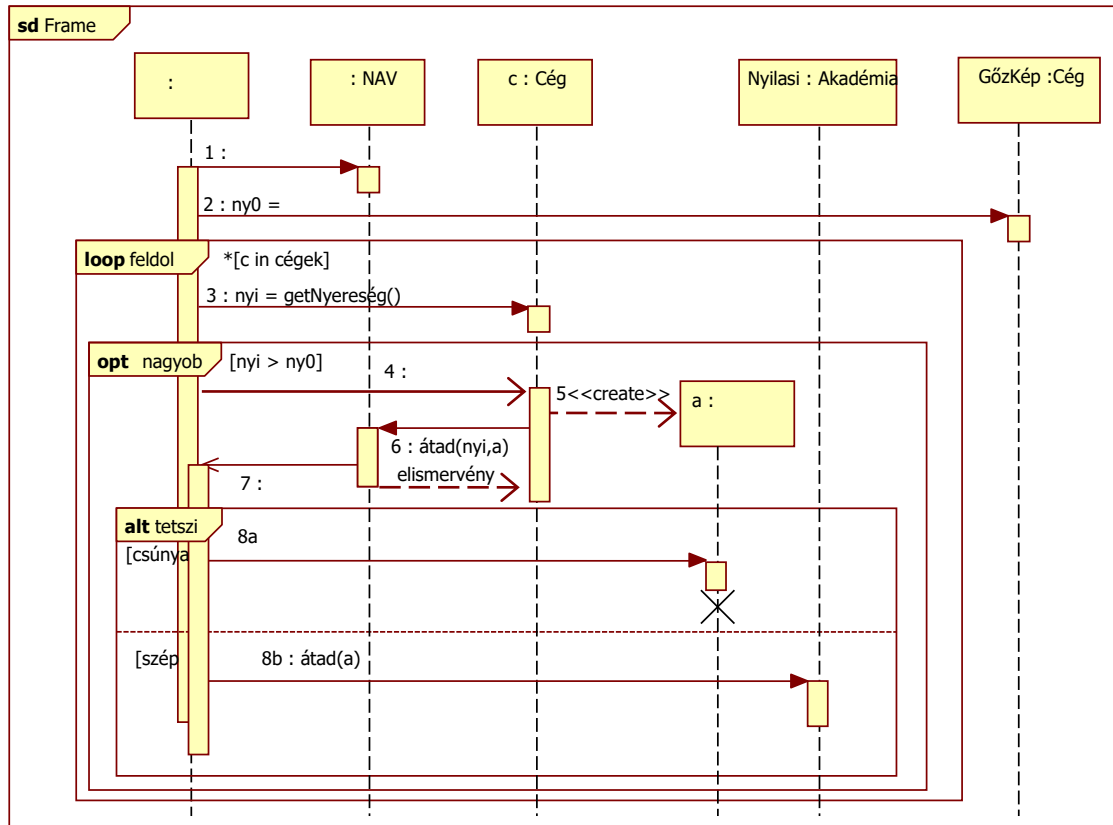
Az alábbi történet alapján készítsen UML2 szekvencia diagramot!

Izidor készít egy üzleti tervet. Ezt beküldi a felügyeleti hatóságnak jóváhagyásra. A hatóság visszaküldi, hiánypótlást kérve. Izidor javítja a tervet és újra beküldi. A felügyelet elküldi a tervet a haver Janinak, hátha érdekli őt az üzlet. Ha Janit nem érdekli, akkor visszaküldi a hatóságnak és Izidor megkapja a lepecsételt (engedélyezett) tervet. Ha Janit érdekli a terv, akkor átküldi Izidornak, hogy írja bele őt (Janit) is. Erre Izidor nem hajlandó, inkább megsemmisíti a tervet, így a felügyeletnek nincs mit engedélyezni.



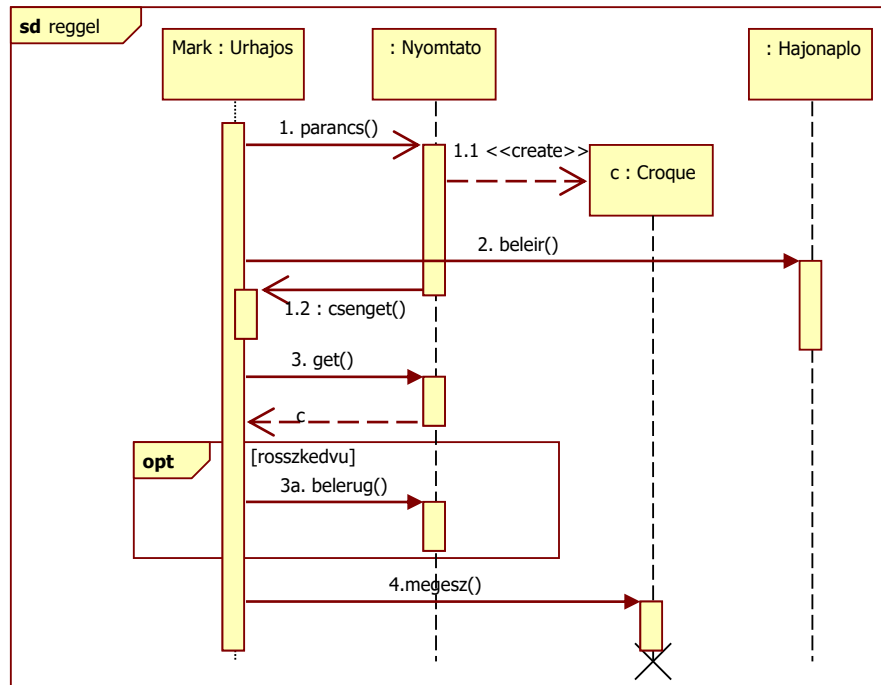
Készítsen UML2 szekvenciadiagramot az alábbi történet alapján!

A Nemzeti Egyensúly Hivatal (NEH) lekéri a NAV-tól a bekötött pénztárgép cégek listáját. A listán szereplő cégektől elkéri az éves eredményt, és amelyik cégnél a GőzKép ZRt-ét meghaladó nyereség keletkezett, levélben felszólítást küld. A cégnek a felszólítás hatására azonnal kis ajándékot kell készítenie, és az ajándékkal együtt át kell adnia a nyereségét a NAV-nak. A NAV az ajándékot azonnal továbbküldi a NEH-nek, majd a cég felé elismervénnyel nyugtázza az adomány átvételét. Ha az ajándék csúnya, a NEH megsemmisíti, ha szép, akkor a Nyilasi Akadémiának adja át.



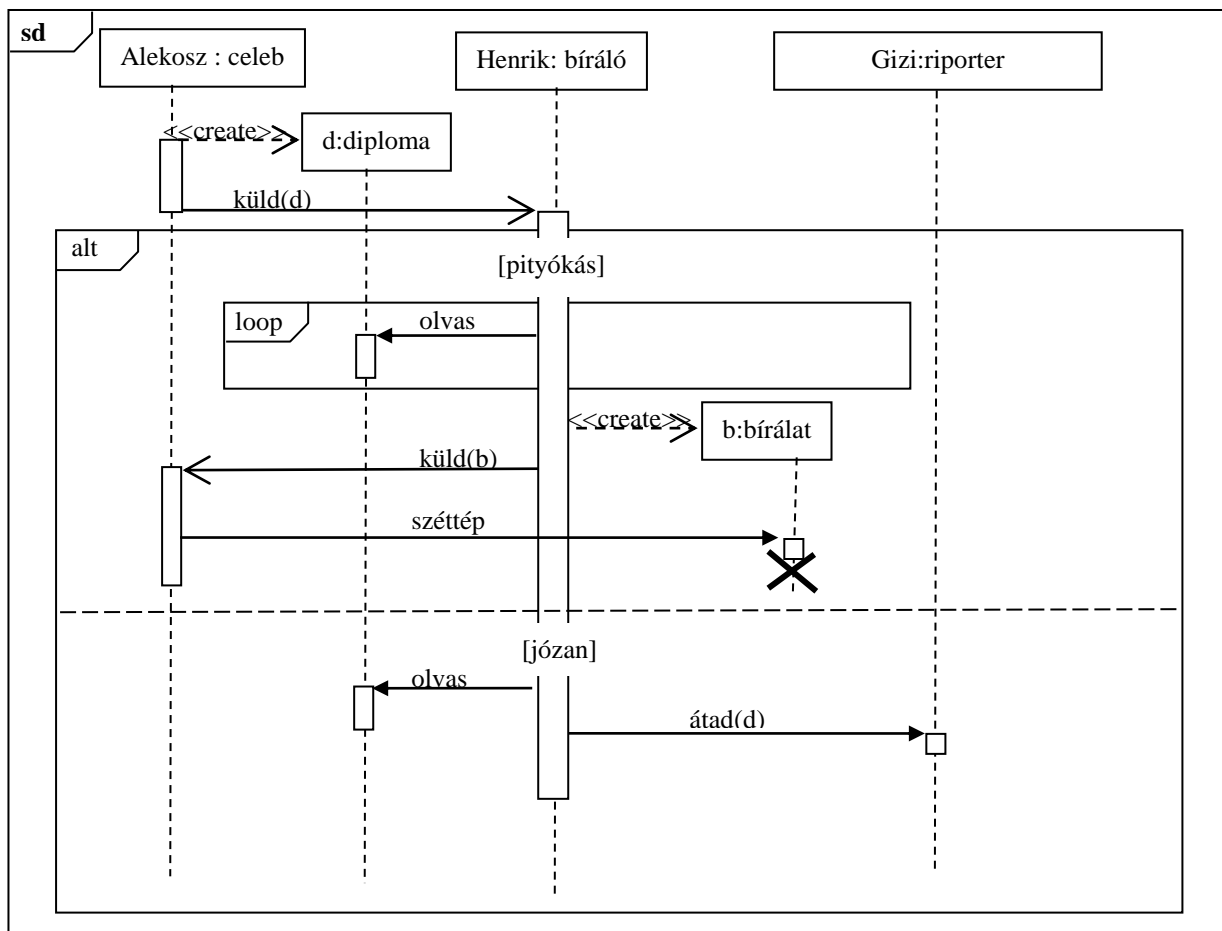
Rajzoljon UML2 szekvenciadiagramot az alábbi leírás alapján!

Mark Watney, a marson ragadt űrhajós reggeli rutinja a következő. Az ételnyomtatónak parancsot ad, hogy készítsen egy Croque Monsieur-t. Amíg az étel készül, Mark beleír a hajónaplóba. Írás közben a nyomtató csenget (amikor befejezte az ételgyártást). Mark, mikor végzett az írással, kiveszi az ételt és az utolsó morzsáig megeszi. Ha rossz passzban van, akkor az étel kivétele után (de mielőtt megenné) belerug a nyomtatóba.



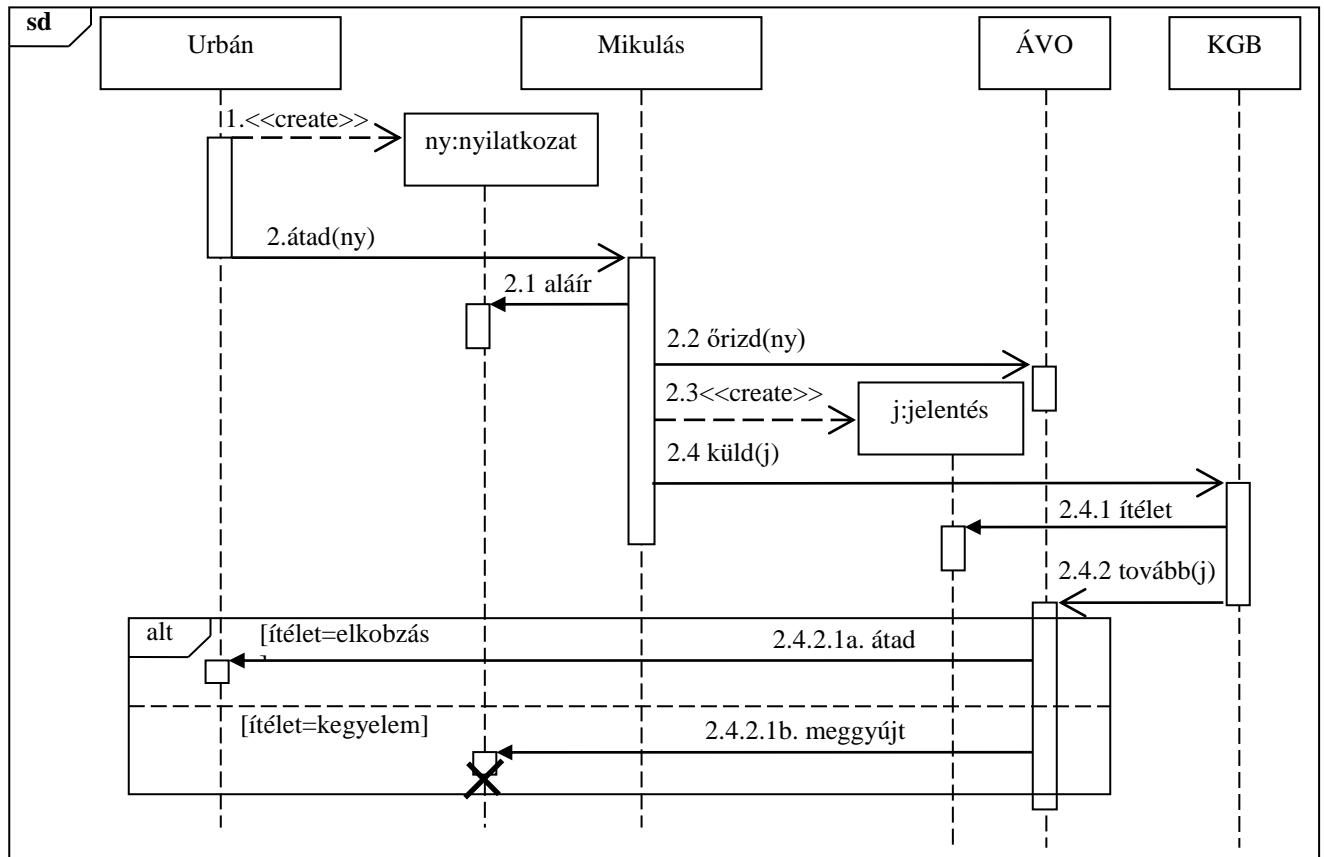
Készítsen UML2 szekvencia-diagramot az alábbi leírás alapján!

Alekosz, a celeb diplomát ír, majd elküldi Hawass Henriknek bírálatra. Ha Henrik pityókás, akkor többször elolvassa a művet, majd ír egy kemény bírálatot, és utóbbit visszaküldi Alekosznak, aki a bírálatot széttépi. Ha Henrik józan, akkor csak egyszer olvassa el a dolgozatot, majd bizalmasan átadja Gizinek, a Flikk oknyomozó riporterének.



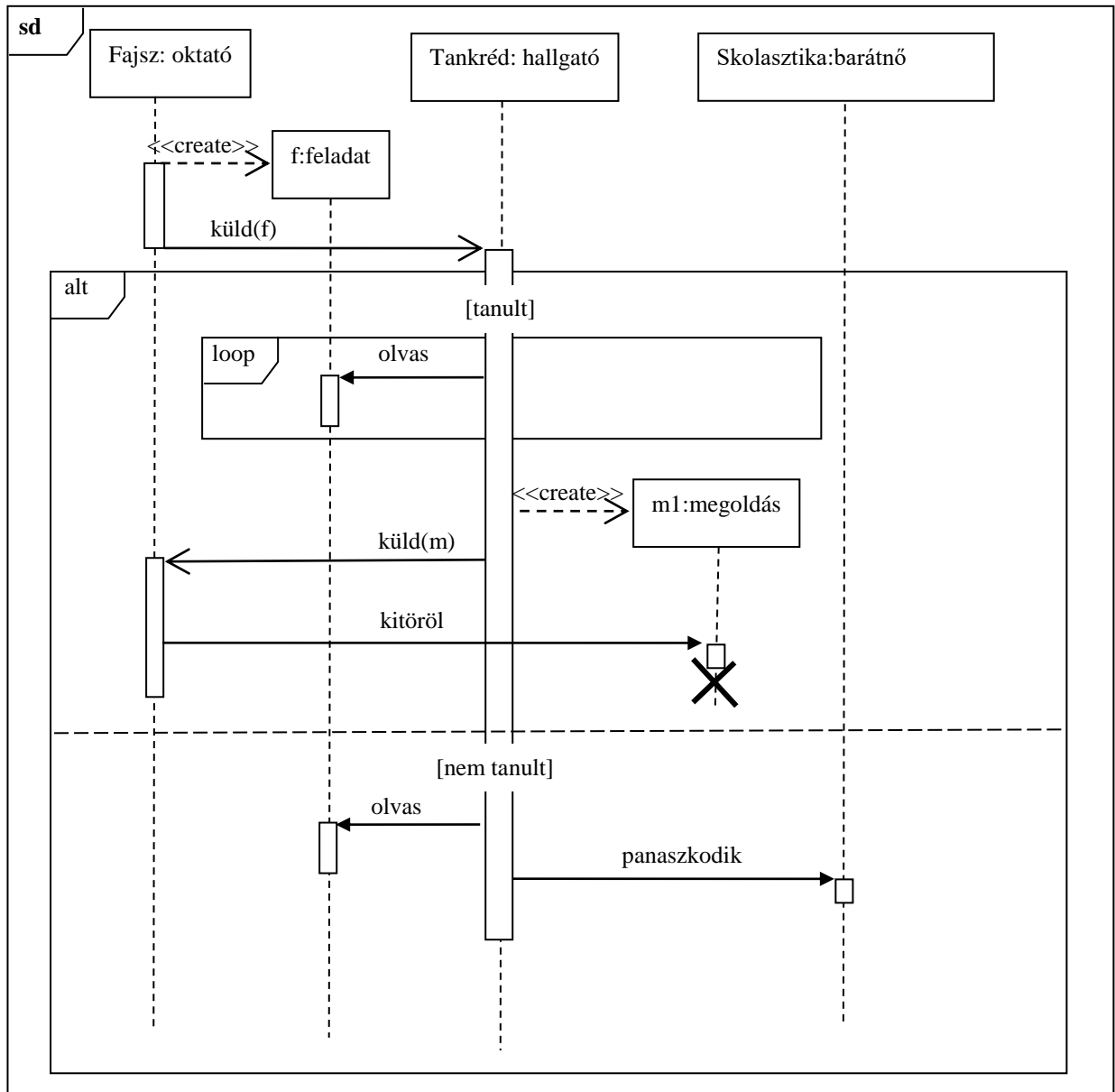
Készítsen UML 2 szekvencia diagramot (sequence diagram) az alábbi leírás alapján! Használjon hierarchikus számozást is!

Virgonc Urbán úgy döntött, hogy maradni kíván a MalacNyúzó Pribékek (MaNyüP) társaságában. Emiatt, az előírásoknak megfelelően, a Mikulásnak átad egy általa készített nyilatkozatot. A Mikulás a nyomtatványt aláírja, majd elküldi az Ákombákom Vizslató Orrszarvúnak (ÁVO), megőrzésre. A Mikulás ezután egy jelentést készít és küld a Központi Gépészeti Bizottságnak (KGB), amely a jelentéshez hozzáírja az ítéletet, és továbbküldi az Orrszarvúnak. Ha az ítélet vagyonelkobzás, akkor az Orrszarvú erről személyesen értesíti Urbánt. Ha méltányos kegyelem, akkor meggyűjtja a nyilatkozatot, és megvárja amíg megsemmisül.



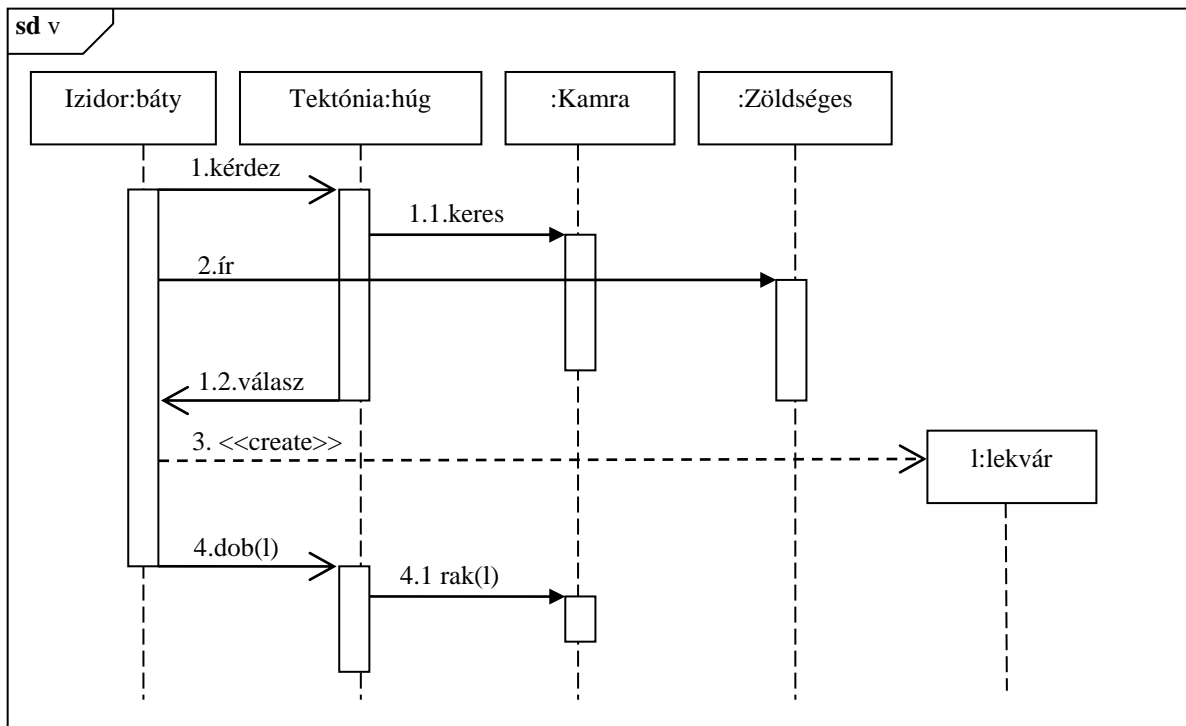
Készítsen UML2 szekvencia-diagramot az alábbi leírás alapján!

A brigittális technika oktatója, Nagy Fajsz elkészíti Tankréd pótházifeladat-kiírását, és el is küldi a hallgatónak e-mailben. Ha Tankréd tanult, akkor többször is elolvassa a feladatot, majd megoldást készít, és visszaküldi Fajsznak. Fajsz a megoldást „véletlenül” letörli. Ha azonban Tankréd nem tanul, akkor csak egyszer olvassa el a feladatot, majd barátjának, Skolasztikának elpanaszolja, hogy mennyire kiszúrtak vele.



Készítsen UML2 szekvencia-diagramot az alábbi történet alapján! Ne feledkezzen el a hierarchikus számozásról sem!

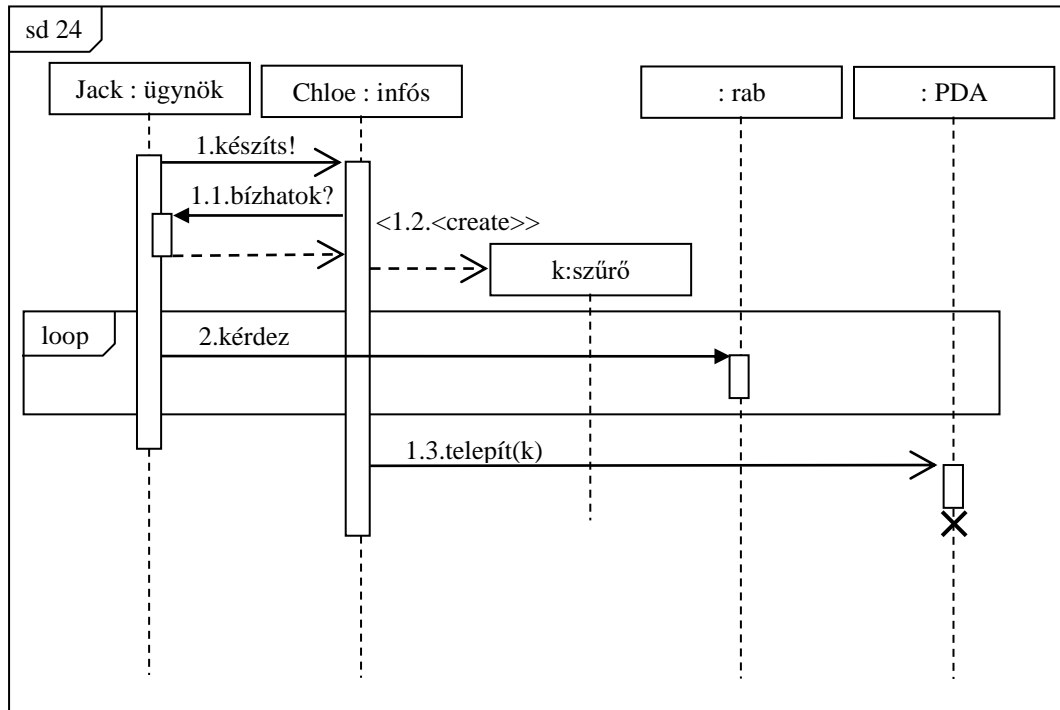
Izidor a zöldségesnél rájön, hogy lekvárt akar főzni, ezért SMS-ben megkérdi hűgát, Tektóniát, hogy van-e otthon befőző cukor. Tektónia átkutatja a kamrát, és talál cukrot, amiről (szintén SMS-ben) értesíti bátyját. Izidor eközben vicces szöveget ír a zöldséges hátára, amíg a választ meg nem kapja, majd hazamegy, és megfőzi a lekvárt. A kész lekvárt választ sem várva odadobja hűgának, és elsiet. Tektónia a lekvárt beteszi a kamrába.





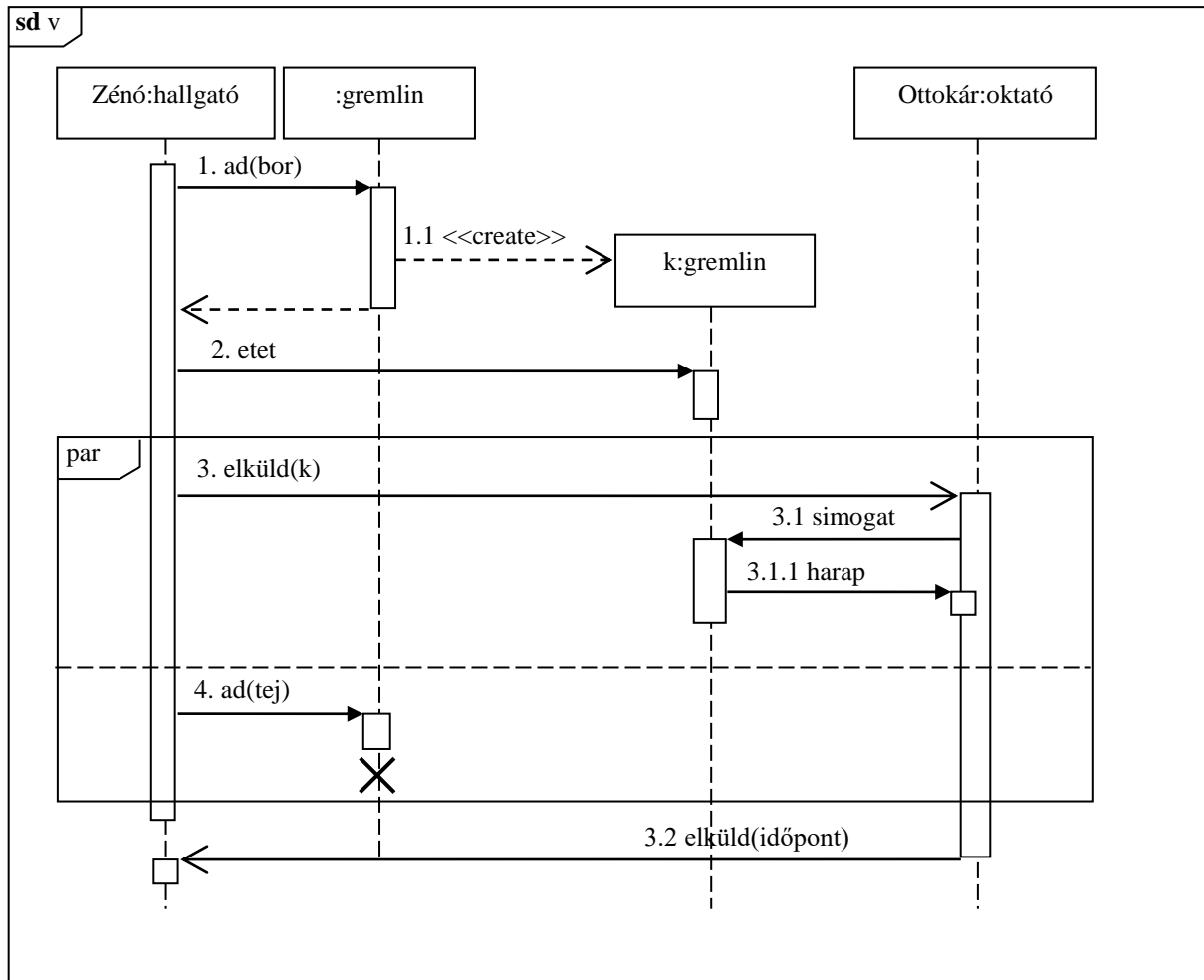
Készítsen UML2 szekvencia-diagramot az alábbi történet alapján! Ne feledkezzen el a hierarchikus számozásról sem!

Jack Bauer, a terroristák veszedelme visszatér. Információra van szüksége, ezért készített egy Kalman-szűrőt Chloe-val. Chloe, mielőtt nekilátna, megkérdi Jack-et, hogy megbízhat-e benne, majd elkezdí legyártani a szoftvert. Közben Jack az egyik rabot kihallgatja, és többször is megkérdezi, hogy hol vannak a fegyverek. A kihallgatás végére készül el a szűrő, amit Chloe elkezd feltelepíteni Jack PDA-jára, de közben másra is figyel. A PDA a telepítés hatására tönkremegy.



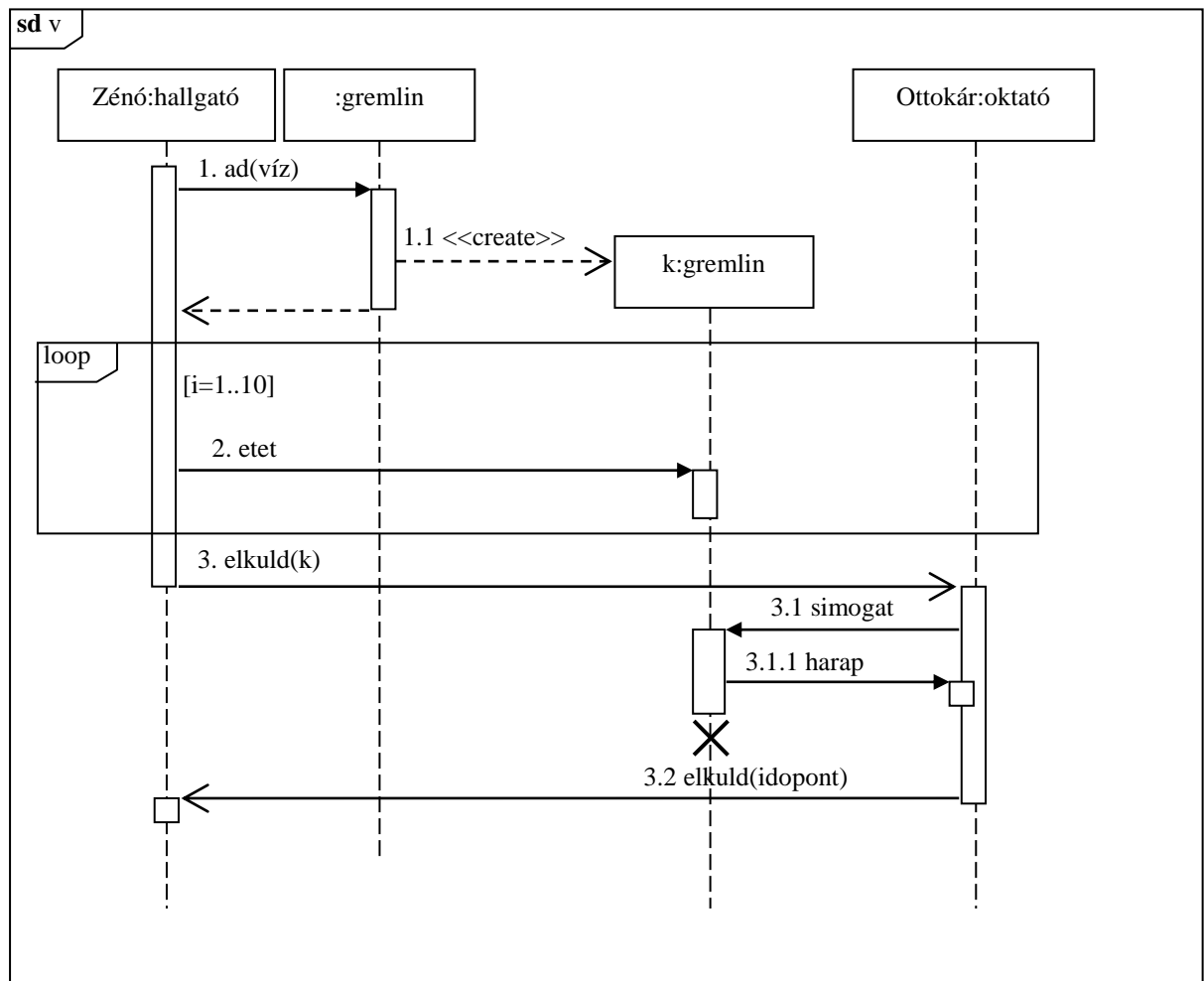
Készítsen UML2 szekvencia-diagramot az alábbi történet alapján! Ne feledkezzen el a hierarchikus számozásról sem!

Zénó a karácsonyra kapott gremlinjének véletlenül bort ad, mire az egy kisgremlinnek ad életet. Zénó nagyon megörül, és egyszer megeteti a kisgremlint, majd elküldi oktatójának, Gyíkcú Ottokárnak házi feladat helyett. Ottokár megsimogatja a kisgremlint, aki simogatás közben megharapja. Közben Zénó az eredeti gremlinnek tejet ad, aki ebbe belepusztul. Végül Ottokár elküldi Zénónak a pótleadás időpontját..



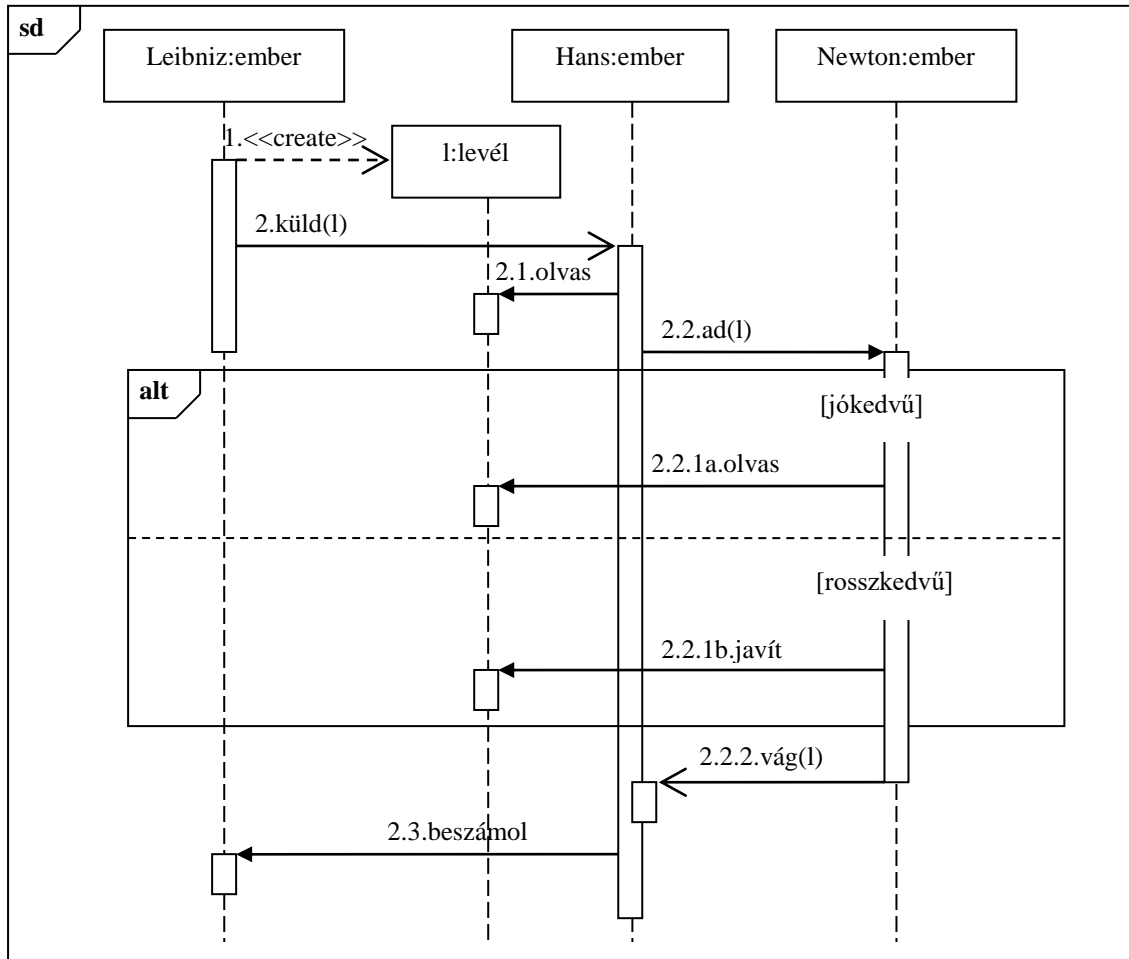
Készítsen UML2 szekvencia-diagramot az alábbi történet alapján! Ne feledkezzen el a hierarchikus számozásról sem!

Zénó a karácsonyra kapott gremlinjének véletlenül vizet ad, mire az egy kisgremlinnek ad életet. Zénó nagyon megörül, és 10-szer megeteti a kisgremlint, majd elküldi oktatójának, Gyíkcú Ottokárnak házi feladat helyett. Ottokár megsimogatja a kisgremlint, aki simogatás közben megharapja, de Ottokár vitriolos véréből el is pusztul. Ottokár elküldi Zénónak a pótleadás időpontját.



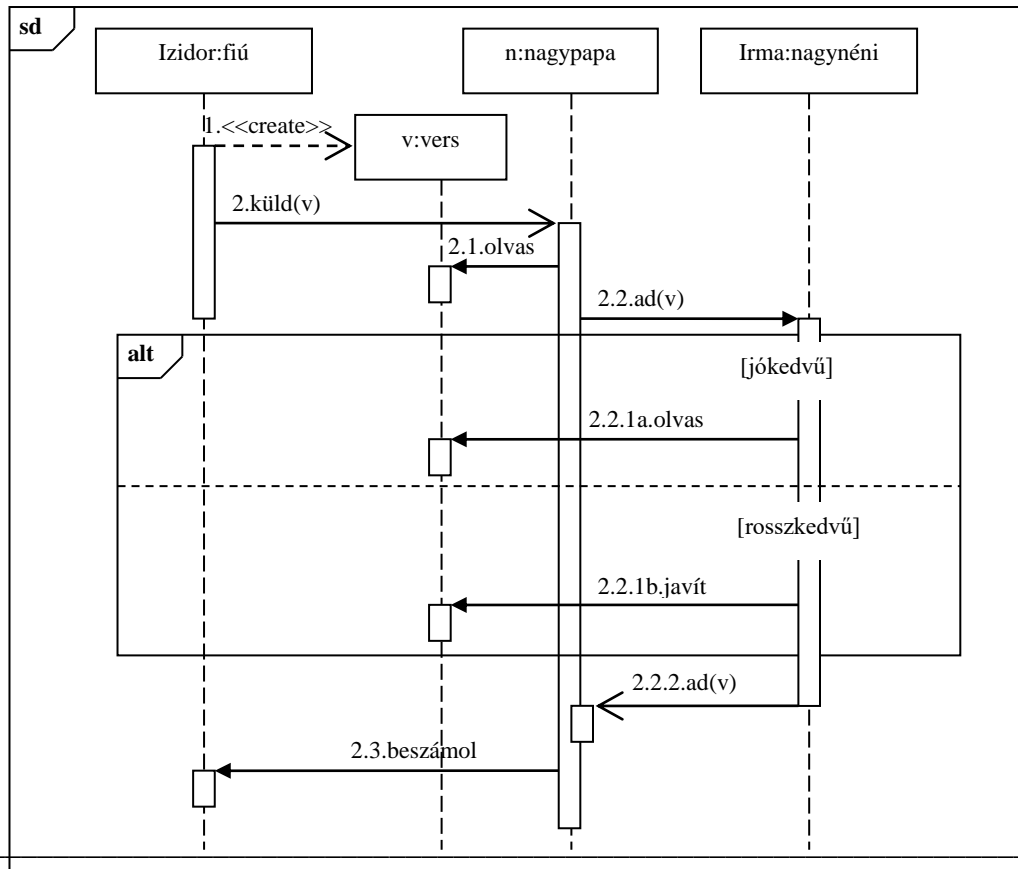
Készítsen UML2 szekvencia-diagramot az alábbi történet alapján! Ne feledkezzen el a hierarchikus számozásról sem!

Gottfried Wilhelm Leibniz szeretne tálkozni Sir Isaac Newtonnal. Ír egy latin nyelvű levelet, amelyben differenciálszámítással kapcsolatos eredményeit ecseteli. A levelet hű barátjának, az éppen Angliában tartózkodó Hans Georg von Hirscheissenfeldnek küldi azzal, hogy adja át Newtonnak. Hans kíváncsi, és elolvassa a levelet. Ezután találálkozik Newtonnal, és odaadja neki a levelet. Ha Newton jókedvű, akkor a levelet elolvassa, ha rosszkedvű, akkor a levélben aláhúzza a nyelvtani hibákat. Mindezek után a levelet hozzávágja Hanshoz, és elsiet. Hans hazautazik, és az eseményről beszámol Leibniznek.



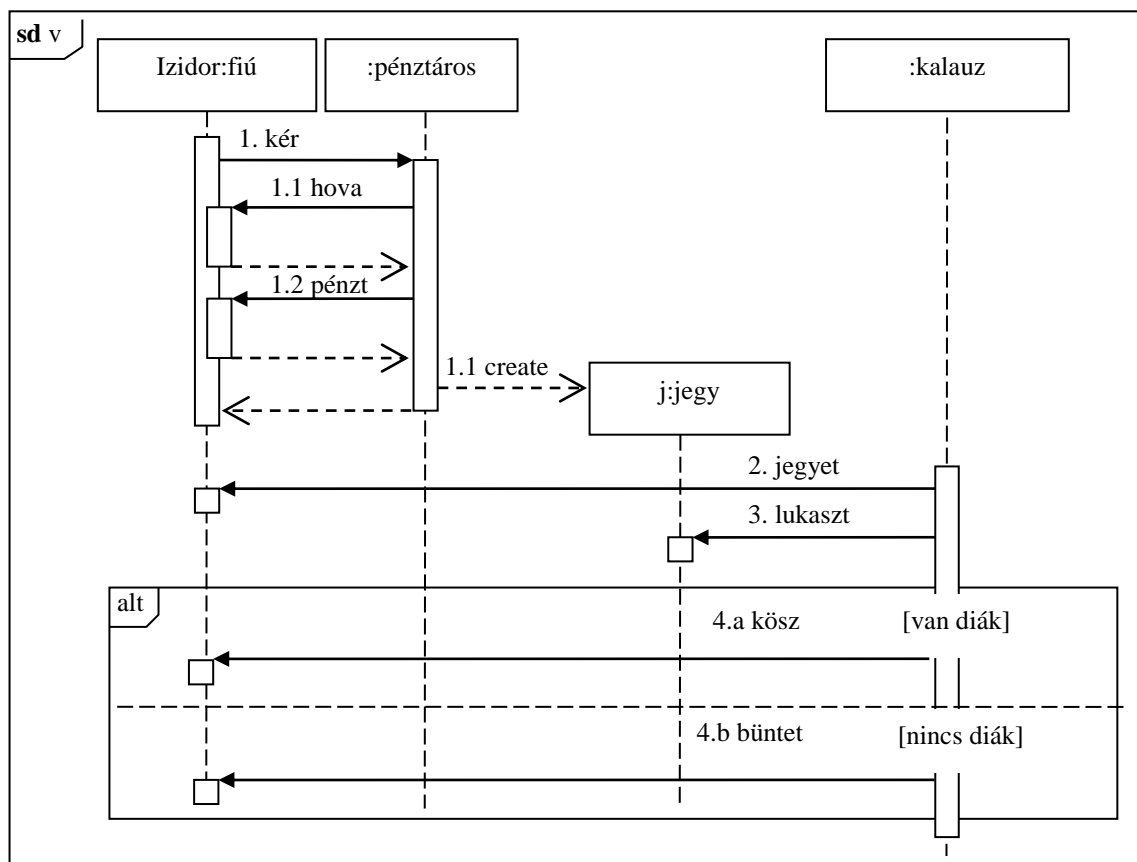
Készítsen UML2 szekvencia-diagramot az alábbi történet alapján! Ne feledkezzen el a hierarchikus számozásról sem!

Izidor verset ír ajándékként vidéken élő nagynénjének, Irmának. Az ajándékot a nagypapának küldi azzal, hogy adja át a nagynéninek. A nagypapa kíváncsi, és elolvassa a verset. Ezután odaadja Irmának, és várja a hatást. Ha a nagynéni jókedvű, akkor a verset felolvassa, ha rosszkedvű, akkor aláhúzza benne a nyelvtani hibákat. Mindezek után a verset visszaadja a nagypapának, és elsiet. A nagypapa ezután felutazik Pestre, és a történekről beszámol Izidornak.



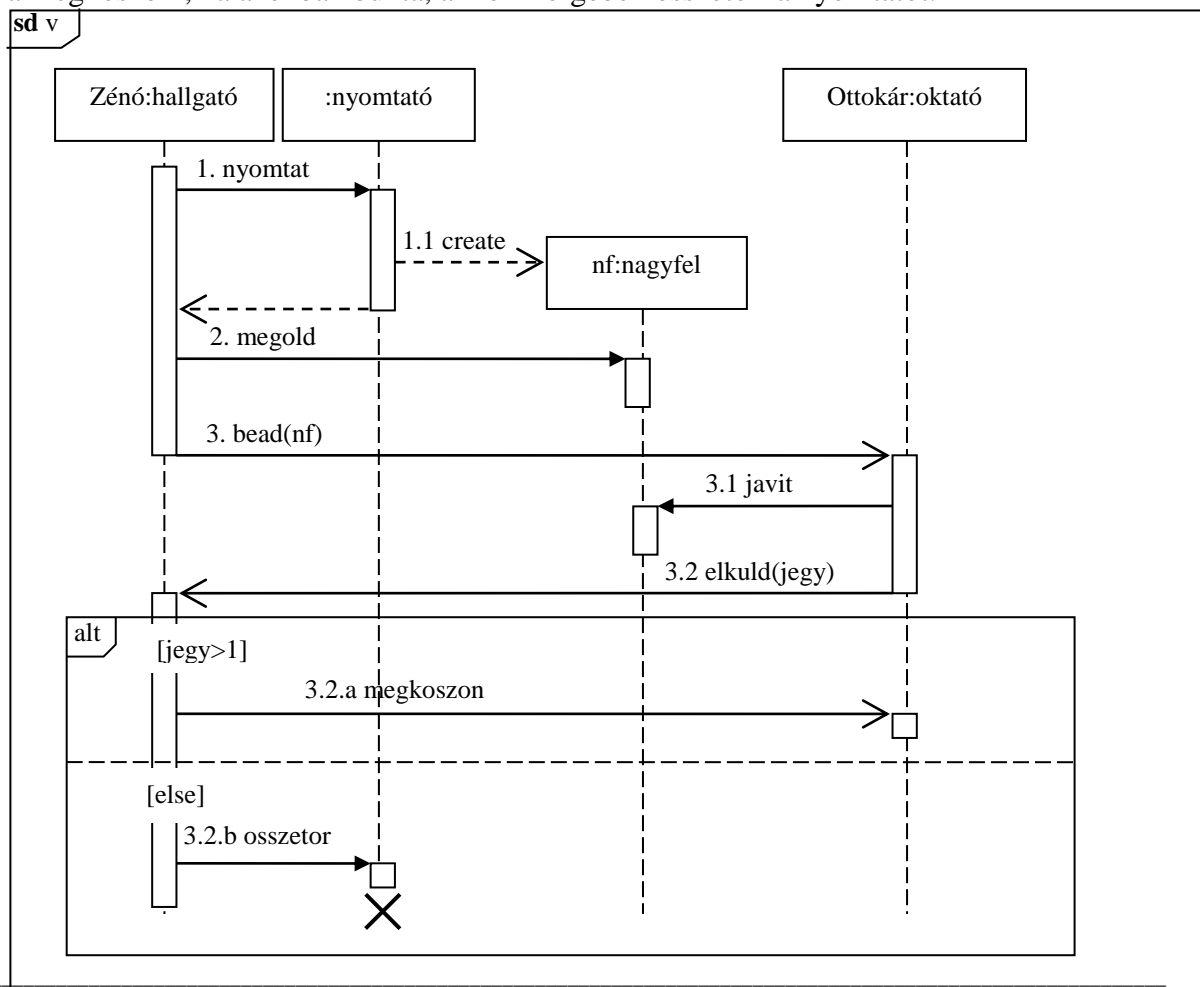
Rajzoljon UML 2.0 szekvenciadiagramot (sequence diagram) az alábbi leírás alapján!

Izidor vonatjegyet szeretne venni, hogy elutazzon nagymamájához. A jegypénztárnál kér egy retúrjegyet. A pénztáros megkérdezi, hogy hova. A pénztáros elkéri a pénzt, majd kinyomtatja a jegyet, és a visszajáróval együtt Izidornak adja. Később (már a vonaton) a kalauz elkéri a jegyet és kilukasztja. Ha Izidornál nincs nála a diákigazolványa, akkor (a kalauz) megbünteti, ha nála van, akkor (a kalauz) megköszöni.



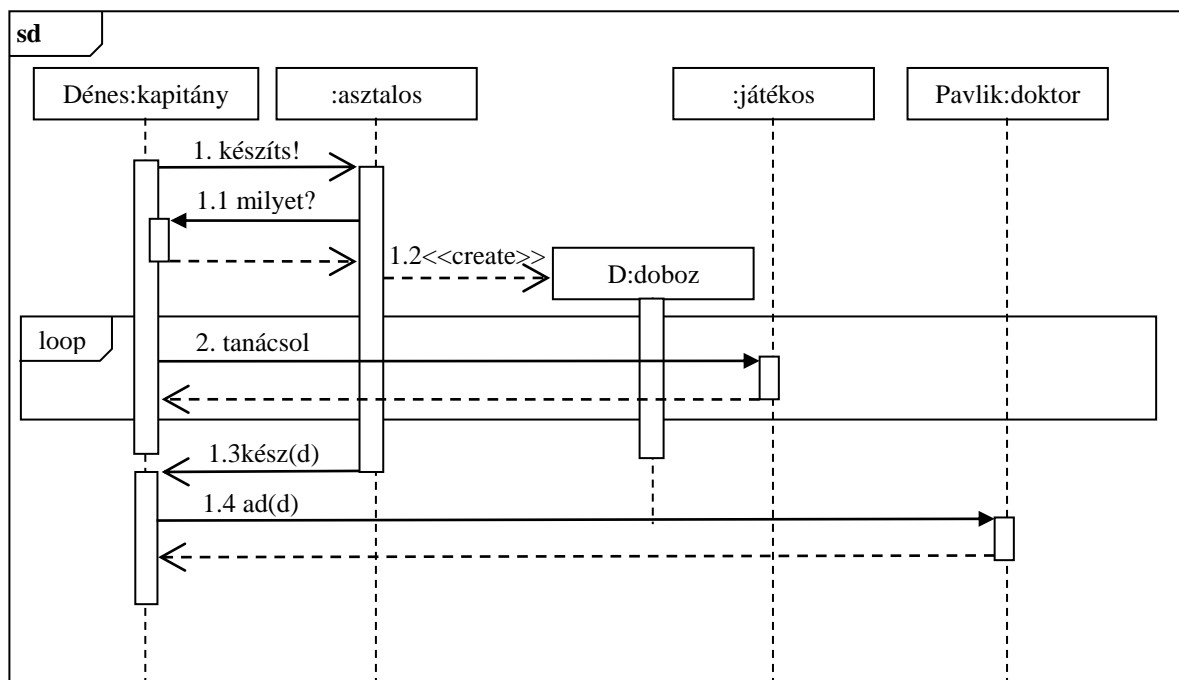
A történet alapján rajzoljon UML 2.0 szekvenciadiagramot (sequence diagram). **Az üzeneteket hierarchikus számozással lássa el !**

Zénó (Izidor bátyja) otthoni printerén kinyomtatja Bitgörbítés nagyfeladatát, a papíron megoldja, és elküldi Gyíkarcú Ottokárnak, a tárgy oktatójának javításra. Ottokár egyből nekilát és kijavítja a feladatot, majd ezzel a lendülettel vissza is küldi a jegyet Zénónak. Zénó, ha jobb jegyet kap, mint elégtelen, akkor választ sem várva megköszöni, ha azonban bukta, akkor mérgében összetöri a nyomtatót.



Rajzoljon UML 2.0 szekvenciadiagramot (sequence diagram) az alábbi leírás alapján!

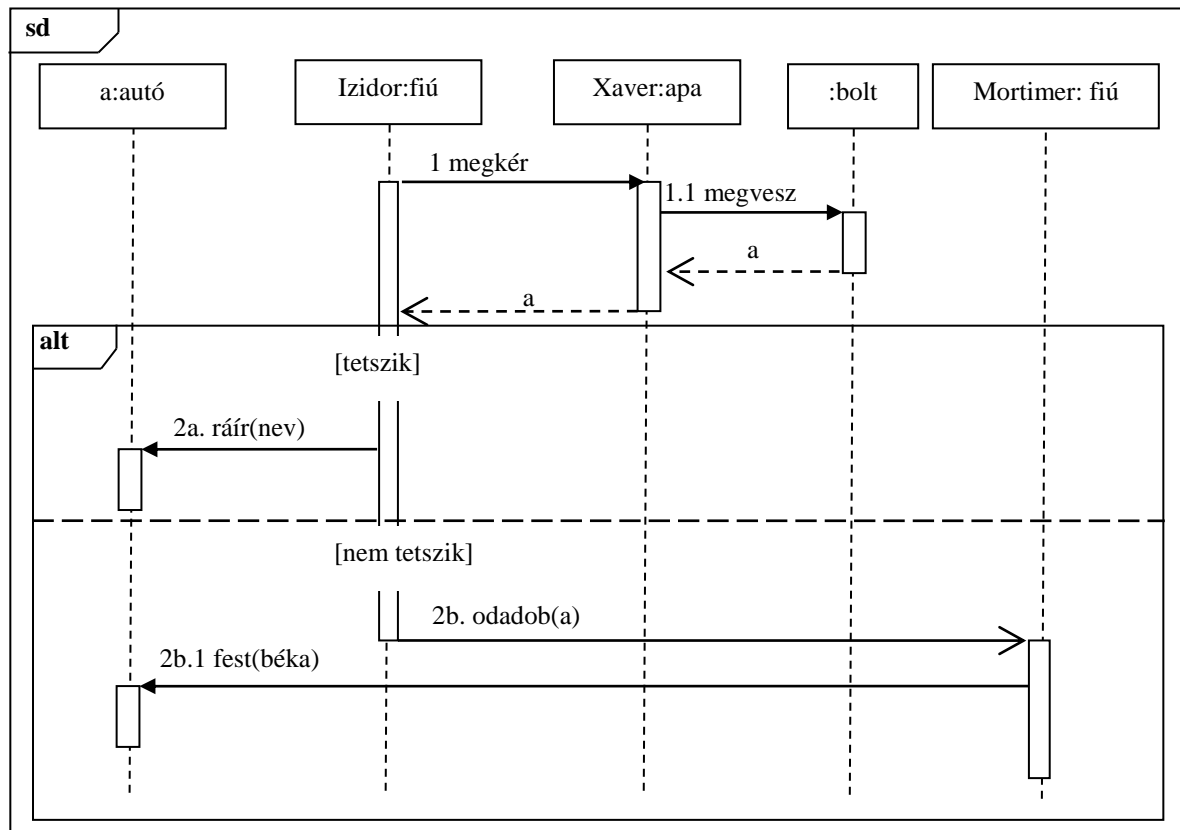
Kemény Dénes (a vízilabda-válogatott szövetségi kapitánya), mivel tudja, hogy Pavlik doktornak nincs miben tartania az olimpiai aranyait, ezért egy intarziás fadobozt csináltat kedvenc asztalosával. Az asztalos, mielőtt elkezdené a munkát, megkérdi Dénest, hogy pontosan milyen minta legyen a dobozon, majd elkezdí legyártani a remekművet. Közben a kapitány minden egyes játékosát egyenként szakmai tanácsokkal látja el. Mikor a doboz elkészül, az asztalos elküldi Dénesnek, aki fogja, és azon nyomban átadja a doktornak.





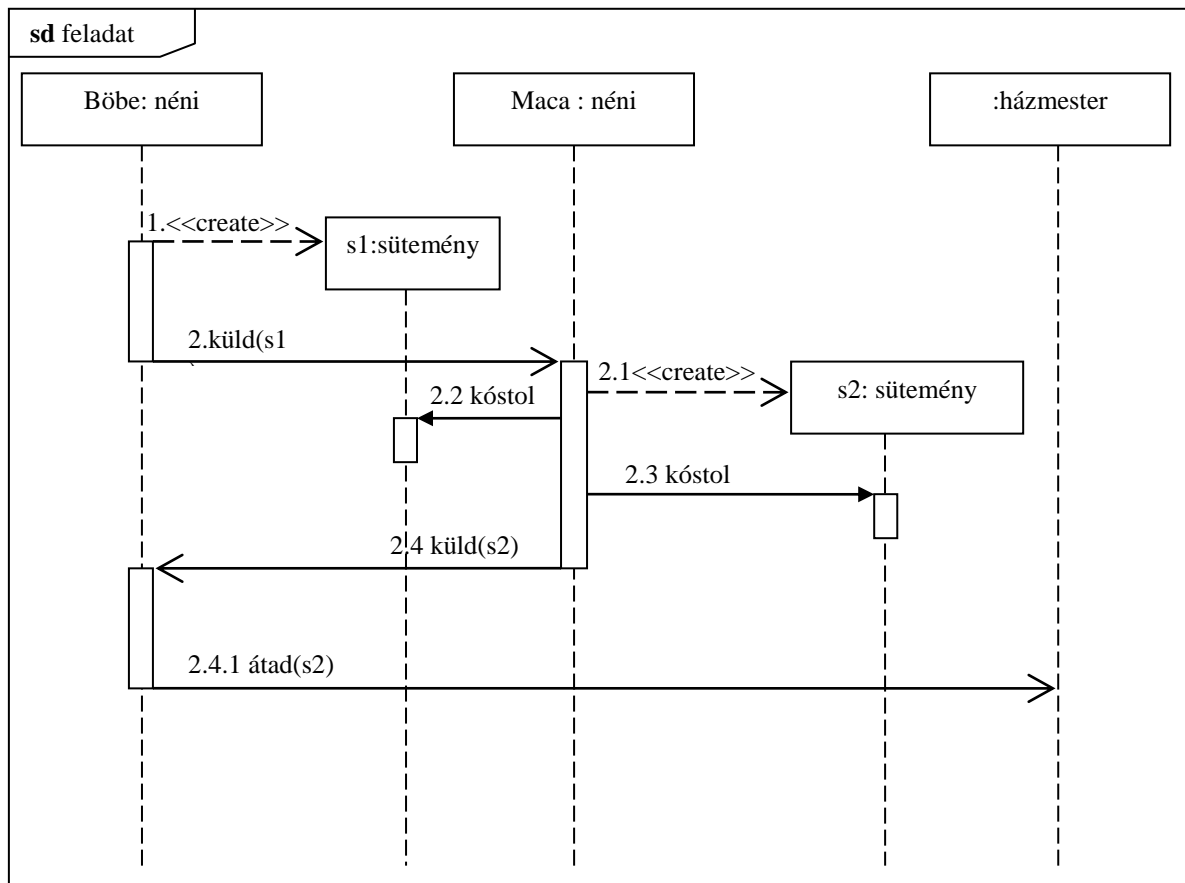
Rajzoljon UML2 **szekvenciadiagramot** ! Az üzeneteket hierarchikus számozással lássa el !

Izidor megkéri apukáját, Xavért, hogy vegyen neki egy versenyautót, mire együtt elmennek a boltba, ahol Xavér megveszi a fiának a versenyautót, majd elsiet a dolgára. Izidornak ha tetszik az autó, akkor ráírja a nevét, ha nem, akkor odadobja az öccsének, Mortimernek és elrohan. Utóbbi esetben Mortimer az autóra ráfest egy békát.



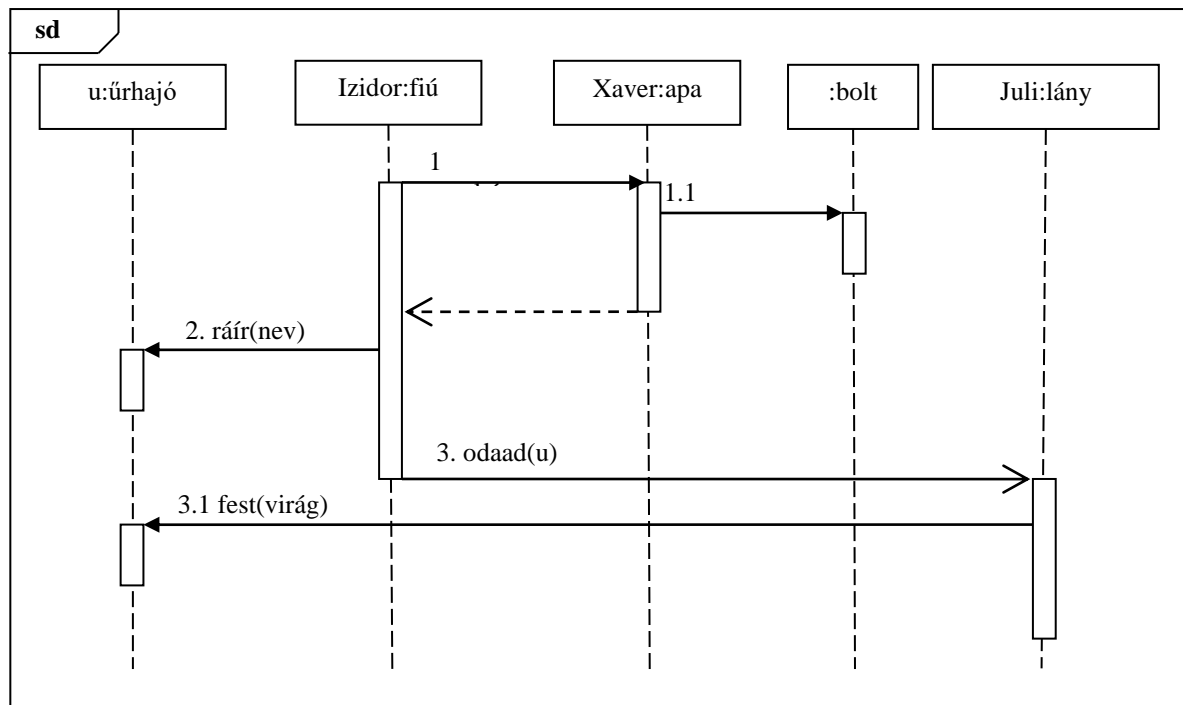
Rajzoljon UML2 **szekvenciadiagramot** ! Az üzeneteket hierarchikus számozással lássa el !

Böbe néni süt egy bejglit, és elküldi Maca néninek. Maca erre gyorsan süt egy másikat, mindkettőt megkóstolja, és a sajátját visszaküldi Böbének. Böbe a süteményt, választ sem várva, átküldi a házmesternek.



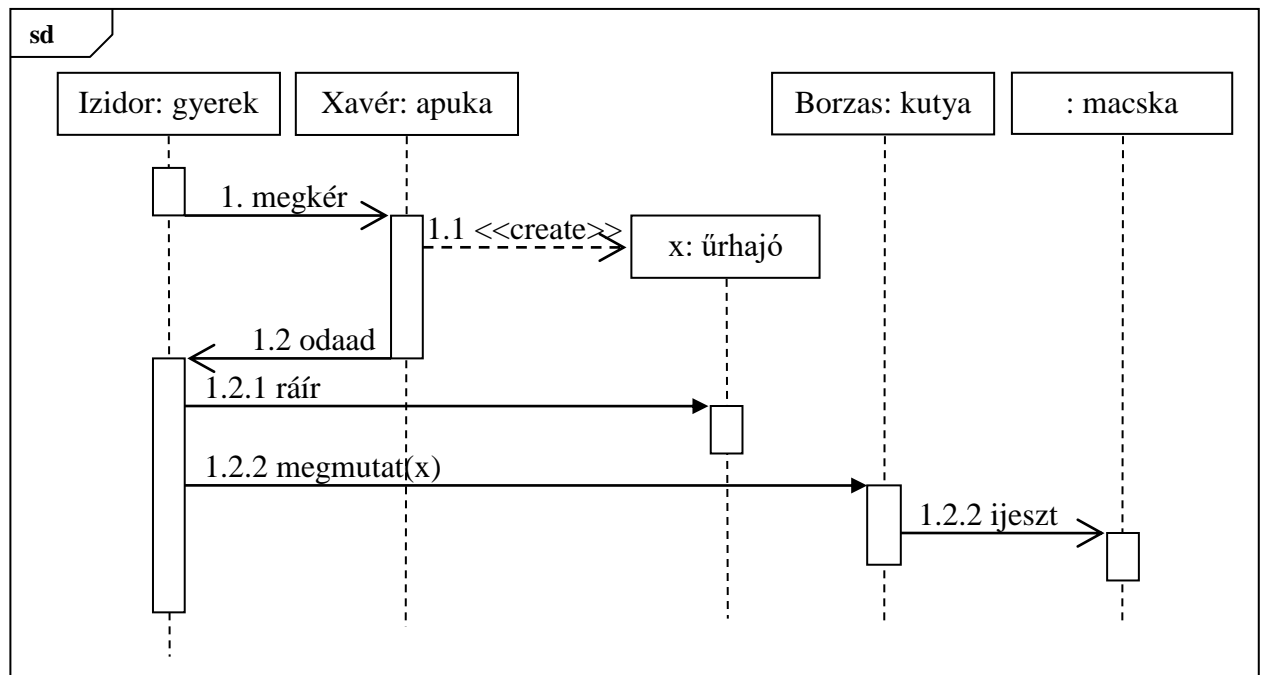
Rajzoljon UML2 **szekvenciadiagramot** ! Az üzeneteket hierarchikus számozással lássa el !

Izidor megkéri apukáját, Xavért, hogy vegyen neki egy úrhajót, mire együtt elmennek a boltba. Xavér megveszi az úrhajót a boltban, odaadja Izidornak, majd elsiet, mert sok a dolga. Izidor az úrhajóra ráírja a nevét, majd odaadja Julinak és elrohan. Juli az úrhajóra ráfest egy virágot.



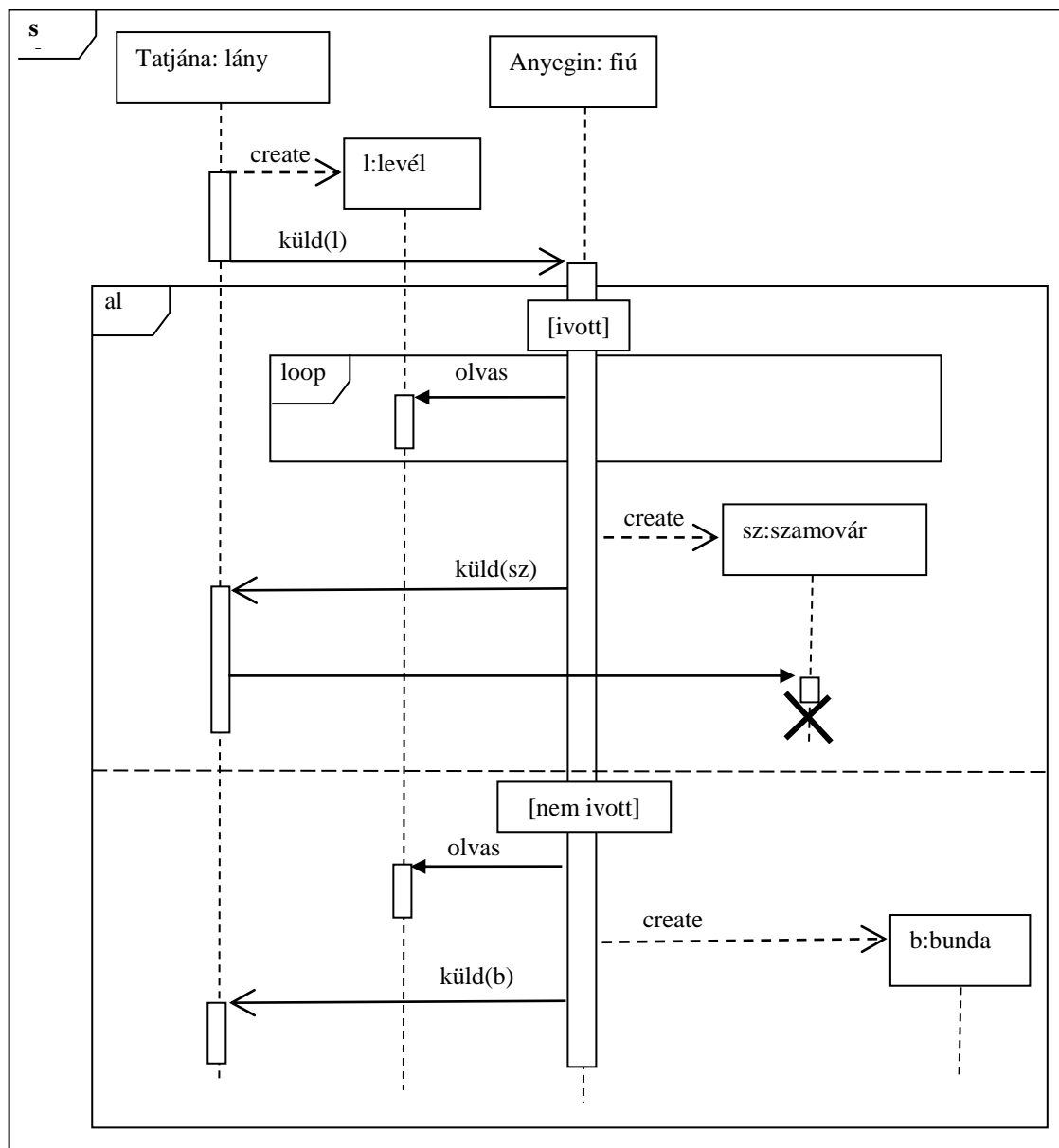
Rajzoljon UML2 **szekvenciadiagramot** ! Az üzeneteket hierarchikus számozással lássa el !

Izidor megkéri apukáját, Xavért, hogy készítsen neki egy úrhajót. Mikor Xavér kész van, az úrhajót odaadja Izidornak, majd elsiet, mert sok a dolga. Izidor az úrhajóra ráírja a nevét, majd megmutatja a legjobb barátjának, Borzas kutyának, aki a mutogatás közben megijeszt egy macskát.



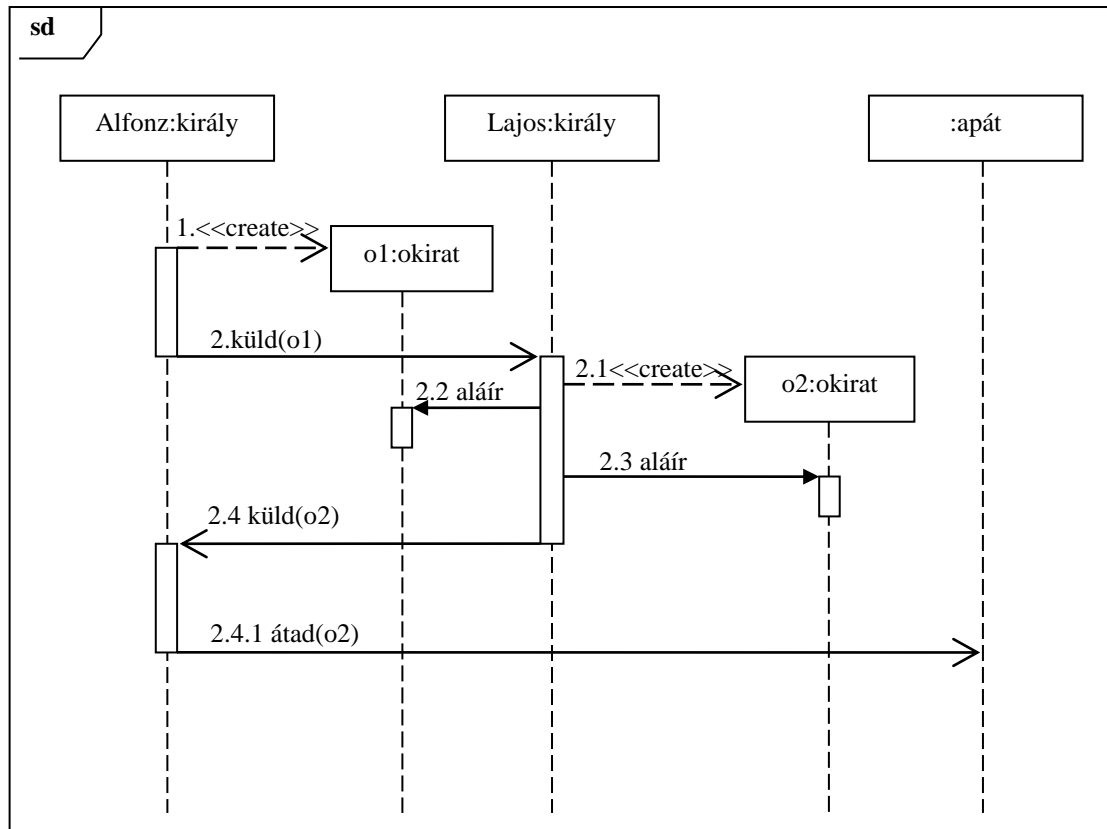
Rajzoljon UML 2 szekvenciadiagramot !

Tatjana levelet ír, majd elküldi Anyeginnek. Ha Anyegin ivott vodkát, akkor többször is elolvassa a levelet, majd készít egy gyönyörű, aranyozott számovárt, és elküldi Tatjánának, aki (mivel nem erre számított) megsemmisíti a számovárt. Ha azonban Anyegin nem ivott, akkor csak egyszer olvassa el a levelet, majd bundát csinál, és ezt a bundát küldi Tatjánának.

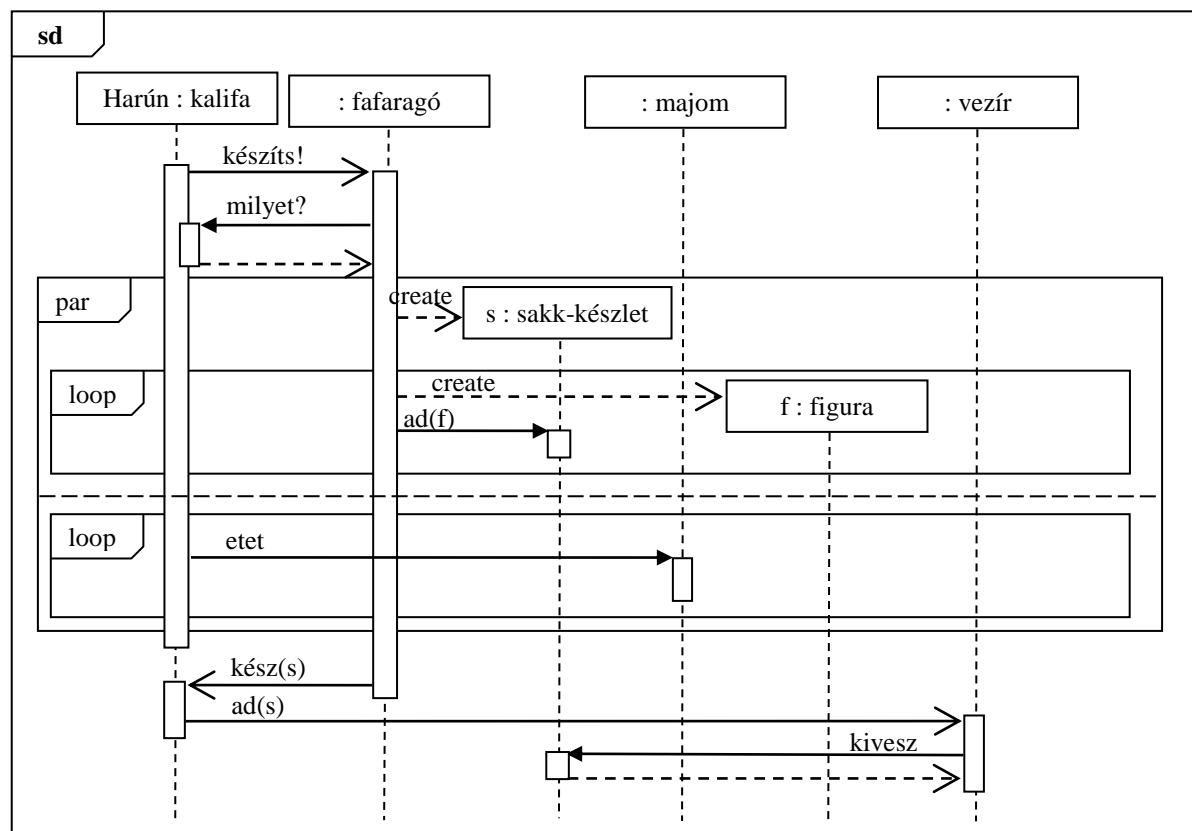


Rajzoljon UML 2 szekvenciadiagramot !

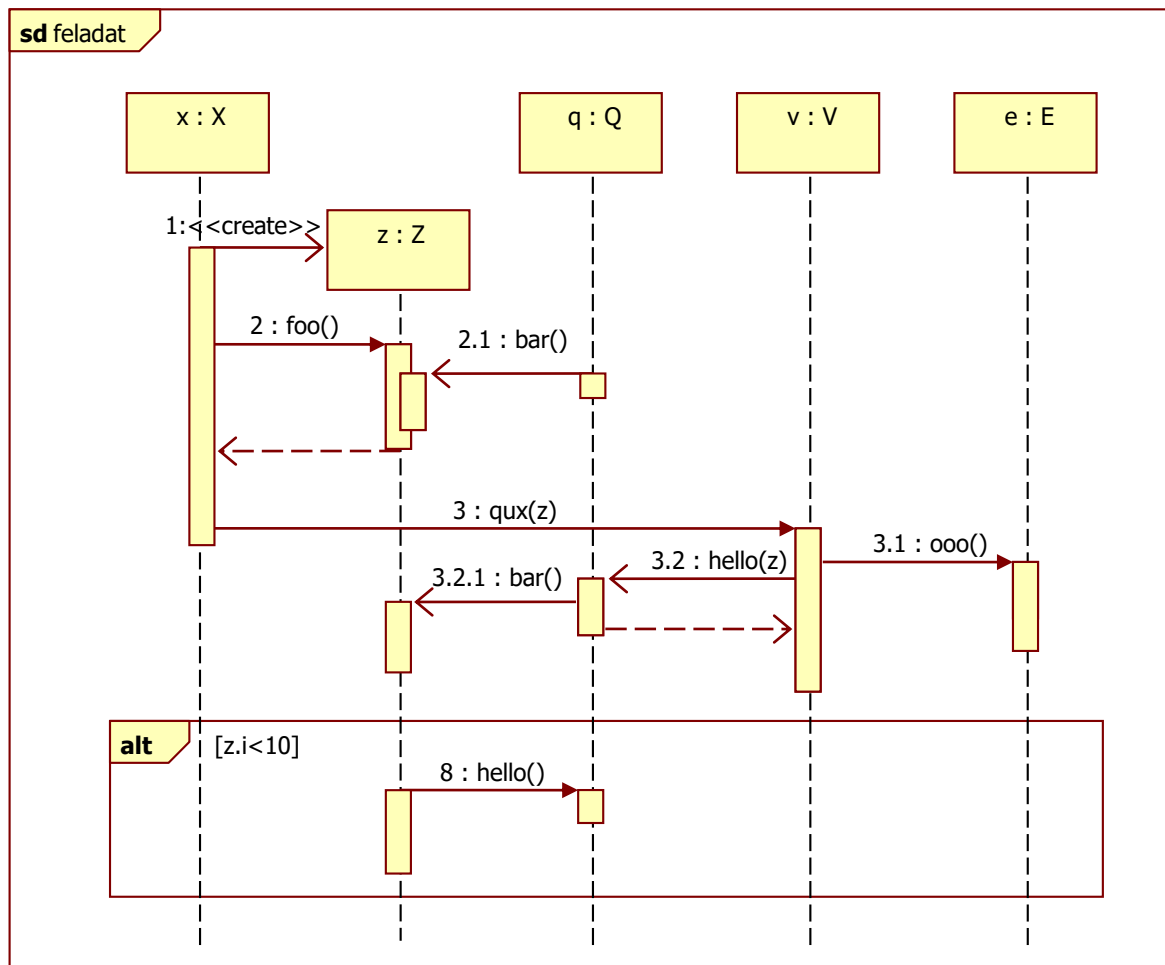
X. (Bölcs) Alfonz, Kasztília királya szövetséget kíván kötni IX. (Szent) Lajossal, Franciaország királyával. Ezért elkészít egy okiratot, amit el is küld a francia uralkodónak. Lajos, amint megkapja a dokumentumot, készített egy másolatot, majd mindkettőt kézjeggyével látja el, és a másolatot visszaküldi Alfonz-nak, aki az okiratot a toledói apátnak adja át megőrzésre.



Harún ar-Rasíd, a bölcs kalifa meg kívánja jutalmazni vezírjét, ezért készített egy sakk-készletet udvari fafaragójával. A fafaragó, mielőtt nekilátna, megkérdi a kalifát, milyen fából készüljön, majd elkezdí legyártani a remekművet. A készlethez egyenként faragja figurákat. Eközben a kalifa a kedvenc majmával szép sorban egyenként megeteti a kezében levő összes fügét. Mikor a sakk-készlet megvan, a faragó átadja a kalifának, aki továbbadja vezírjének. A vezír később, otthon kivesz a készletből egy fehér lovat.



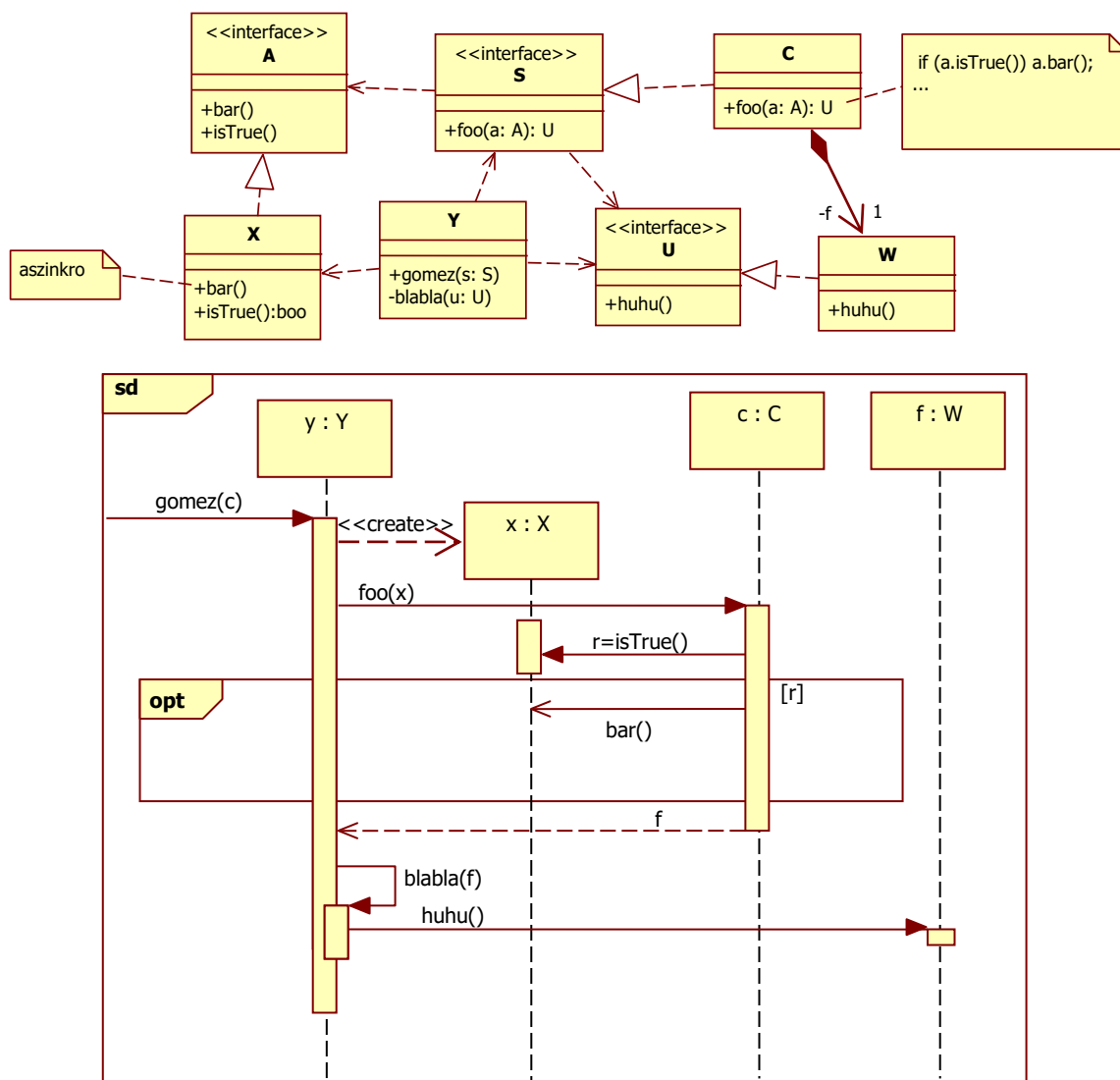
Az alábbi UML2 szekvenciadiagramon szintaktikai (jelölésbeli) és szemantikai (értelmi) hibák is előfordulnak. Sorolja fel őket!



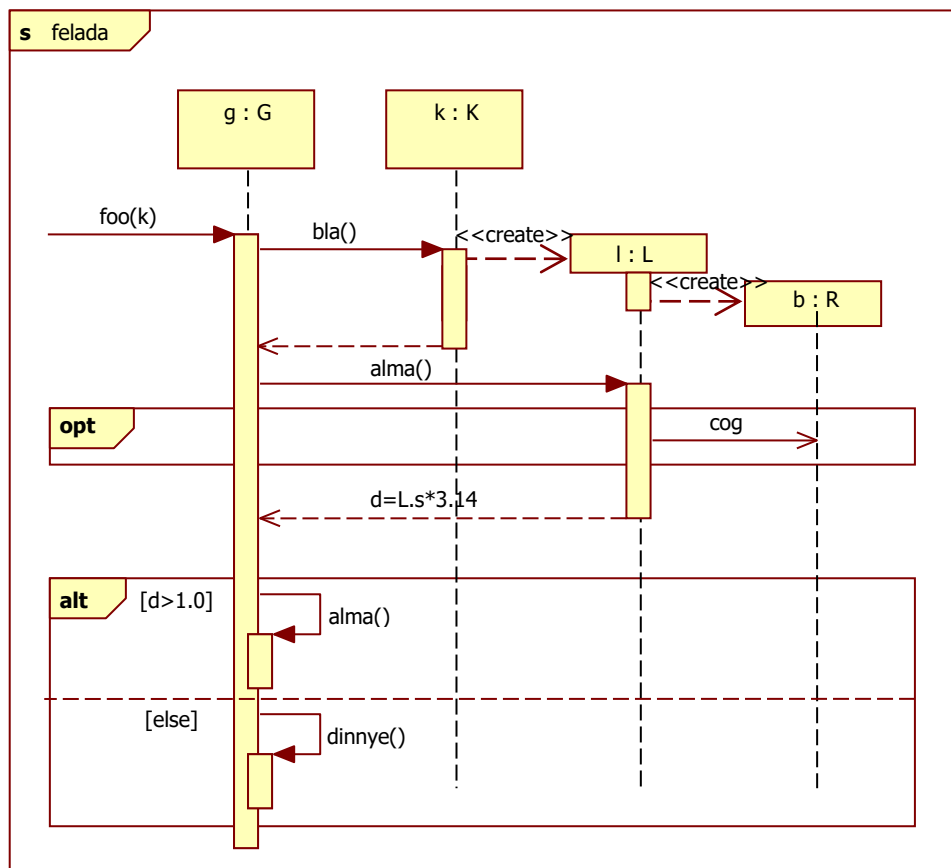
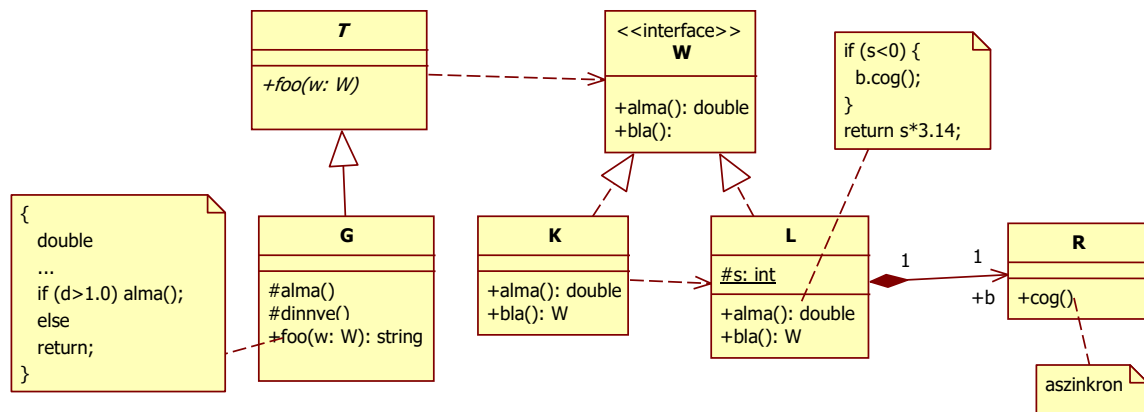
1. create nyila rossz
2. bar a semmiből, q nem ismerheti z-t
3. qux(z) után x aktivitása megszűnik, aszinkron kellene legyen
4. ooo után a visszatérés előtt hello hívódik, ooo aszinkron kellene legyen
5. hello aszinkron, utána visszatérés.
- 6: alt doboz csak egy rekesszel
7. hello a semmiből.



Az alábbi UML2 osztálydiagram alapján rajzoljon olyan UML2 szekvenciadiagramot, amin minden metódus pontosan egyszer szerepel (az azonos szignatúrájú metódusok közül is csak egy szerepeljen). Számozást nem szükséges alkalmaznia. Vegye figyelembe a metódusokra vonatkozó megjegyzésdobozokban szereplő Java kódrészleteket is! A "... " jelölés a nem specifikált részleteket jelöli. Minden visszatérési értéket használjon fel! Az első, kívülről jövő metódushívás legyen egy helyesen paraméterezett *gomez(s:S)*.



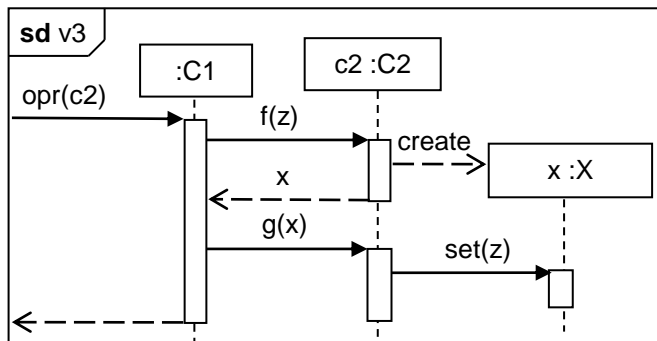
Az alábbi UML2 osztálydiagram alapján rajzoljon olyan UML2 szekvenciadiagramot, amin minden metódus pontosan egyszer szerepel (az azonos szignatúrájú metódusok közül is csak egy szerepeljen). Számozást nem szükséges alkalmaznia. Vegye figyelembe a metódusokra vonatkozó megjegyzésdobozokban szereplő Java kódrészleteket is! A "... " jelölés a nem specifikált részleteket jelöli. Minden visszatérési értéket használjon fel! Az első, kívülről jövő metódushívás legyen egy helyesen paraméterezett *foo(w:W)*.



Tételezze fel, hogy az alábbi UML2 szekvenciadiagramon szereplő objektumok osztályai között nincs más egyéb – a diagramból nem kiolvasható – kapcsolat (pl. öröklés) ! Mi a kapcsolat C2 és X között ? Válaszát egy, a magyar nyelv szabályainak megfelelő, olvasható MONDATtal indokolja ! Indoklás nélkül a választása nem érvényes.

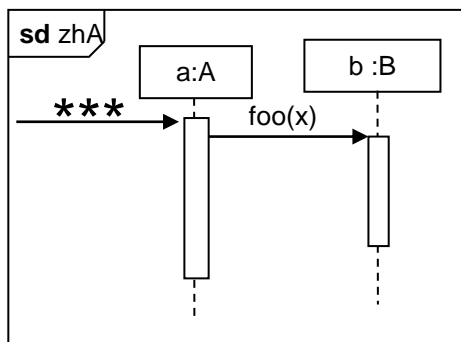
- ☐ példányosítás (instantiation)
- ☐ asszociáció (association)
- ☐ kollaboráció (collaboration)
- ☒ függőség (dependency) C2 függ X-től
- ☐ függőség (dependency) X függ C2-től
- ☐ interakció (interaction)
- ☐ implementálás (implementation)

Indoklás:

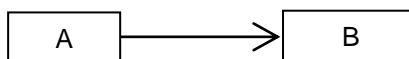


**C2 létrehoz, használ, de nem kell emlékeznie .....**

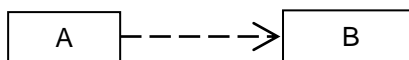
Tételezze fel, hogy az alábbi (zhA nevű) UML2 szekvenciadiagramon szereplő objektumok osztályai között nincs más egyéb – a diagramból nem kiolvasható – kapcsolat (pl. öröklés) ! Rajzolja be az A és B között kapcsolatot, ha



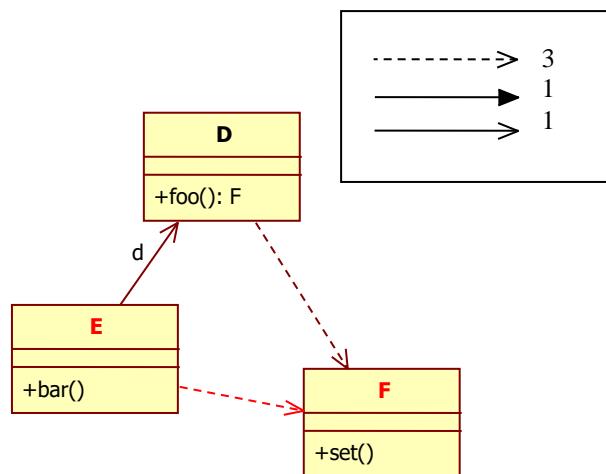
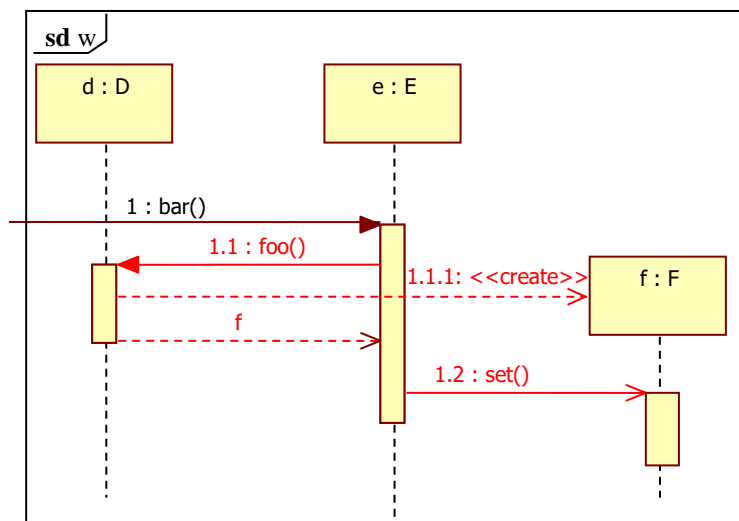
\*\*\* = bar(x)



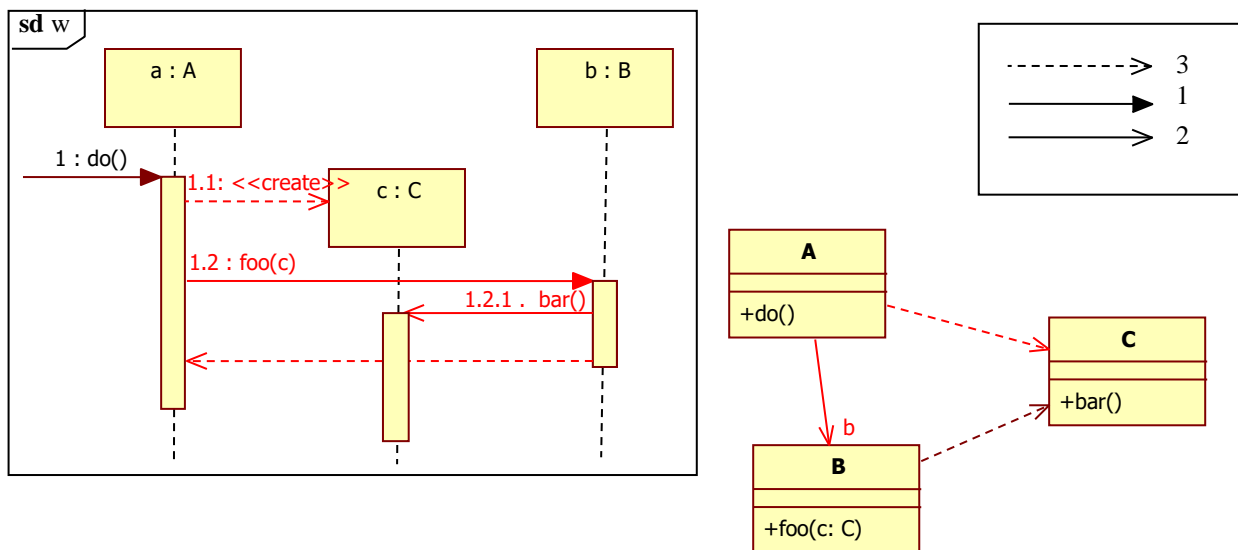
\*\*\* = qwx(b)



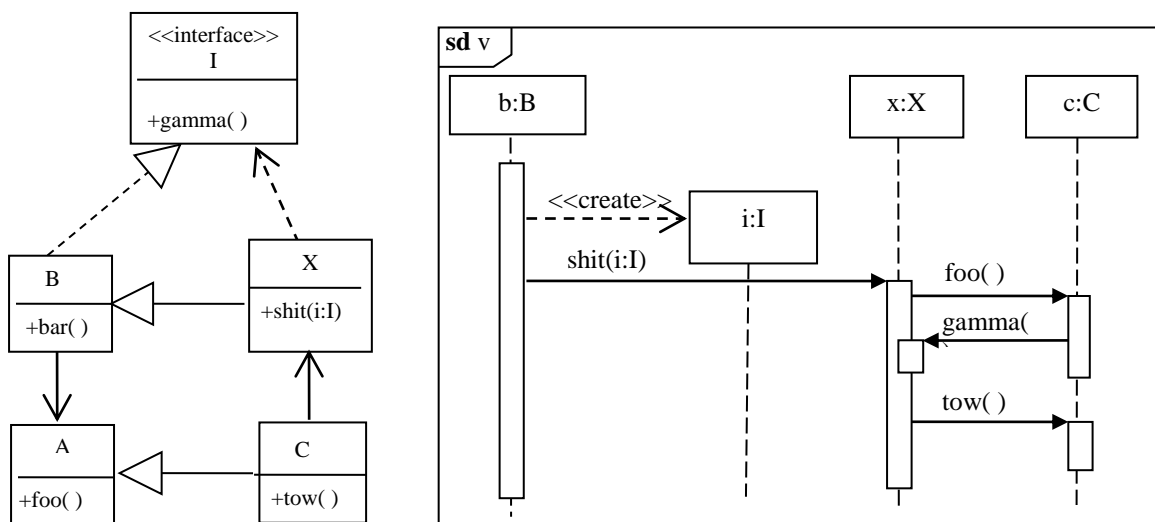
Az ábrán egy UML2 osztálydiagram és egy UML2 szekvenciadiagram látszik. A két diagram szemantikailag összefügg, de hiányos. Rajzolja be a hiányzó jelölő-elemeket! Ahol lehet, lássa el őket feliratokkal is! A felhasználható jelölőelemek és számosságuk a mellékelt keretben látható.



Az ábrán egy UML2 osztálydiagram és egy UML2 szekvenciadiagram látszik. A két diagram szemantikailag összefügg, de hiányos. Rajzolja be a hiányzó jelölő-elemeket! Ahol lehet, lássa el őket feliratokkal is! A felhasználható jelölőelemek és számosságuk a mellékelt keretben látható.



Az ábrán egy UML2 osztálydiagram és egy UML2 szekvenciadiagram látszik. A két diagram szemantikailag összefügg. Az osztálydiagramot hibátlannak tekintve milyen hibákat, ellentmondásokat talál a szekvenciadiagramban? Sorolja fel a hibákat !

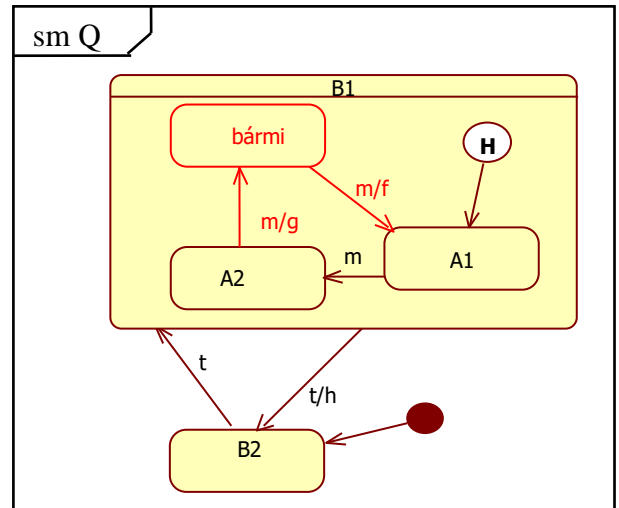
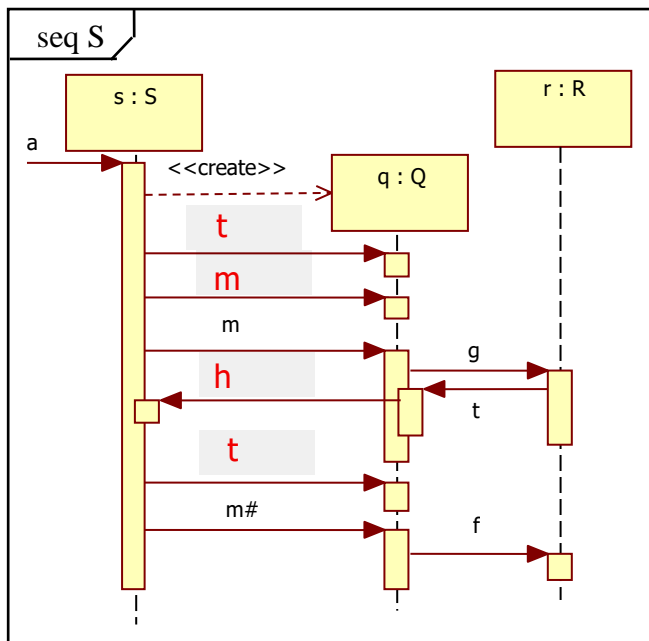


**Interfész nem példányosítható**

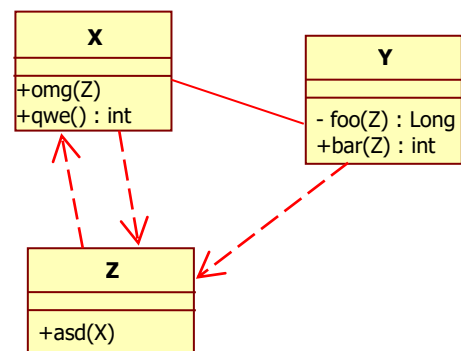
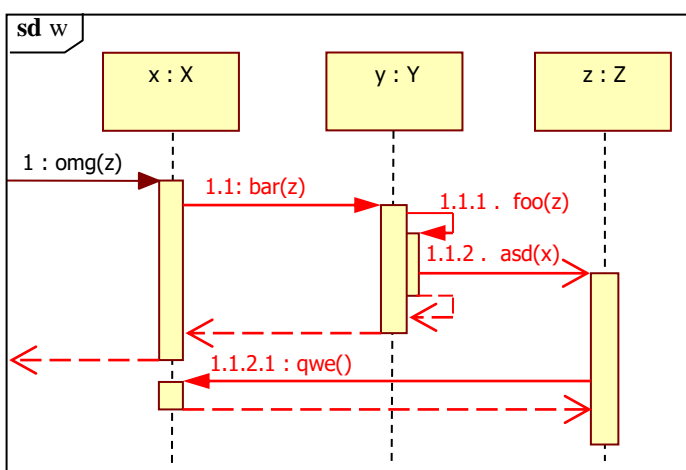
**B nem látja X-et, ezért annak shit(i:I) metódusa nem hívható**

**X nem láthatja C tow() metódusát**

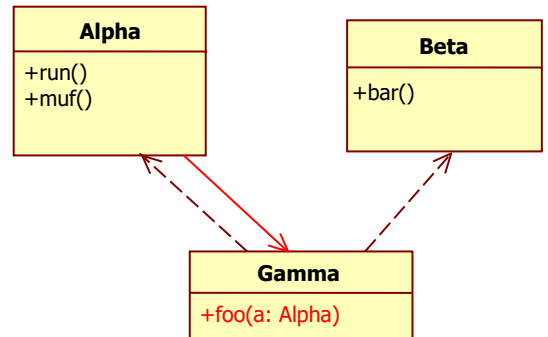
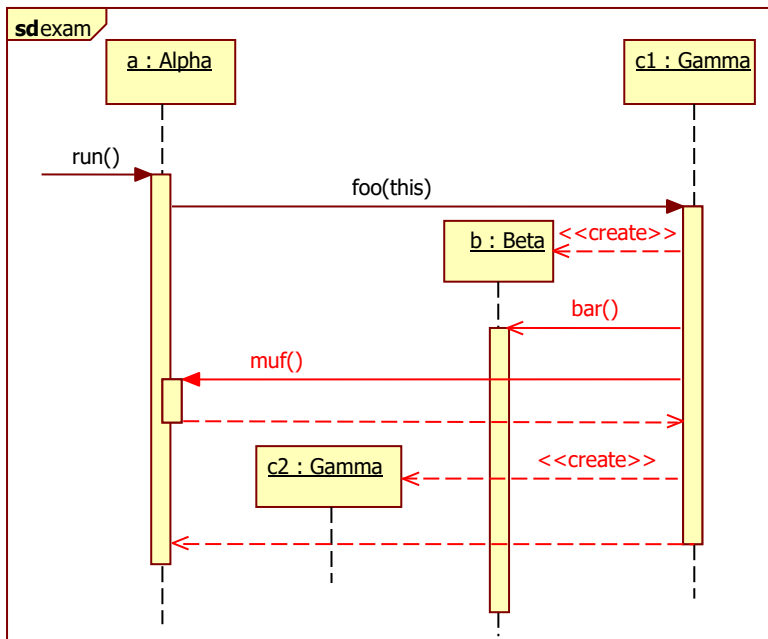
Az állapotgépet CSAK kiegészíteni szabad, az adottakat módosítani NEM!



Izidor rajzolt egy UML2 osztály- és egy szekvenciadiagramot. A két diagram szemantikailag összefügg. Izidor öccse, Ágoston kitörölt néhány elemet a rajzokról. A szekvenciadiagramról hiányoznak az üzenetek, az osztálydiagramról a kapcsolatok. Rajzolja be a hiányzó jelölő-elemeket! Az üzeneteket hierarchikusan számozza! Minden metódus egyszer hívódik meg. A híváskor kapott paraméter értékes a hívás előtt nem volt ismert a hívott számára.

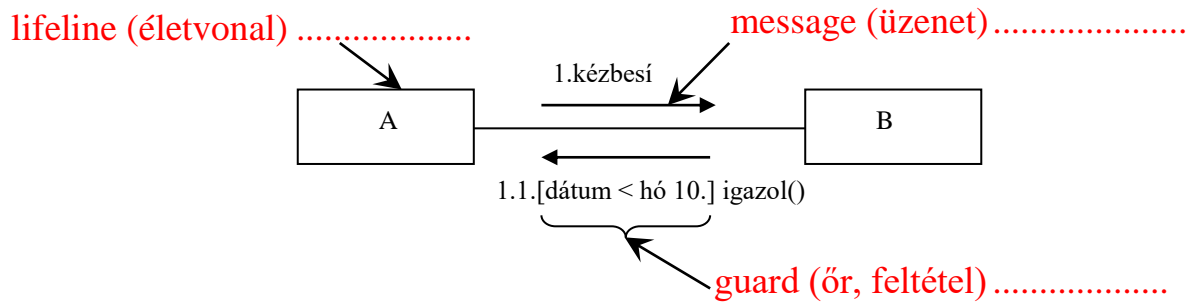


Izidor rajzolt egy UML2 osztály- és egy szekvenciadiagramot. A két diagram szemantikailag összefügg. Izidor öccse, Ágoston kitörölt néhány elemet a rajzokról. A szekvenciadiagramról hiányoznak üzenetek, az osztálydiagramról két UML2 elem. Rajzolja be a hiányzó elemeket! Minden metódus, csak egyszer hívódik meg! Jelölje be a hívásból történő visszatérést is!

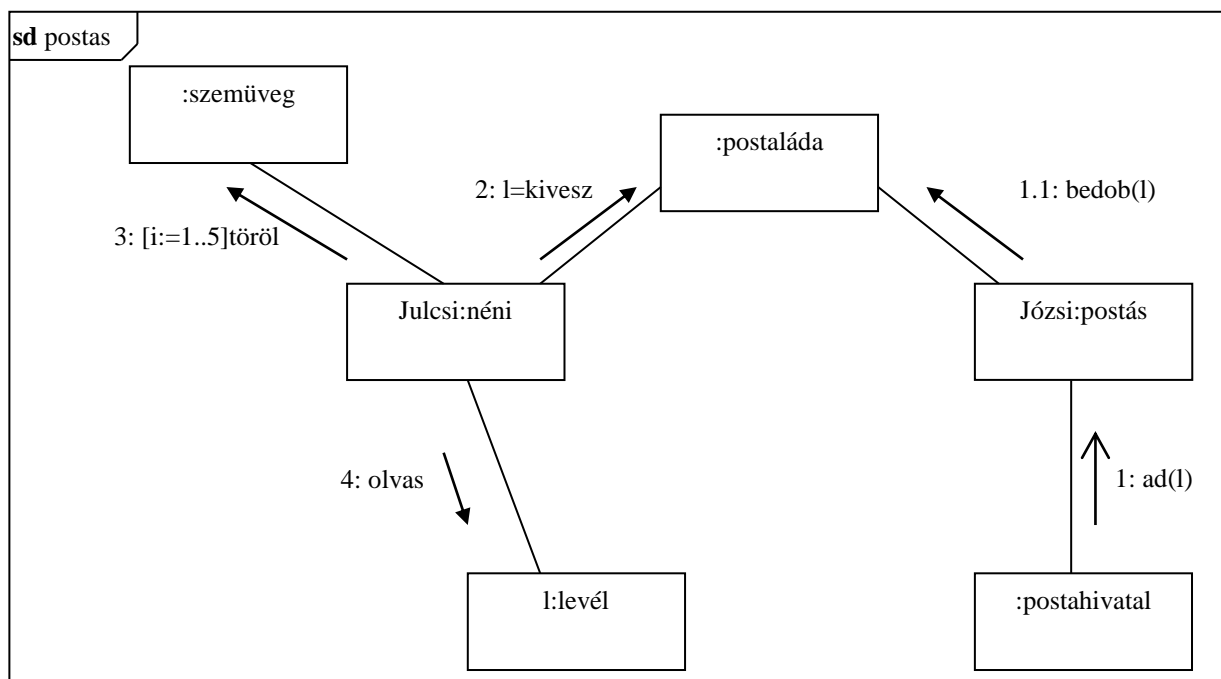


## 6 Kommunikációs diagram

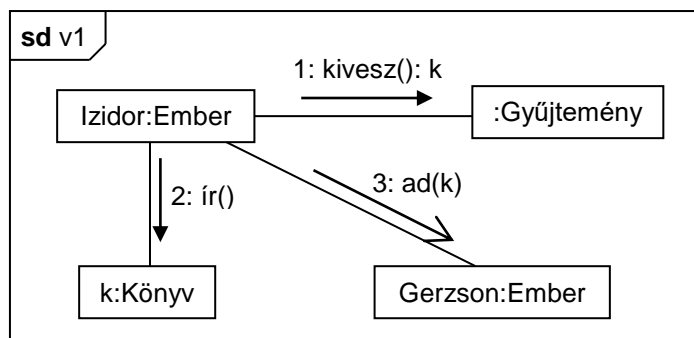
4. Az alábbi ábrán három UML2 modell elemet megjelöltünk. Adja meg elemenként, hogy az melyik UML2 meta-modell elem példánya !



A postahivatal Józsinak, az énekes postásnak adja a Julcsi néninek szóló levelet. A postás a levelet bedobja a címzett postaládájába. Julcsi néni este, hazamenet, kiveszi a postaládából a levelet, majd ötször megtörli a szemüvegét (az utcán esett az eső), és elolvassa levelet. Rajzoljon UML2 kommunikációs diagramot !

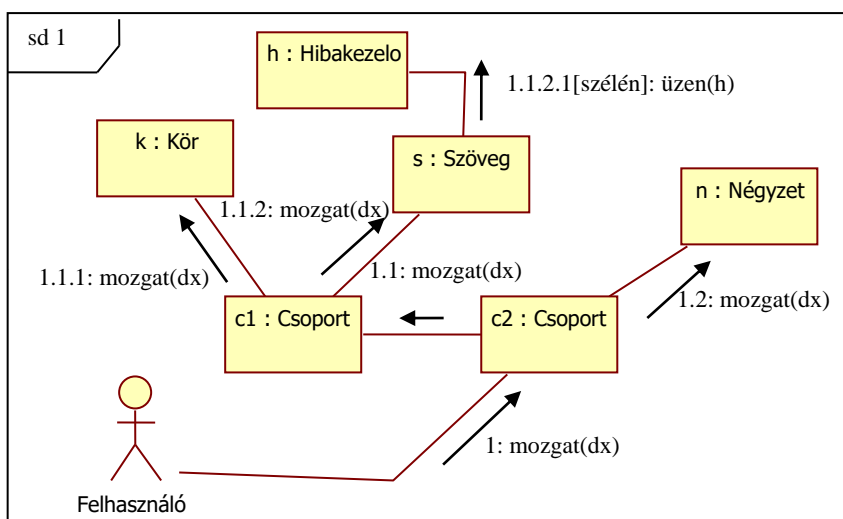
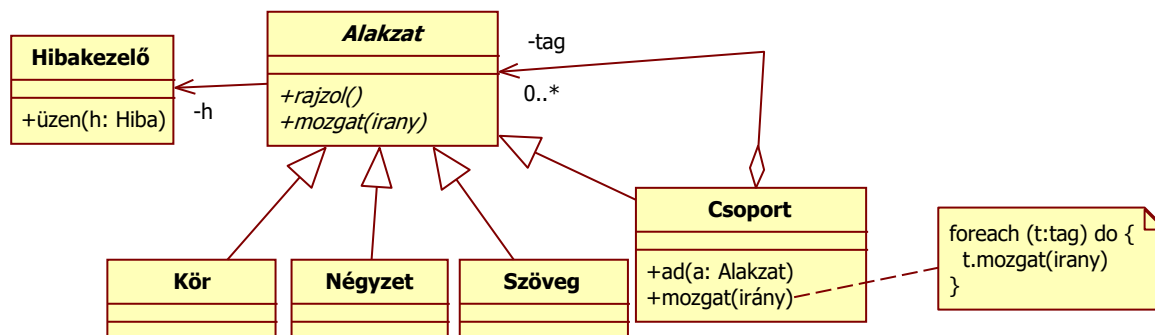


Izidor kiveszi a könyvgyűjteményéből kedvenc könyvét, ajánlást ír bele, majd odaadja Gerzsonnak. Rajzoljon UML2 kommunikációs diagramot !

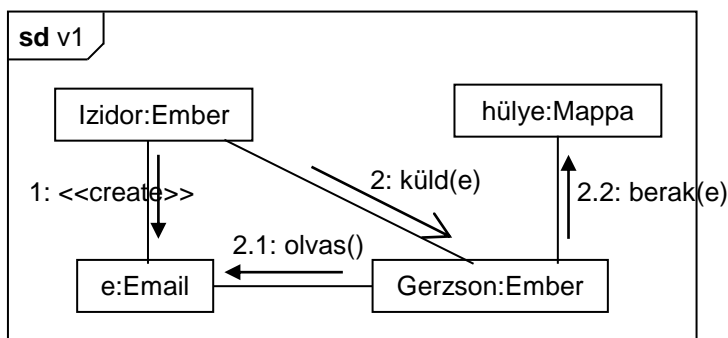


Az alábbi UML osztálydiagram segítségével, a szöveg alapján rajzoljon UML 2.0 kommunikációs diagramot!

Egy rajzszerkesztő program modellje az ábrán látható. A felhasználó korábban készített egy kört, egy szöveget és egy négyzetet. A kört és a szöveget csoportba foglalta, majd az így kapott csoportot és a négyzetet újabb csoportba tette. (Az ezzel kapcsolatos metódushívásokat ne jelölje az ábrán!) Jelölje viszont a kommunikációs diagramon, ahogy a felhasználó meghívja az utolsó csoporton a mozgat() tagfüggvényt! A szöveg, ha a kép szélére kerül, értesíti a hibakezelő objektumot. Alkalmazzon hierarchikus számozást!

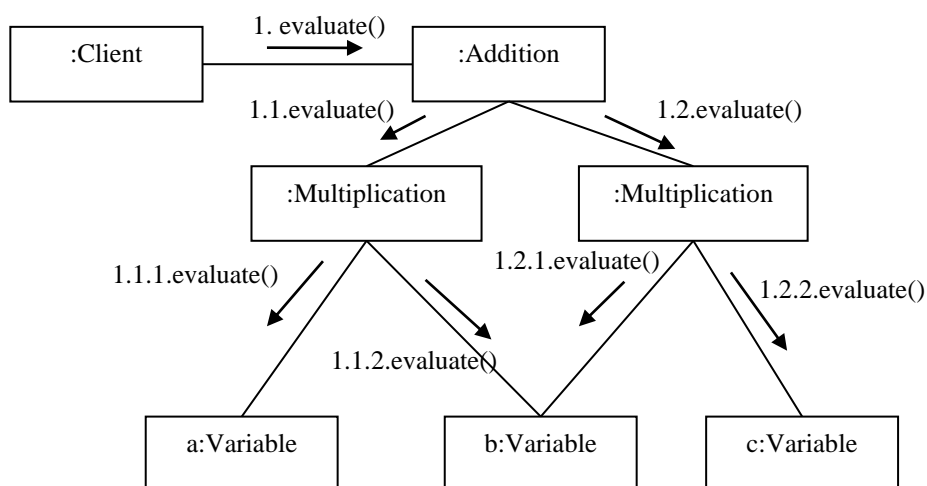
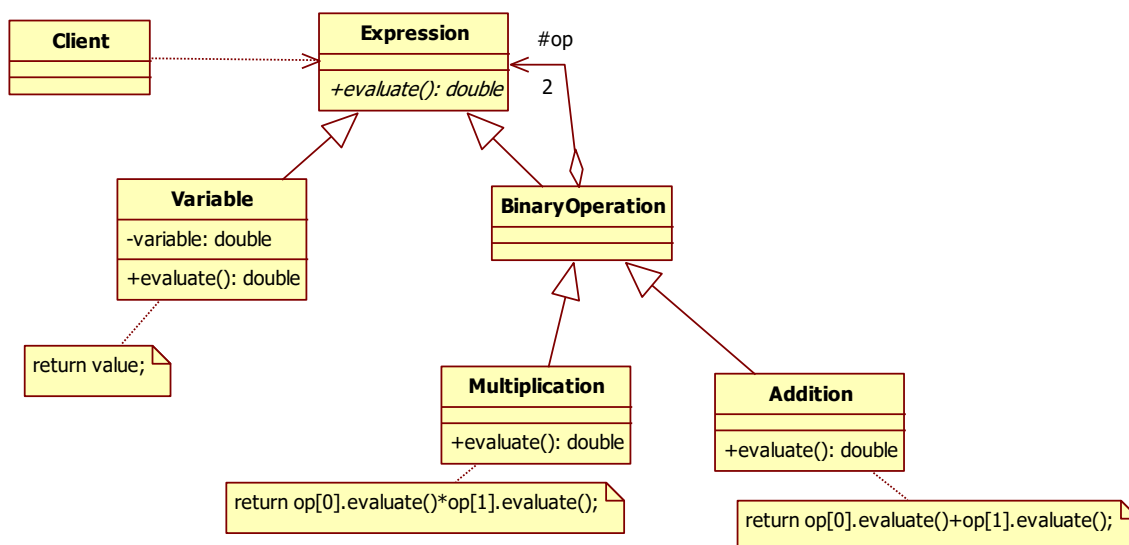


Izidor ír egy e-mailt, amit elküld Gerzsonnak. Gerzson a levelet elolvassa, majd beteszi a „hülye” mappába. Rajzoljon UML2 kommunikációs diagramot ! Alkalmazzon hierarchikus számozást !

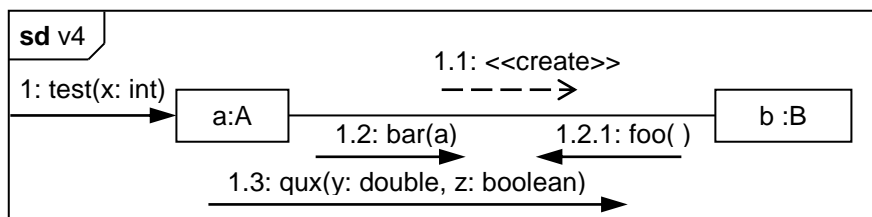




Az alábbi diagramon szereplő osztályokat használjuk algebrai kifejezések modellezésére. Készítsen UML2 kommunikációs diagramot arra az esetre, ha egy kliens kiértékeli az  $(a * b) + (b * c)$  kifejezést! A rendszer nem készít felesleges objektumokat. Az üzeneteket hierarchikusan számozza!

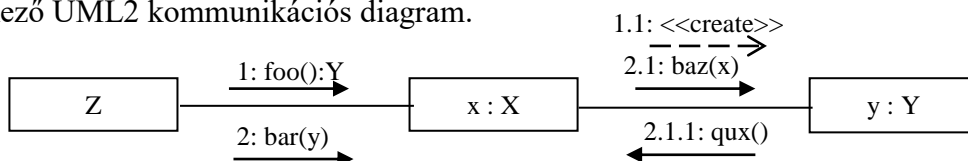


Adja meg, hogy az alábbi UML2 kommunikációs diagramon leírt viselkedés megvalósításához minimálisan milyen példány attribútumokat szükséges definiálni az osztályokban!

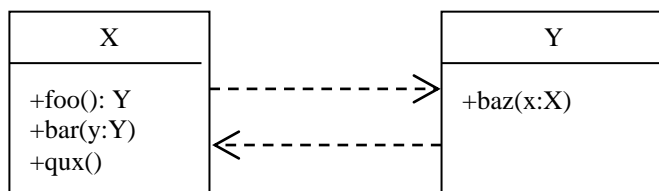


osztály	nem kell attribútum	attribútum típusa
A	<input type="checkbox"/>	
B	<input type="checkbox"/>	

Adott a következő UML2 kommunikációs diagram.



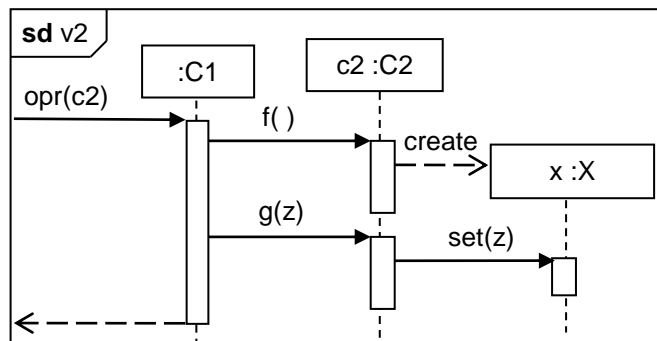
Feltételezve, hogy a fenti diagramon szereplő objektumok osztályainak nincsenek – a diagramból nem kiolvasható – további metódusai, közöttük nincs más egyéb kapcsolat (pl. öröklés), az alábbi ábrát korrekt UML2 osztálydiagrammá alakítva ábrázolja az osztályokat a metódusok szignatúráival együtt, valamint a két osztály közötti kapcsolatot !



2. Tételezze fel, hogy az alábbi UML2 szekvenciadiagramon szereplő objektumok osztályai között nincs más egyéb – a diagramból nem kiolvasható – kapcsolat (pl. öröklés) ! Mi a kapcsolat C2 és X között ? Válaszát egy, a magyar nyelv szabályainak megfelelő, olvasható MONDATtal indokolja ! Indoklás nélkül a választása nem érvényes.

- ☐ példányosítás (instantiation)
- ☒ asszociáció (association)
- ☐ kollaboráció (collaboration)
- ☐ függőség (dependency) C2 függ X-től
- ☐ függőség (dependency) X függ C2-től
- ☐ interakció (interaction)
- ☐ implementálás (implementation)

Indoklás:

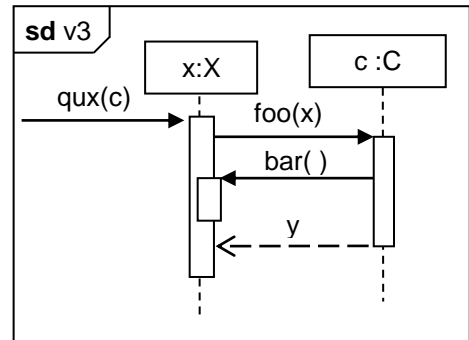
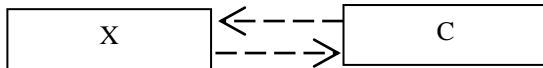


C2-nak "emlékeznie kell" X-re .....

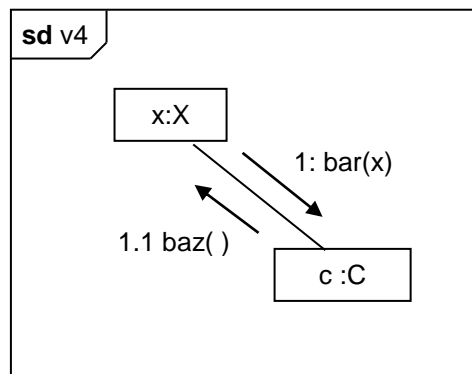
Tételezze fel, hogy az alábbi (v3 nevű) UML2 szekvenciadiagramon szereplő objektumok osztályai között nincs más egyéb – a diagramból nem kiolvasható – kapcsolat (pl. öröklés) ! Mi a kapcsolat X és C között ?

- ☐ implementálás (implementation)
- ☐ kollaboráció (collaboration)
- ☐ függőség (dependency) C függ X-től
- ☐ függőség (dependency) X függ C-től
- ☐ asszociáció (association)
- ☐ példányosítás (instantiation)
- ☐ interakció (interaction)

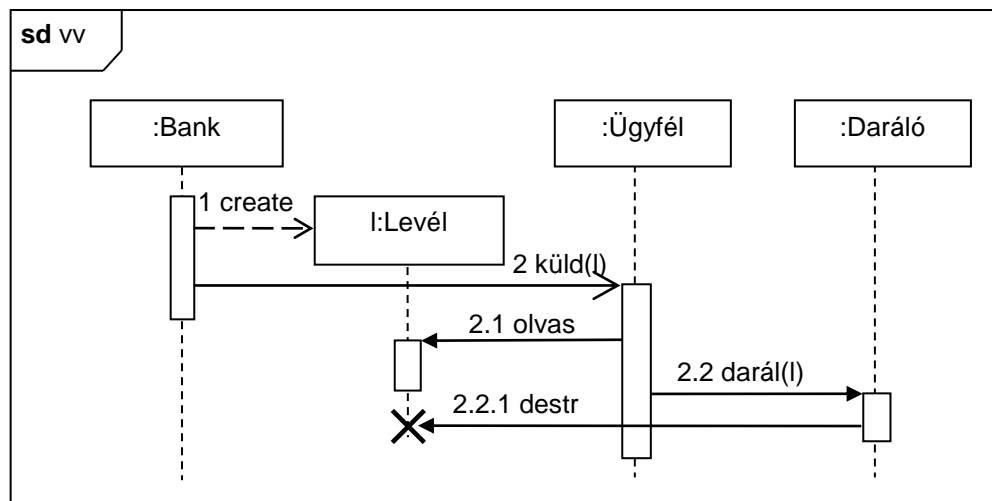
Rajzolja be választását az alábbi osztálydiagramba !



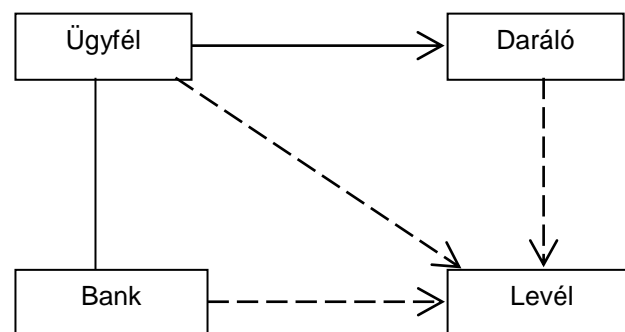
Rajzolja le a fenti (v3 nevű) szekvenciadiagramnak megfelelő UML2 kommunikációs diagramot !



Rajzoljon UML2 **szekvenciadiagramot** ! Az üzeneteket hierarchikus számozással lássa el !  
 Az InterCredit Bank felszólító levelet ír, amelyet elküld egyik ügyfelének, Gerzsonnak, akinek hiteltartozása van. Gerzson a levelet elolvassa, majd a darálón ledarálja.



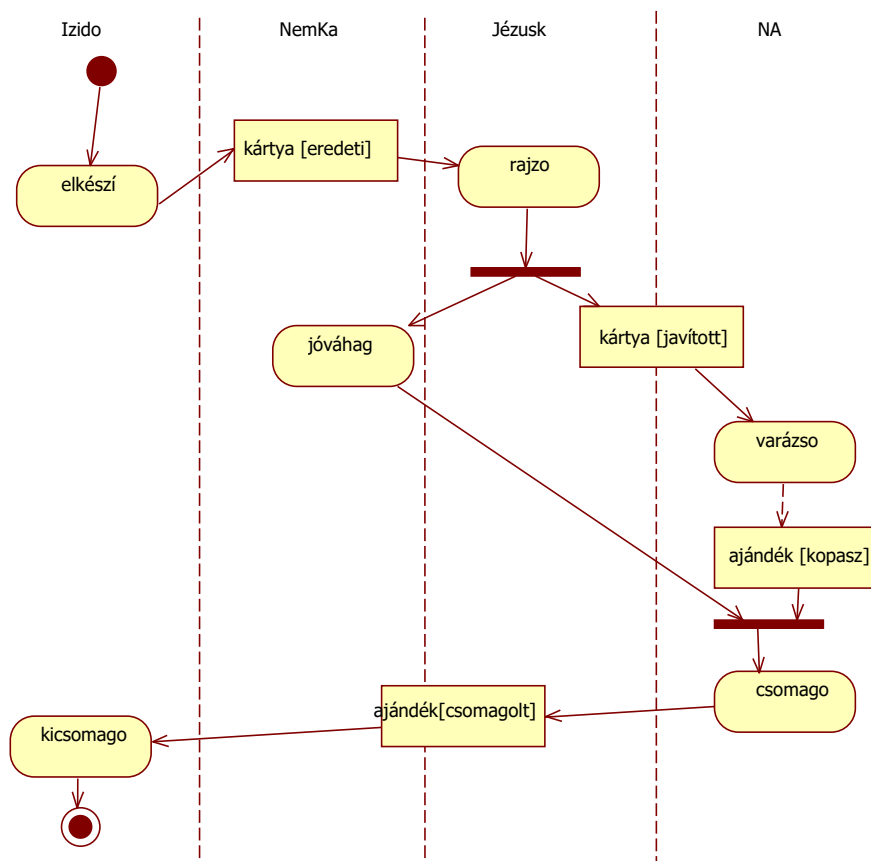
Tételezze fel, hogy a fenti történetben szereplő objektumok osztályai között nincs más egyéb – a történetből nem kiolvasható – kapcsolat (pl. öröklés) ! Az alábbi UML2 osztálydiagramba rajzolja be az osztályok közötti kapcsolatokat !



## 7 Aktivitás diagram

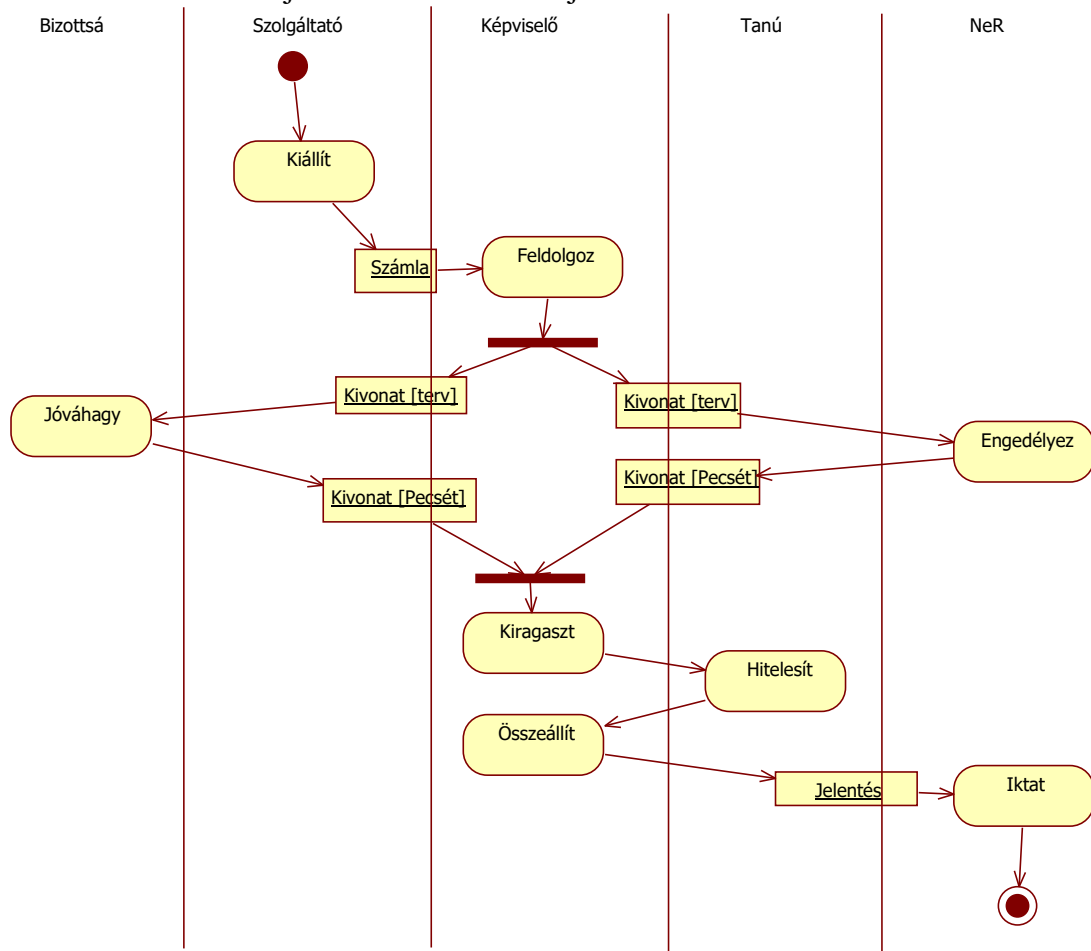
Az alábbi leírás alapján rajzoljon UML aktivitás-diagramot az objektum áramlást is berajzolva!

Izidor szeretne kapni egy életnagyságú legó stadiont, ezért készít egy fotóval ellátott kívánságkártyát, amit eljuttat a Jézuskának. Jézuska a kártyára ötkarikás szívet rajzol, majd ismeretlen sorrendben megküldi a kívánságot a Nemzeti Karácsony Partnerhez (NemKaP), a kártyát pedig a Nemzeti Ajándékgyártó Hivatalba (NAB). A NAB-ban elővarázsolják a kért ajándékot. Időközben a NemKaP jóváhagyó döntést hoz, és erről értesíti a NAB-ot. Amikor a NAB-hoz megérkezik a jóváhagyás, akkor becsomagolják az ajándékot és elküldik a csomagot Izidornak. Izidor kicsomagolja az ajándékot.

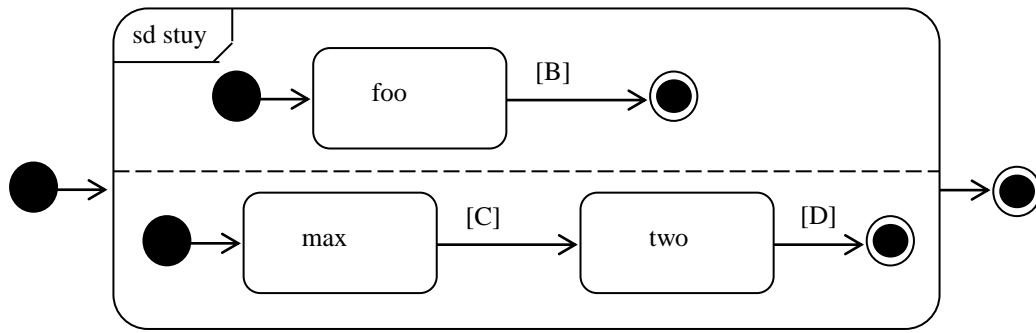
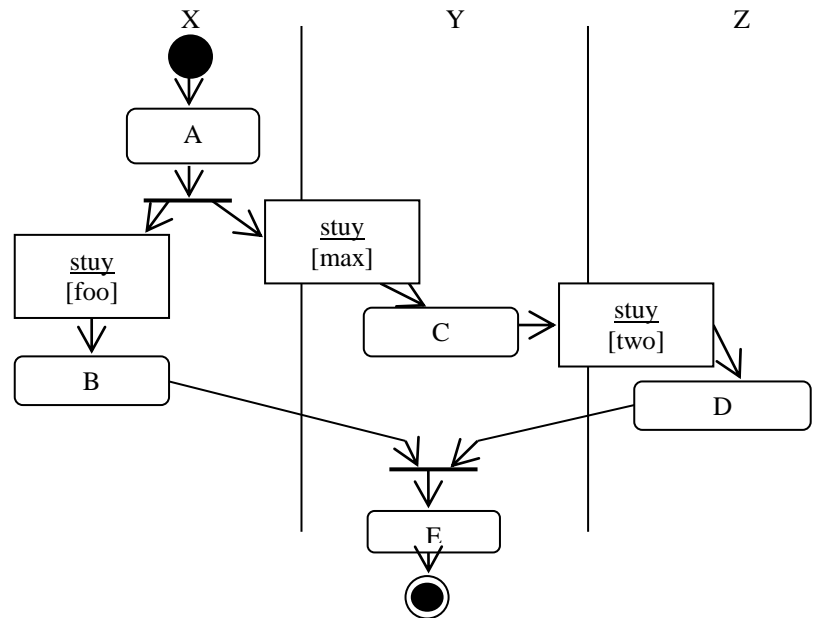


Készítsen object flow-val kiegészített UML2 aktivitás diagramot (activity diagram) az alábbi történet alapján!

A szolgáltató számlát állít ki a társasház részére, amit a közös képviselő kap meg. A számlát a képviselő feldolgozza, majd a számlakivonatot (terv állapotában) megküldi a számvevő bizottságnak és a Nemzeti Rezsicsökkentési Osztálynak (NeRO). Ezek jóváhagyják (pecsételik) a saját példányukat, és így visszaküldik a képviselőnek, aki kiragasztja a falújságra. Ezt egy tanú hitelesíti. Végül a képviselő jelentést készít, amit elküld a NeRO-nak. A jelentést a NeRO iktatja.

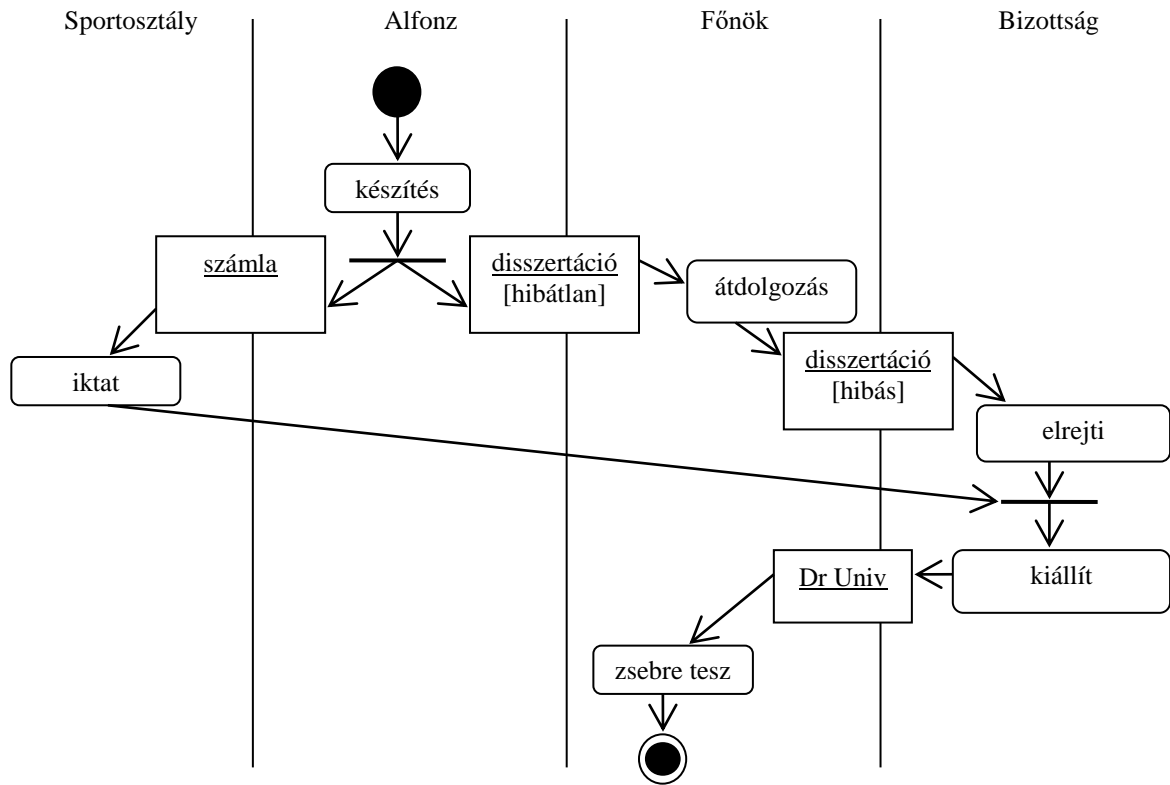


Adott a mellékelt – object flow-val  
kiegészített – aktivitás-diagram (activity  
diagram) ! Rajzolja meg azon objektumok  
UML2 state-chartját, amelyeknek az ábra  
alapján több állapota is van!



Készítsen UML 2 aktivitás-diagramot (activity diagram) az alábbi leírás alapján! Jelölje az action-object flow-t is! Használja a kövéren szedett kifejezéseket!

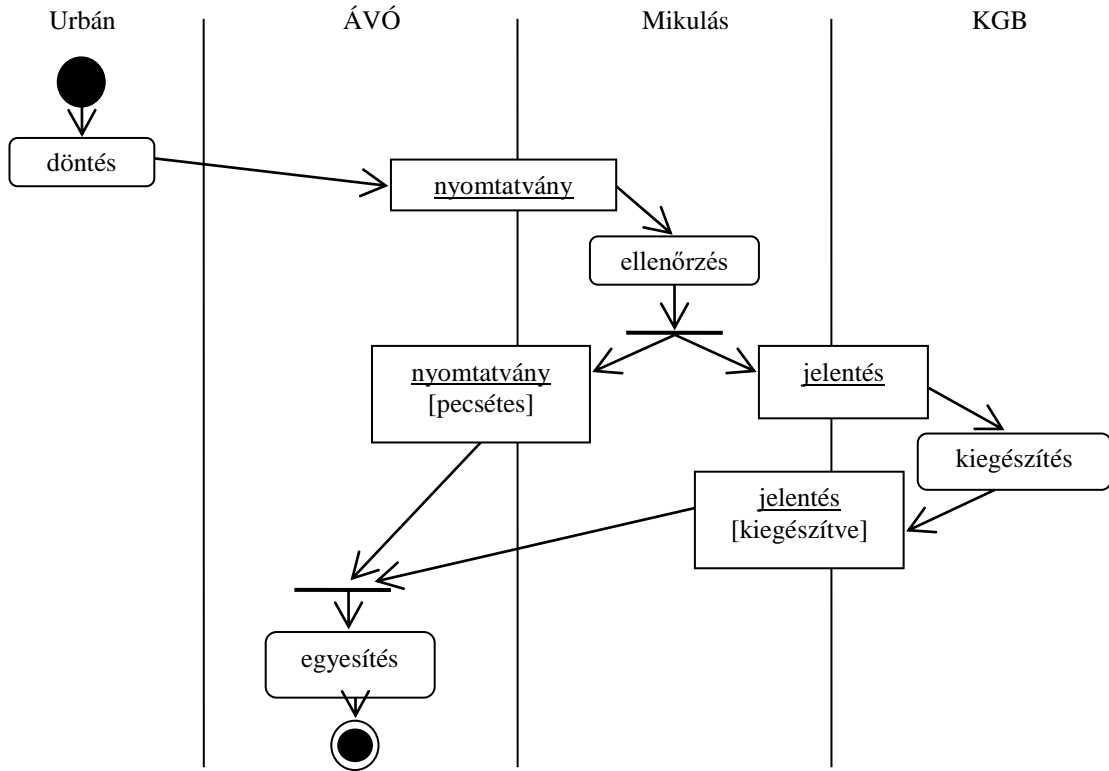
Senki **Alfonz** opponens **disszertációt készít**. A hibátlan disszertációt elküldi a **főnökének átdolgozásra**, és ezzel párhuzamosan **számlát** ad a **sportosztálynak**. A főnök az átdolgozás során néhány helyesírási hibát tesz a disszertációba, és átadja a **bizottságnak**, ahol **elrejtik**. A sportosztály **iktatja** a számlát, és ezt jelzi a bizottság felé. A bizottság, miután végzett a rejtéssel és megkapta a sportosztály jelzését, **kiállítja** a doktori **bizonyítványt**, amit elküld a főnöknek, aki a címet legjobb tudása szerint **zsebre teszi**. A folyamatnak itt vége szakad.



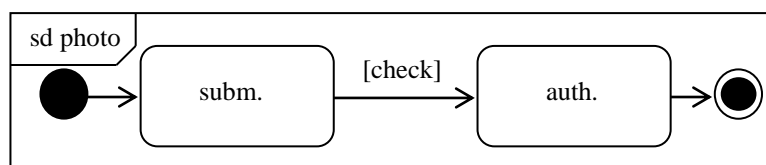
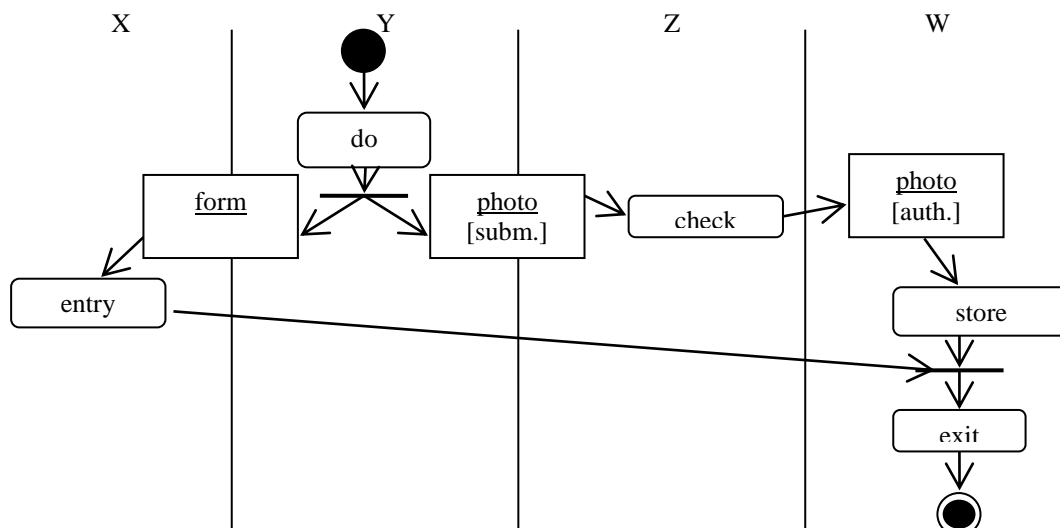


Készítsen UML 2 aktivitás-diagramot (activity diagram) az alábbi leírás alapján! Jelölje az action-object flow-t is! Használja a kövéren szedett kifejezéseket!

Virgonc **Urbán** úgy **döntött**, hogy maradni kíván a MalacNyúzó Pribékek (MaNyüP) társaságában. Emiatt, az előírásoknak megfelelően, a **Mikulás**nak átad egy formanyomtatványt, amin szerepel a DNS mintája, az ujjlenyomata és az íriszképe. A Mikulás a nyomtatványt **ellenőrzi**, majd **lepecsételve** elküldi az Ákombákom Vizslató Orrszarvúnak (**ÁVÓ**), megőrzésre. A Mikulás az ellenőrzés után egy **jelentést** is küld a Központi Gépészeti Bizottságnak (**KGB**), amely a jelentést **kiegészíti** az ítélettel, és továbbítja az Orrszarvúnak, aki a két iratot **egyesíti**.



Legyen adott az alábbi – object flow-val kiegészített – aktivitás-diagram (activity diagram) ! Rajzolja meg azon objektumok UML2 state-chartját, amelyeknek az ábra alapján több állapota is van!

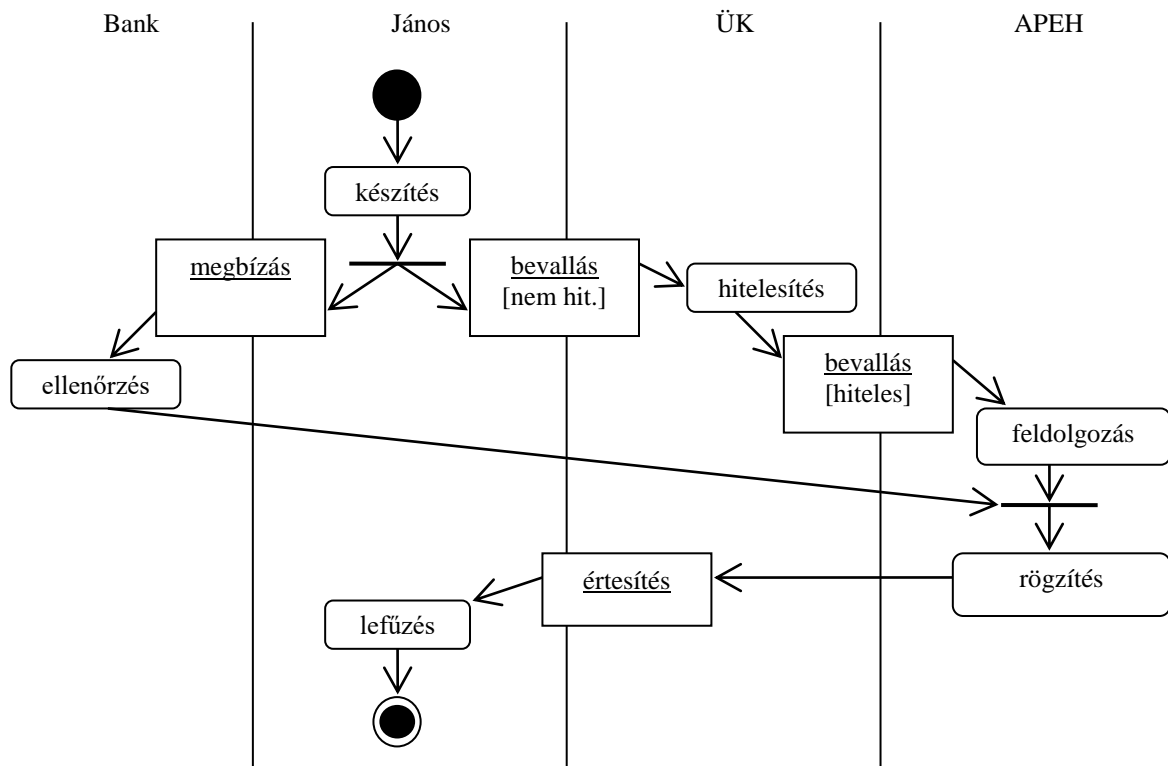


Az UML2 Activity diagram egy másik UML2 diagram speciális esetének tekinthető. Melyik ez a diagram? Hasonlítsa össze a két diagramot!

Activity diagram	Állapot..... diagram
state – tevékenységet hajt végre	state (eseményre vár)
transition – a tevékenység befejezésekor (belső) automatikusan	transition (külső esemény hatására)

Készítsen UML 2 aktivitás-diagramot (activity diagram) az alábbi leírás alapján! Jelölje az action-object flow-t is! Használja a kövéren szedett kifejezéseket!

Regenkurt **János** vállalkozó adóbevallást **készít**. A **bevallást** feltölti az ügyfélkapura (**ÜK**), és ezzel párhuzamosan a **bankjánál** átutalási **megbízást** ad az adóhátralék befizetésére. Az ügyfélkapu **hitelesíti** az adóbevallást, és továbbküldi az **APEH**-nek, ahol **feldolgozzák**. A bank **ellenőrzi** az átutalási megbízást, majd a pénzt átutalja az APEH-nek. Amikor az APEH feldolgozta a bevallást és megkapta az átutalást, akkor **rögzíti** az állapotot, és **értesítést** küld Jánosnak, hogy minden rendben. Ezt az értesítést János **lefűzi**. Ezzel az adóbevallás véget ér.

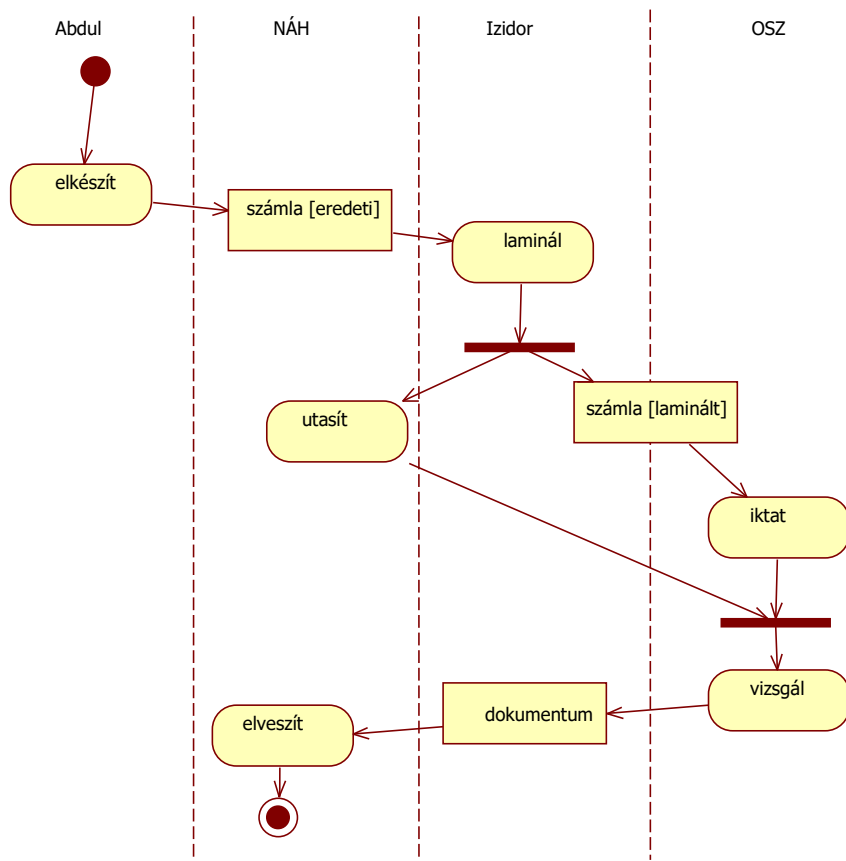


Jelölje meg, hogy az UML2 aktivitás diagramon megjelenő nyilazott élek mit jelentenek !

- ☐ csak vezérlési folyamat
- ☐ csak adatfolyamat
- ☐ vezérlési folyamat vagy adatfolyamat
- ☒ vezérlési folyamat vagy adatfolyamat vagy mindkettőt

Az alábbi leírás alapján rajzoljon – objektum-áramlással (object flow) kiegészített – UML2 aktivitás-diagramot!

Abdul számlát készít a nemzeti géroszról Izidornak. Izidor a számlát laminálja. Később a Nemzeti Áfacsaláskísérletmegelőzési Hatóságot értesíti az ügyletről, a számlát az Országos Számlabegyűjtőbe küldi, ahol iktatják. A Hatóság utasítja a Számlabegyűjtőt, hogy a számlát vizsgálják meg. A Számlabegyűjtő megvizsgálja a számlát (ha 18%-os és van rajta zsírnyom, akkor nem volt szalvéta a géroszosnál, így nem jogosult a 18% kulcsra, ez áfacsalás, stb). A vizsgálat eredményét (54 oldalas dokumentum) elküldik a Hatóságnak, ahol elveszítik.

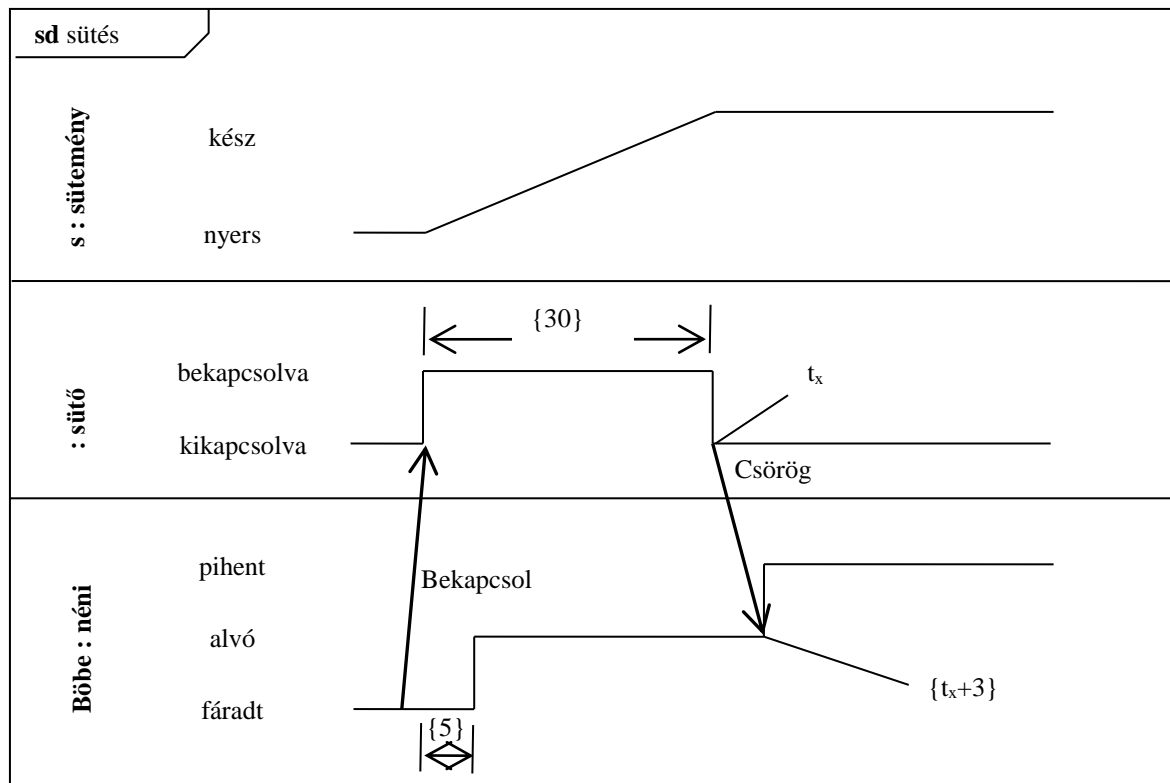


## 8 Időzítési diagram

Az alábbi történet alapján rajzoljon UML 2 időzítési diagramot (timing diagram)!

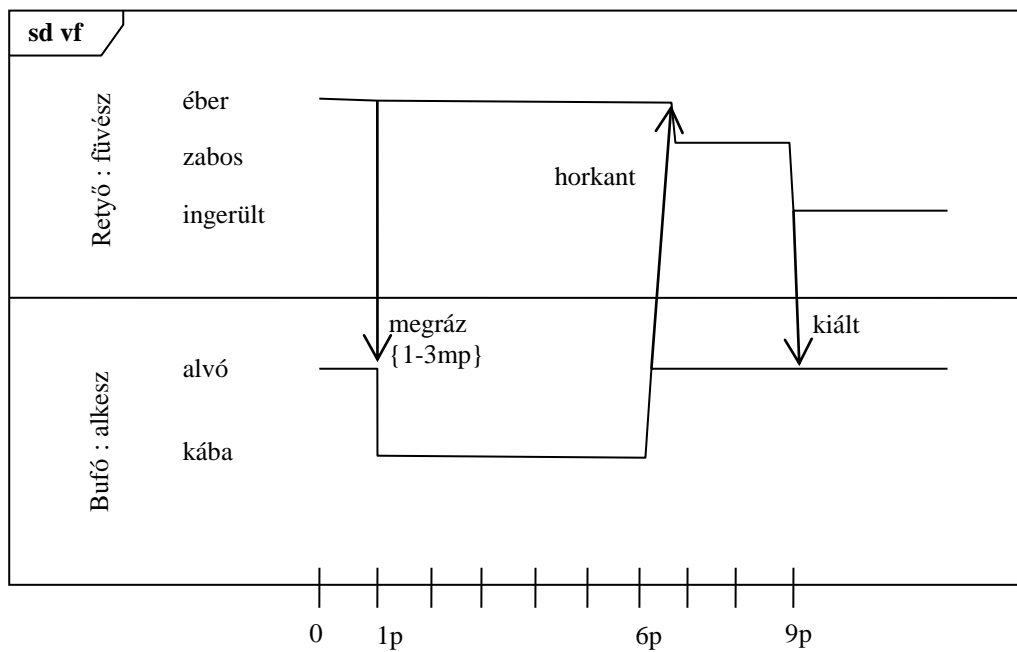
Böbe néni fáradt, de másnapra süteményt kell sütnie. A nyers süteményt beteszi a sütőbe, amit aztán bekapcsol. Öt perc múlva Böbe néni elalszik. A sütő fél óra elteltével kikapcsol és csörög, amire Böbe néni 3 perc múlva kipihenten felébred, és kiveszi a kész süteményt.

Böbe néni diszkrét állapotai: fáradt, alvó, pihent. A sütő diszkrét állapotai: ki, be. A sütemény folytonosan változik a nyers és a kész között.



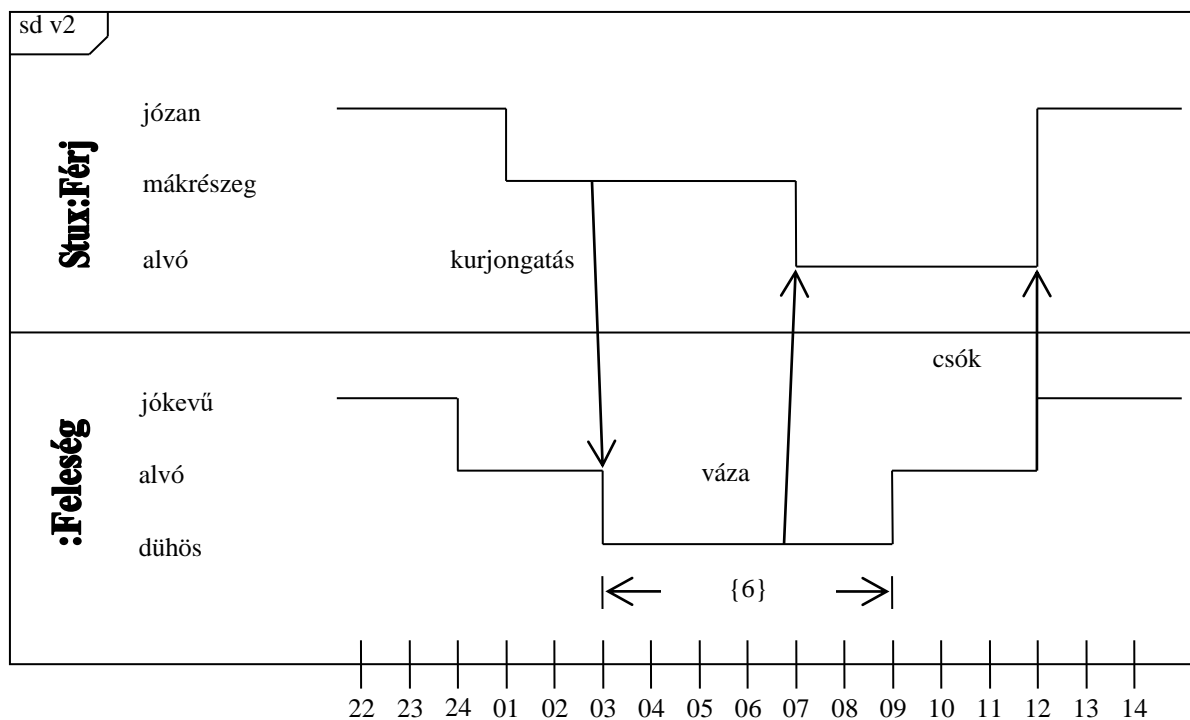
Az alábbi történet alapján rajzoljon UML 2.0 időzítési diagramot (timing diagram)!

Retyő, a fűvész és Bufó, az alkesz barátok. A történet kezdetekor Retyő éber, Bufó alvó állapotban van. 1 perc után Retyő (1-3 másodpercig) megrázza Bufót, mire az kába állapotba kerül. Bufó 5 perc elteltével visszaalszik, és horkant, amire Retyő zabos lesz. 3 perc múlva Retyő ingerültté válik és Bufóra rákiált, aki erre nem reagál.

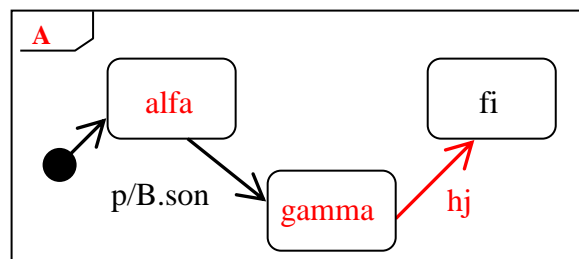
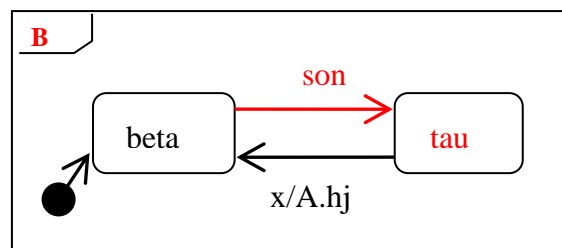
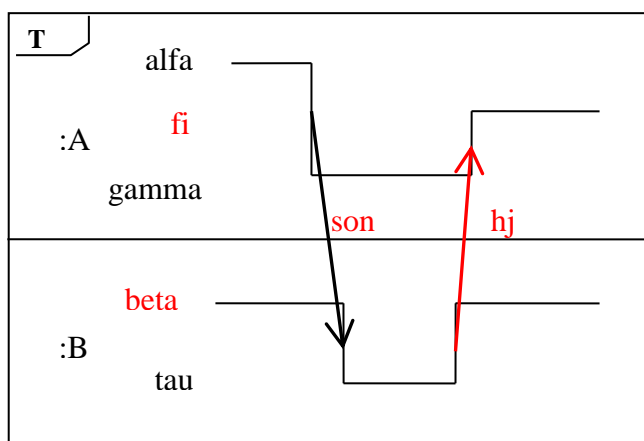


Az alábbi történet alapján rajzoljon UML 2.0 időzítési diagramot (timing diagram)!

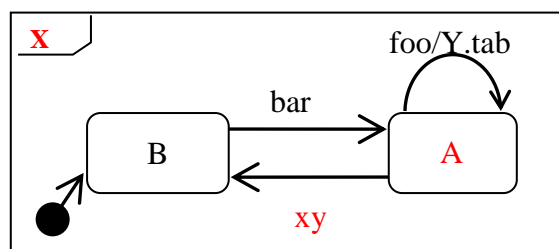
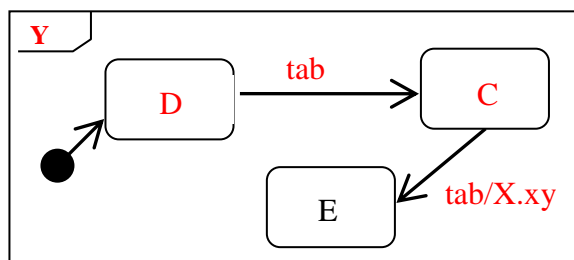
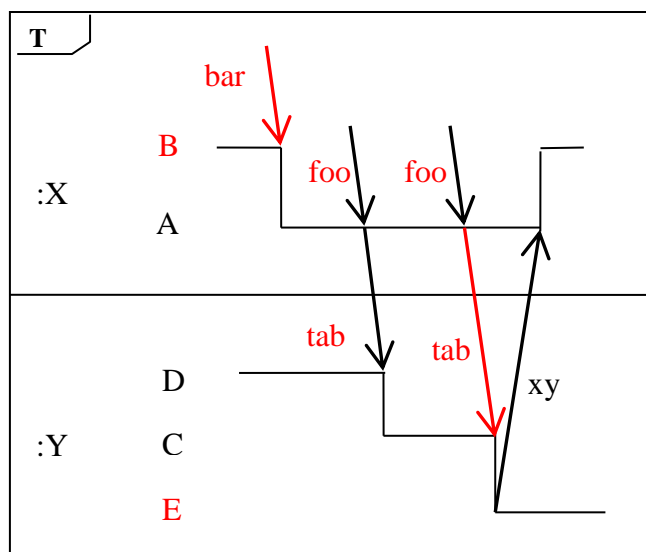
Stux este 10-kor józanul ment el otthonról, egyedül hagyva jókedvű feleségét, aki 12-kor elaludt. Stux hajnali 1-re lett mákrészeg, és hajnali 3-ra ért haza. Ekkor a kurjongatásra felesége dühösen ébred, és hat órán át dühös is marad. 7-kor felesége egy vázát vág hozzá, amitől Stux elalszik. Miután az asszony kidühöngte magát, elalszik. Mikor délben jókedvűen felébred, megcsókolja férjét, aki józanul ébred.



Az alábbi ábrákról hiányzik néhány elem (szöveg, vonal, nyíl stb.) Rajzolja be a hiányzó elemeket és feliratokat, úgy, hogy az ábrák helyesek és összefüggőek legyenek!

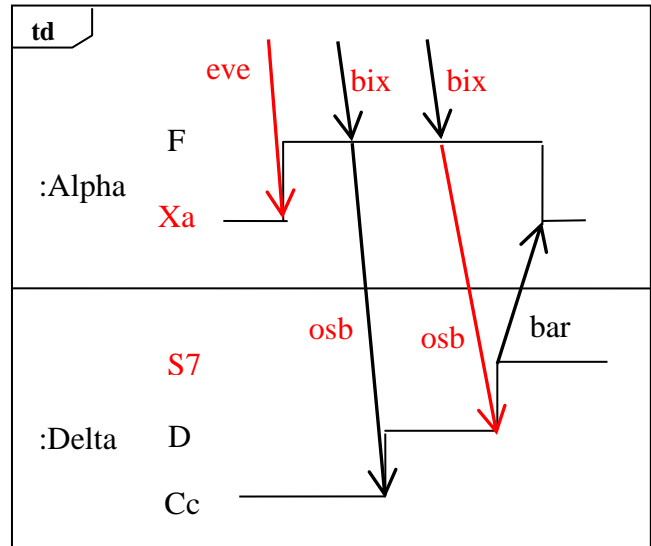
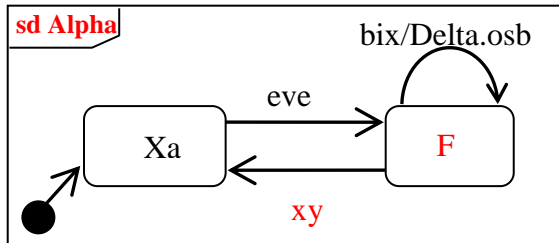
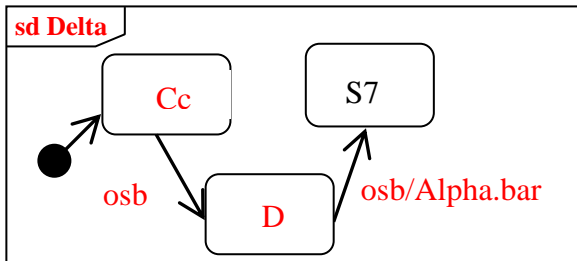


Az alábbi ábrákról hiányzik néhány elem (szöveg, vonal, nyíl stb.) Rajzolja be a hiányzó elemeket és feliratokat, úgy, hogy az ábrák helyesek és összefüggőek legyenek!



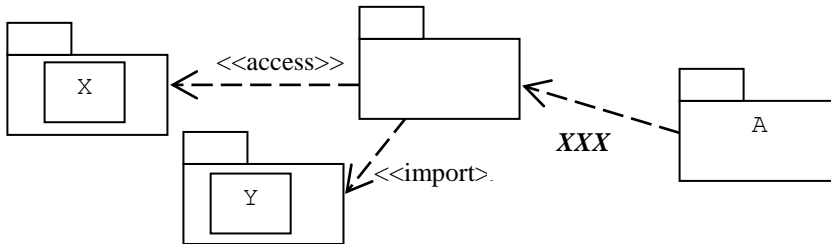


Az alábbi ábrákról hiányzik néhány elem (szöveg, vonal, nyíl stb.) Rajzolja be a hiányzó elemeket és feliratokat, úgy, hogy az ábrák helyesek és összefüggőek legyenek!



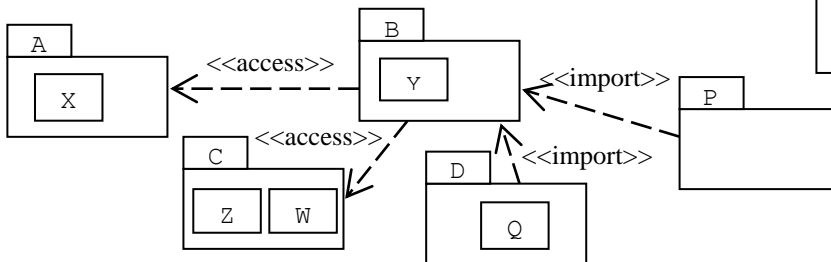
## 9 Package diagram

A táblázatba írja be, hogy az ábrán látható UML2 csomagdiagramban szereplő A csomagban milyen elemeket látunk – annak függvényében, hogy mi az **XXX** jelű sztereotípia !



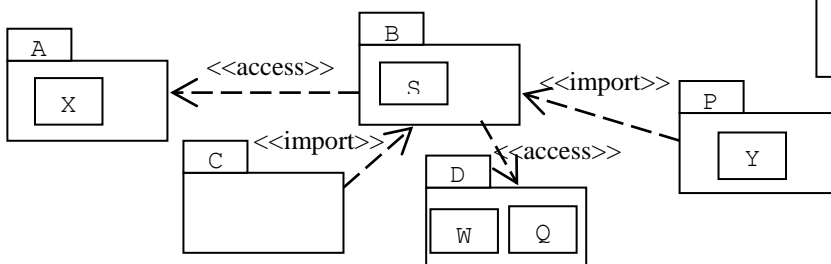
XXX	elemek
<<access>>	Y
<<import>>	Y

Adja meg a P csomagban látható elemeket !



Y .....

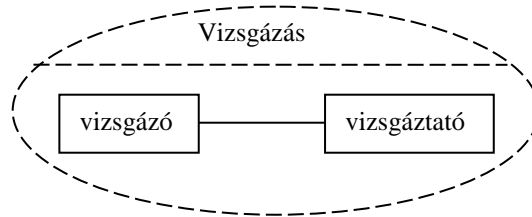
Adja meg a C csomagban látható elemeket !



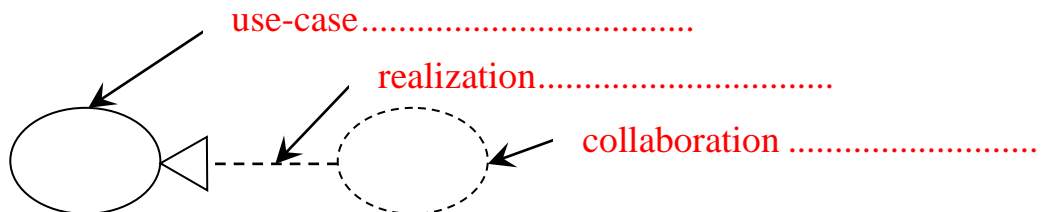
S .....

## 10 Kollaboráció

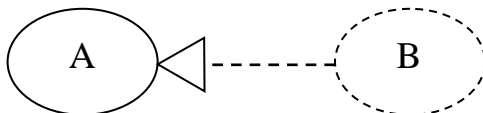
A **Vizsgázás** funkció a **vizsgázó** és a **vizsgáztató** szerepeket megvalósító objektumok együttműködéseként valósul meg. Rajzoljon UML2 kollaborációt (collaboration) a szerepek feltüntetésével !



Adja meg, hogy a jelölt elem melyik UML meta-modell elem példánya !



Mi **A** és **B** az alábbi UML diagramon ?



A		B	
<input type="checkbox"/>	operáció	<input checked="" type="checkbox"/>	kollaboráció
<input type="checkbox"/>	állapot	<input type="checkbox"/>	metódus
<input checked="" type="checkbox"/>	use-case	<input type="checkbox"/>	beágyazott állapot
<input type="checkbox"/>	processz	<input type="checkbox"/>	feltételes use-case

## 11 Rational Unified Process

A RUP (Rational Unified Process) egyik munkafolyamatában (workflow) szerződés (contract) készítését javasolja illetve írja elő. Melyik munkafolyamatban esedékes szerződés készítése ? Kik között kell szerződést készíteni ? Milyen fontosabb pontjai vannak a szerződésnek ?

Munkafolyamat (workflow): **analízis** .....

Szerződő felek: **operációk és az operációk felhasználói** .....

Szerződés fontosabb pontjai: **Responsibilities, Pre-conditions, Post-condition, Types, Crossrefs, .....**

**Exceptions, Output** .....

Az alábbi táblázatban adja meg a Rational Unified Process (RUP) fő munkafolyamatait és nevezze meg a hozzá tartozó nézetet.

RUP fő munkafolyamatok/ RUP Process Workflows	Nézetek/Views
<b>Requirement</b>	<b>Use-case</b>
<b>Analysis</b>	<b>Logical</b>
<b>Design</b>	<b>Component</b>
<b>Implementation</b>	<b>Process</b>
<b>Deployment</b>	<b>Deployment</b>

A következő táblázatban adja meg a Rational Unified Process (RUP) felsorolt munkafolyamataihoz tartozó nézetet.

RUP fő munkafolyamatok/ RUP Process Workflows	Nézetek/Views
<b>Analysis</b>	<b>Logical</b>
<b>Design</b>	<b>Component</b>
<b>Implementation</b>	<b>Process</b>

Mi a Rational Unified Process (RUP) életciklus modelljének utolsó fázisa? Milyen tevékenységek tartoznak ebbe a fázisba?

utolsó fázis: **.transition**.....

tevékenységek: **manufacturing, delivering, training** .....

A RUP tervezési munkafolyamatában (workflow) az architektúra tervezésekor milyen döntéseket hozunk ?

- Organize the system into packages (subsystems)
- Identifying concurrency
- Allocating packages to processors
- Storage and Persistence
- Handling global resources
- Choosing software control
- Handling boundary conditions

---

Sorolja fel a Rational Unified Process (RUP) élekciklus modelljében szereplő „támogató munkafolyamatokat” (supporting workflows)!

konfigurációs menedzsment  
menedzsment  
környezet

---

A RUP-ban (Rational Unified Process) alkalmazott használati esetek (use-case-ek) különböző szempontok szerint csoportosíthatóak. Jellemezze a **lényeges** (essential) és a **valóságos** (real) use case-eket! Miben különböznek egymástól?

lényeges - eszköz, implementáció független  
valóságos - implementációs (ablakok, mezők, triggererek)  
a technológiai függőségében

---

A RUP a Use Case (használati eset) modellek milyen formáit alkalmazza ?

high level or expanded.....  
essential (free of technology), real.....  
primary, secondary, optional.....

Milyen UML2 metamodel elem az actor és a use case közötti “vonal” ?

asszociáció .....

---

A RUP (Rational Unified Process) use-case vezérelt.

Miben különbözik a magas szintű (high level) és a kiterjesztett (expanded) use-case ?

A leírás részletezettségében

(magas szintű - név, aktorok, cél, attekintés, referencia

kiterjesztett - eseményfolyam, elő- és utófeltételek, kiterjesztési pontok, relációk, aktivitás és use-case diagram referenciák)

Miben különbözik a lényeges (essential) és a valóságos (real) use-case ?

(A technológia függőségében

lényeges - eszköz, implementáció független

valóságos - implementáció (ablakok, mezők, triggererek)

---

A Rational Unified Process (RUP) tervezési szakaszában milyen típusú use-case-eket készítünk ? Mi ennek a típusú use-case-nek a fő jellemzője ?

**valóságos (real)**  
**a felhasználói felület elemeire hivatkozik**

---

A Rational Unified Process (RUP) követelmény szakaszában milyen típusú use-case-eket készítünk ? Mi ennek a típusú use-case-nek a fő jellemzője ?

**high level, (some expanded), essential**  
**technológia- és implementáció-független**

---

A RUP (Rational Unified Process) egyik munkafolyamatában (workflow) szerződés (contract) készítését javasolja illetve írja elő. Melyik munkafolyamatban esedékes szerződés készítése ? Kik között kell szerződést készíteni ? Milyen fontosabb pontjai vannak a szerződésnek ?

**Workflow: analízis**                      **Kik: az operációk és az operációk felhasználói között (rendszer szinten).**  
**Főbb pontok: Responsibilities, Pre-conditions, Post-condition, Types, Crossrefs, Exceptions, Output**

---

smertesse a RUP-ban a fogalmi model kialakításakor használt "térképész elv"-et !

- **use the existing names in the territory**
- **exclude irrelevant features**
- **do not add things that are not there**

---

A Rational Unified Process (RUP) követelménykezelő (Requirement) munkafolyamatában melyik UML modellt, és melyik két diagramtechnikát alkalmazzuk ? Hibás válasz pontszámcsökkentő.

**modell: .use-case .....**

**diagramok: use-case diagram, szekvenciadiagram .....**

---

## 12 Egyéb feladatok

### 12.1 Kollekciónak

Az UML2-ben definiált **Bag** gyűjteménynek (kollekciónak) adja meg a tulajdonságait!

igen	nem	nem jellemző	tulajdonság
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	delegált (delegated)
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	minősített (qualified)
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	rendezett (ordered)
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	egyedi (unique)

Az UML2-ben a gyűjteményeknek (kollekcióknak) két fontos tulajdonsága van: rendezettség (ordered) és egyediség (unique). Írja be a táblázatba az UML2 kollekciók nevét !

rendezett	egyedi	UML2 kollekció neve
igen	igen	<b>Ordered Set</b>
igen	nem	<b>Sequence</b>
nem	nem	<b>Bag</b>
nem	igen	<b>Set</b>

Az UML2-ben definiált **Sequence** gyűjteménynek (kollekciónak) adja meg a tulajdonságait!

igen	nem	nem jellemző	tulajdonság
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	egyedi (unique)
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	minősített (qualified)
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	rendezett (ordered)
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	delegált (delegated)

Nevezze meg az UML2-ben definiált gyűjteményeknek (kollekcióknak) a – tipizálásra is alkalmazott – két alapvető jellemzőjét !

1. **rendezettség (ordered)** ..... 2. **egyediség (unique)** .....

## 12.2 Konkurencia

Legyen egy  $X$  osztályunk,  $aaa()$  és  $bbb()$  metódusokkal jellemezve. Egy kliens meghívja az  $aaa()$  metódust. Az  $aaa()$  futása közben egy másik kliens meghívja a  $bbb()$  metódust. Az alábbi táblázatba írja be, hogy a különböző UML2 szemantikák esetében mi a követett eljárás (policy)!

szemantika neve	eljárás (policy)
sorrendi (Sequ)	kizárt – nem fordulhat elő
őrzött (Guarded)	befejezi $aaa$ -t, majd elkezd $bbb$ -t
konkurrens (Conc)	félbeszakítja $aaa$ -t és elkezd $bbb$ -t

Miben különbözik, ha a másik kliens is az  $aaa()$  metódust hívja?

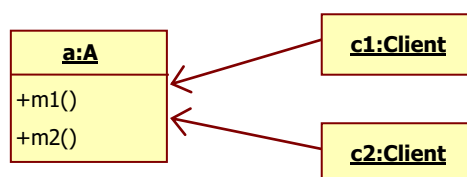
**semmiben** .....

Az alábbi táblázat első oszlopába írja be az UML-ben definiált konkurencia szemantikák nevét !

Egy embernek több telefonja van. Miközben az egyiket beszél, egy másikon is hívás érkezik. A táblázatba írja be, hogy a különböző szemantikák szerint a konkurrens hívás előfordulhat-e, és mi történik az éppen zajló beszélgetéssel és az új hívással !

szemantika neve	előfordulhat-e?	mi történik ?
sorrendi (Sequ)	<input type="checkbox"/>	folytatja
őrzött (Guarded)	<input type="checkbox"/>	folytatja - ha befejezte, felveszi
konkurrens (Conc)	<input type="checkbox"/>	félbeszakítja - azonnal felveszi
	<input type="checkbox"/>	

Adottak az alábbi szerkezeti diagram által definiált objektumok.  $c1$  meghívja az  $m1()$  metódust. Az  $m1()$  futása közben  $c2$  meghívja az  $m2()$  metódust. Az alábbi táblázatba írja be, hogy a különböző UML2 szemantikák esetében mi a követett eljárás (policy)!



szemantika neve	eljárás (policy)
sorrendi (Sequ)	kizárt – nem fordulhat elő
őrzött (Guarded)	befejezi $m1$ -t, majd elkezd $m2$ -t
konkurrens (Conc)	félbeszakítja $m1$ -t és elkezd $m2$ -t



---

Hogyan értelmezzük az UML2-ben a szekvenciális konkurenciát

*callers must coordinate outside the object so that only one flow is in the object at a time.*

Sorolja fel az UML által definiált egyéb konkurencia szemantikákat !

*guarded, concurrent*

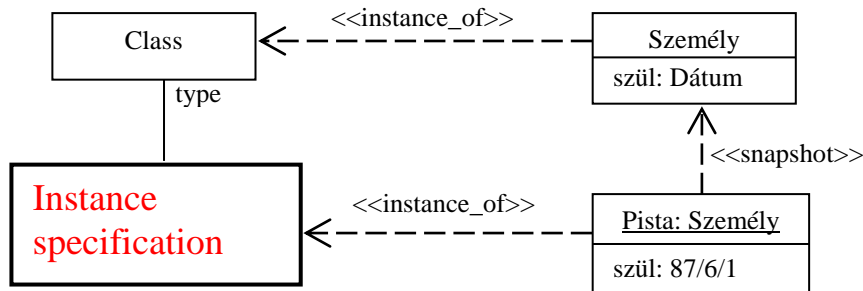
---

Hogyan értelmezzük az UML-ben az őrzött (guarded) konkurenciát ? (A konkurencia szemantikája guarded)  
*multiple calls from concurrent threads may occur simultaneously to one instance, but only one is allowed to commence. Others are blocked*

---

## 12.3 Metaclass

Nevezze meg az alábbi diagramon vastag vonallal rajzolt UML2 elemet ! (Emlékeztető: *Pista* az UML modell eleme, nem pedig a *Személy* futási időben létrejött példánya !)

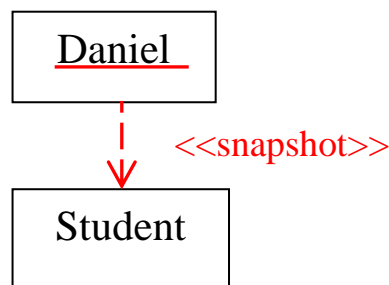


Egy UML2 modelben legyen egy Student osztályunk. Daniel a Student osztály valós idejű példányának UML2-beli modellje.

Kinek a példánya Daniel? : **.Instance specification** .....

Kinek a példánya Student? : **Class**.....

Az alábbi (nem korrekt!) részletet kiegészítve javítsa az ábrát és jelölje be a Student és Daniel közötti kapcsolatot!

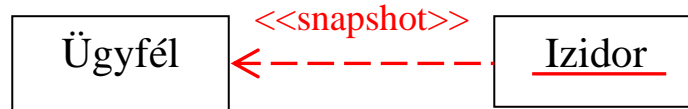


Egy UML2 modelben legyen egy Ügyfél osztályunk. Izidor az Ügyfél osztály valós idejű példányának UML2-beli modellje.

Kinek a példánya az Ügyfél? : **Class** .....

Kinek a példánya Izidor? : **.Instance specification** .....

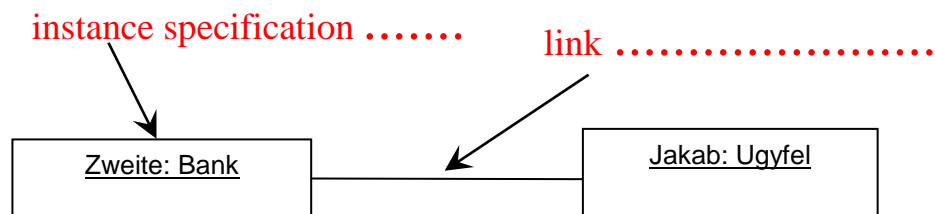
Az alábbi (nem korrekt!) részletet kiegészítve javítsa az ábrát és jelölje be az Ügyfél és Izidor közötti kapcsolatot!



A mellékelt táblázatba jelölje be, hogy a felsorolt fogalmak az UML2 4-rétegű meta-modell szerkezetének melyik rétegébe tartoznak!

	M0	M1	M2	M3
Izidor	X			
Actor			X	
State			X	
Autó		X		
Barnabás	X			
UseCase			X	
Ügyfél		X		
Ember		X		

Adja meg, hogy az alábbi object diagramon a megjelölt elemek mely UML2 meta-modell elem példányai !







A mellékelt táblázatba jelölje be, hogy a felsorolt fogalmak az UML2 4-rétegű meta-modell szerkezetének melyik rétegébe tartoznak!

	M0	M1	M2	M3
Typed Element				X
Actor			X	
State			X	
Autó		X		
Barnabás	X			
UseCase			X	
Named Element				X
Ember		X		

## 12.4 Általános

Jelölje meg, hogy a megadott rajzjelek minek az “ikon”-jai az UML2-ben !

				
felsorolás (enumeration)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
termék (artifact)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
komponens (component)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
interfész (interface)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
diszk, fájl (disc, file)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
eszköz (device)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
nincs ilyen ikon az UML2-ben	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Milyen általános kiterjesztő technikákat (general extension mechanisms) alkalmaz az UML2?

**constraint, stereotype, tagged value .....**

Adja meg az alábbi történet UML2 modelljében előforduló operációk szignatúráit! Használja a történetben szereplő szavakat!

Izidor leveszi egyik könyvét a könyvespolcra. Megnézi, hogy ki a könyv kiadója, majd azt megüzeni Emerenciának.

**levesz() : Könyv**  
**megnéz() : Kiadó**  
**oneway megüzen(Kiadó)**

Jelölje (karikázza be) az állítások igazságtartalmát, ha feltesszük, hogy szabványos UML2 nyelvet használunk!

- I** **H** Egy osztálynak több egymástól független ősz osztálya is lehet.
- I** **H** A nem definiált láthatóság attribútum publikusnak számít.
- I** **H** A use-case leírás és az adatfolyamára egyenértékű.
- I** **H** Az asszociáció, a kompozíció, a függőség és a specializáció közül a specializáció a leggyengébb.
- I** **H** Az aktivitás diagramon szereplő úszósávokat a konkurencia leírására használják.
- I** **H** Az objektum diagramon szereplő “link”-nek nincs multiplicitása.
- I** **H** Az UML2 szabvány definiálja a RUP (Rational Unified Process) fejlesztési módszertant is.
- I** **H** A kommunikációs diagramon az üzenetek sorrendjét számozással lehet megadni.

Jelölje (karikázza be) az alábbi állítások igazságtartalmát!

A polimorfizmus alkalmazásával általában csökken ...

☐ ☒ a relációban szereplő osztályok metódusainak száma

☐ ☒ a leszármazott osztályok közötti csatolás (coupling)

☒ ☐ az alternatívák (case és if szerkezetek) száma

☐ ☒ a leszármazott osztályokon belüli kohézió (cohesion)

---