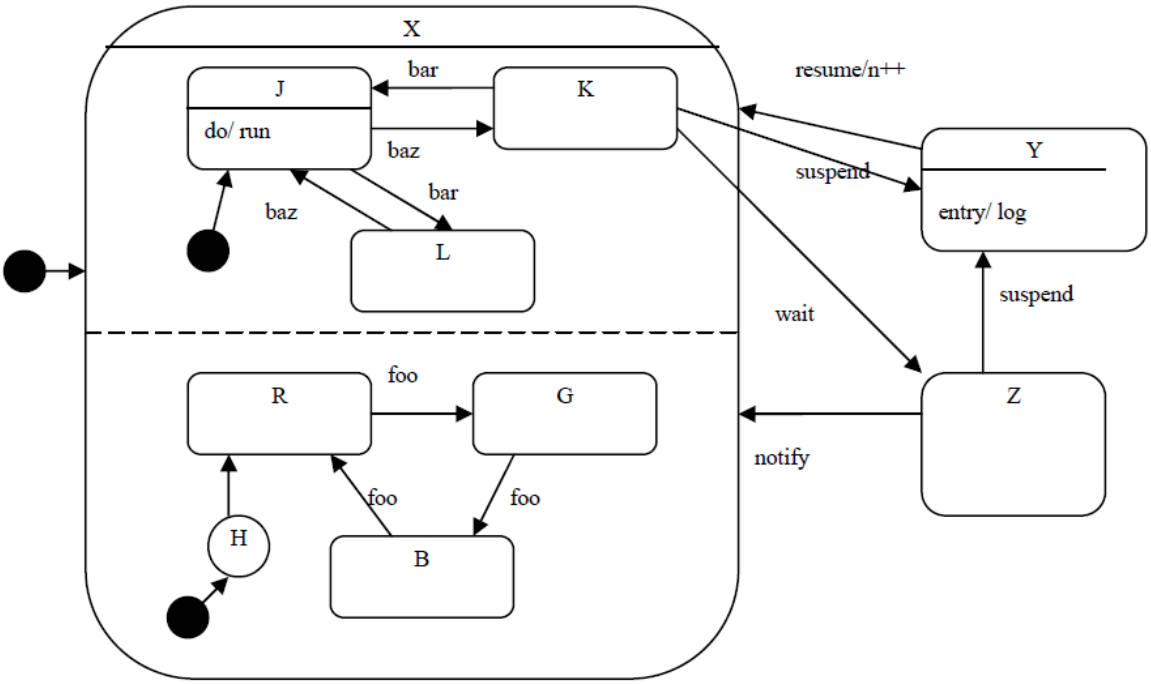
UML állapot diagram - megoldások

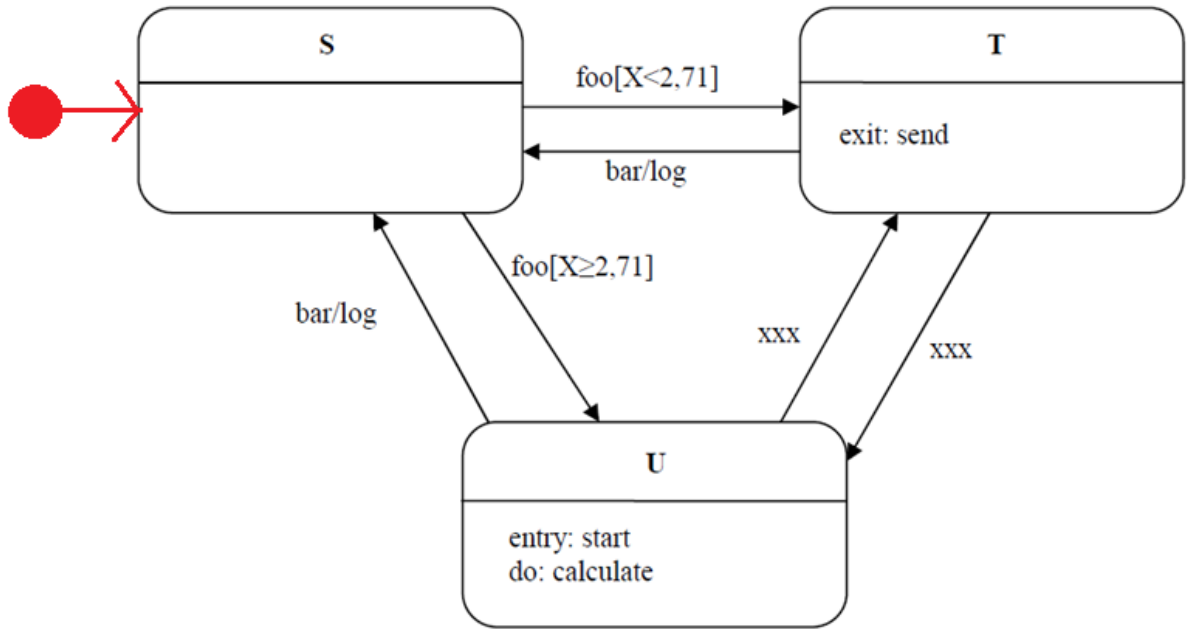
2008.01.15 – 8. Feladat

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!



2008.05.27 – 7. Feladat

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.



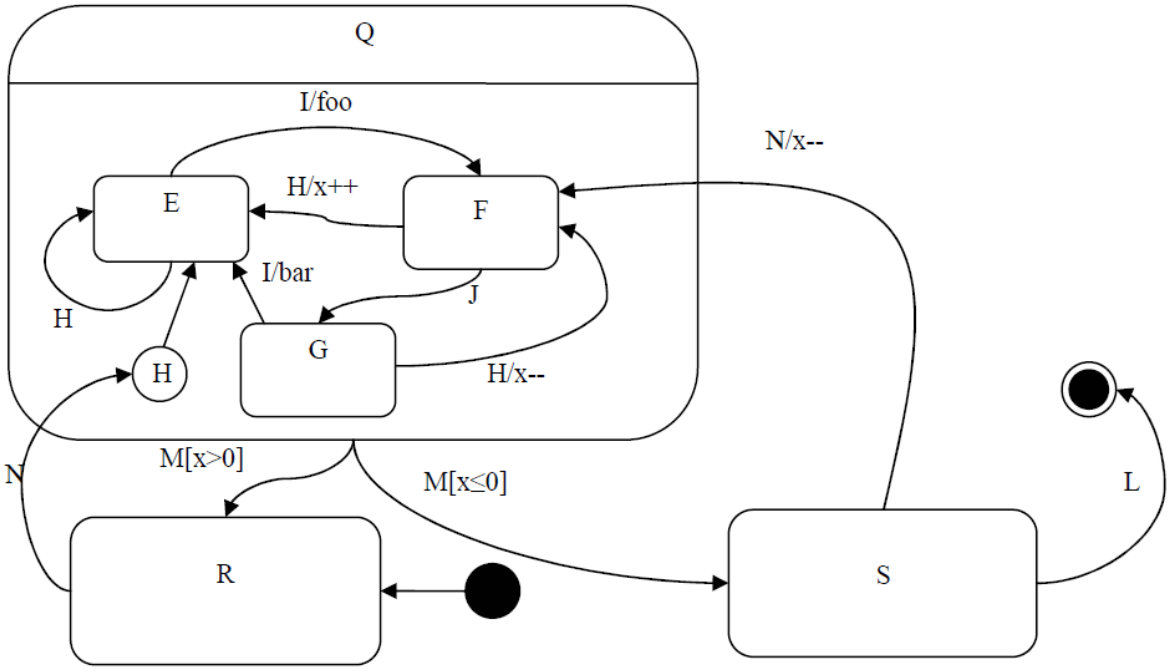
(A megoldókulcsban nem volt ott a kezdőállapot jelölése, pedig minden bizonnyal kell jelölni, hiszen a feladat még ki is tér rá!)

2008.06.17 – 9. Feladat

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:

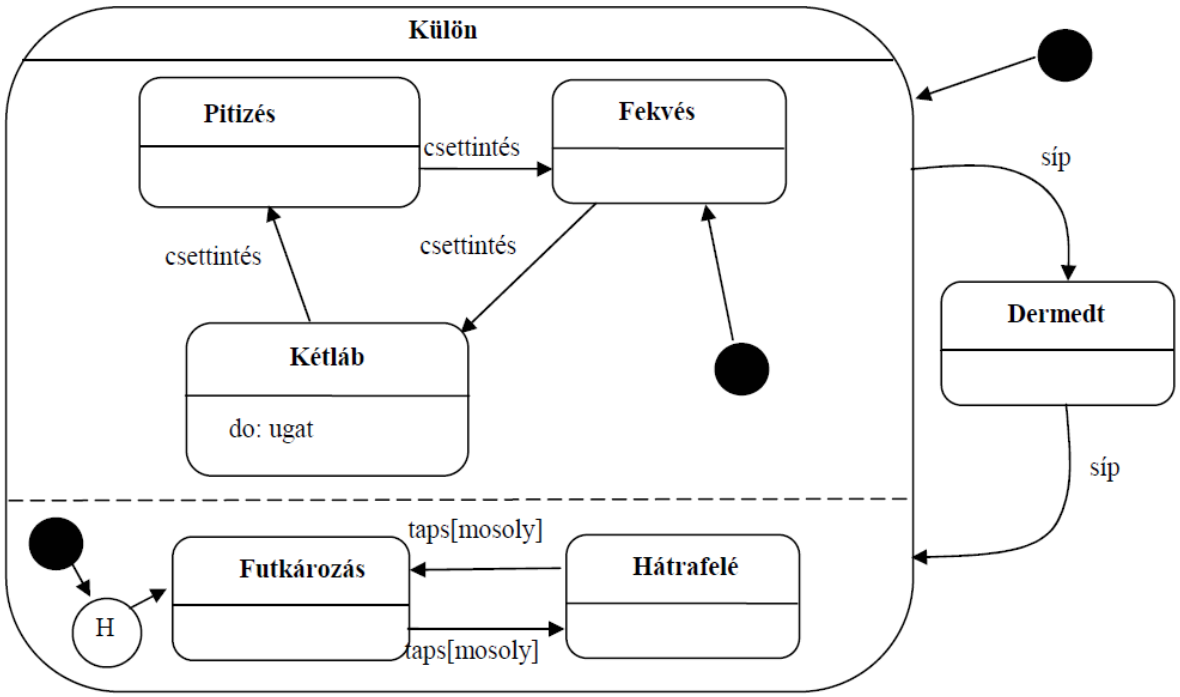
	H	**I**	**J**
E	E	F/foo	-
F	E/x++	-	G
G	F/x--	E/bar	-

A **Q**, **R**, **S** átmenetei a következők: **Q**-ból **M** esemény és pozitív **x** hatására **R**-be, nem pozitív **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 leállását eredményezi. A kezdőállapot az **R**.



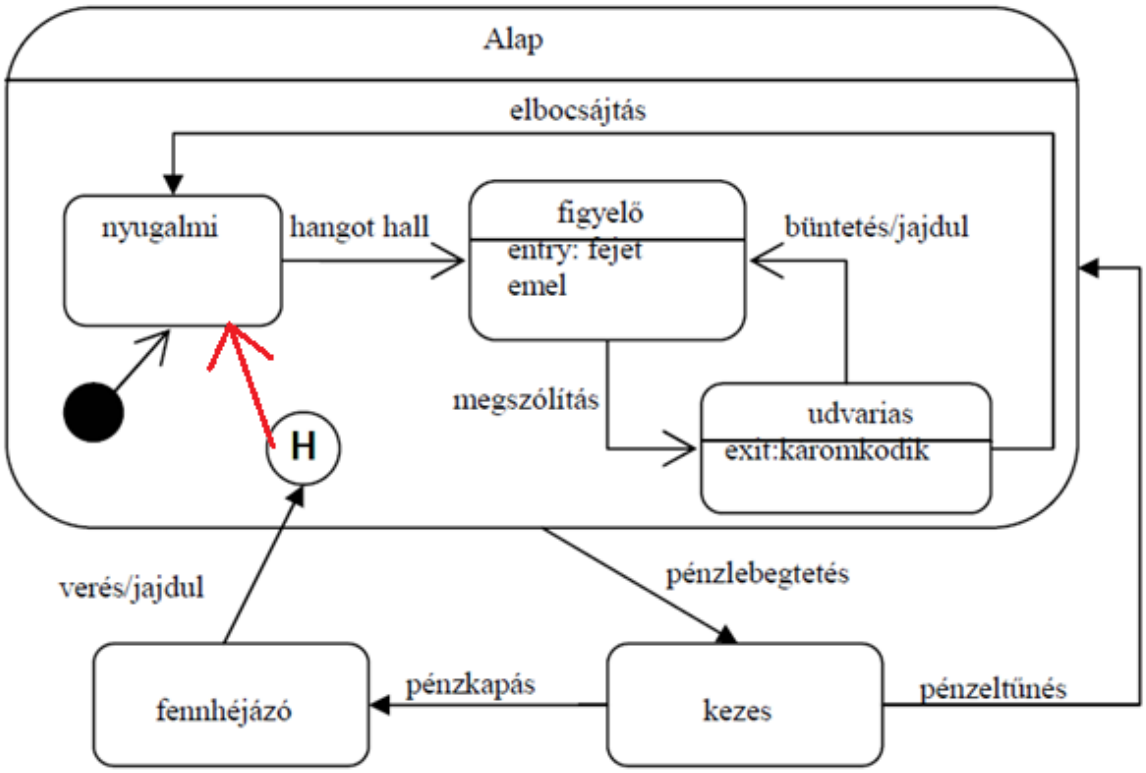
2009.01.06 – 9. Feladat

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 Rühes). 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.



2009.05.28 – 6. Feladat

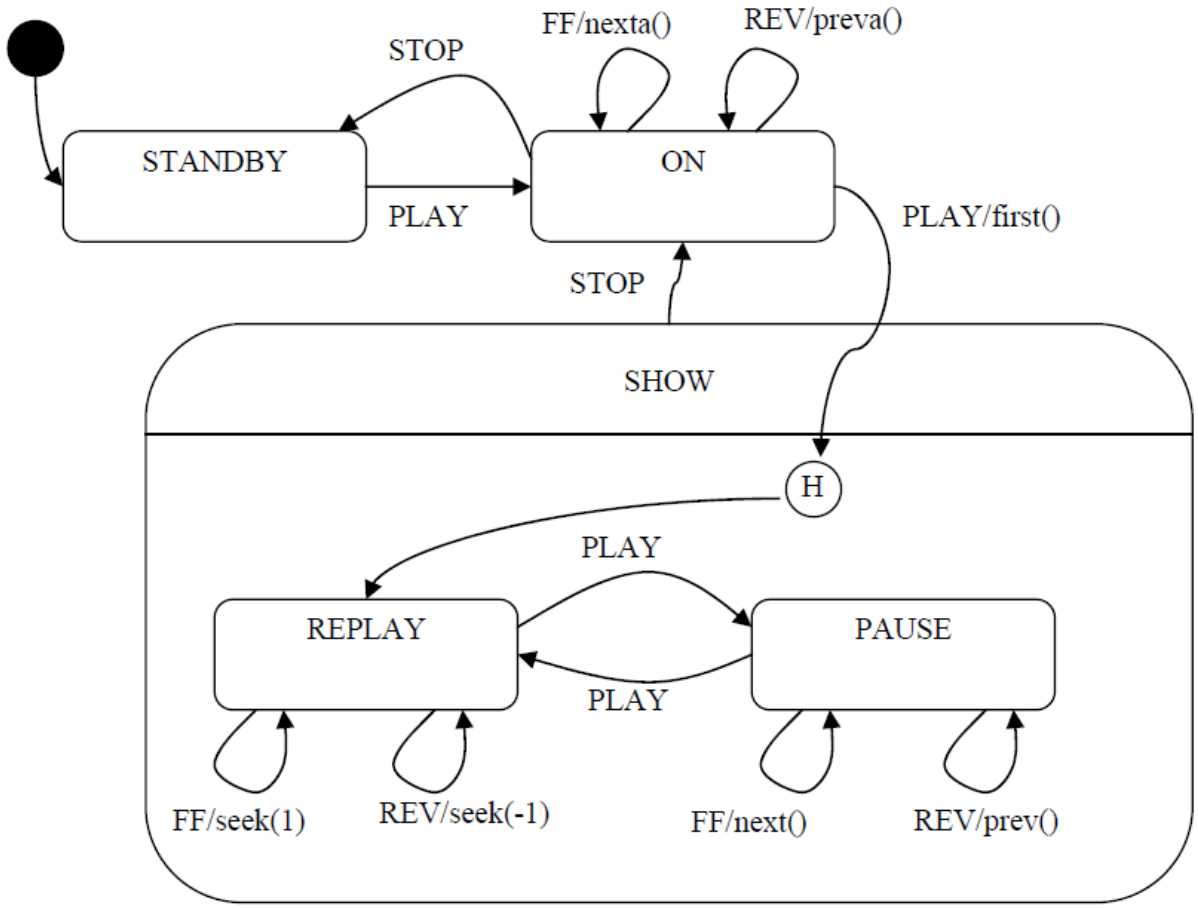
Az alábbi történet alapján rajzoljon a diagramba UML 2 állapotábrát (state chart)!
A Titanon létezik egy Lowyir-nek keresztelt parazita életforma. A Lowyir életét nyugalmi állapotban kezdi. Ha hangot hall, figyelő állásba lép (figyelő állás kezdetén mindig felemeli a fejét). Ha ekkor megszólítják, akkor 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 kezes 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.



(Itt azért szerintem illene vagy a History-t bekötni a kezdőállapotba, vagy a kiindulási pontot a History-ba, és úgy a History-t a kezdőállapotba, lásd az előző, 2009.01.06 – 9. Feladat-t.)

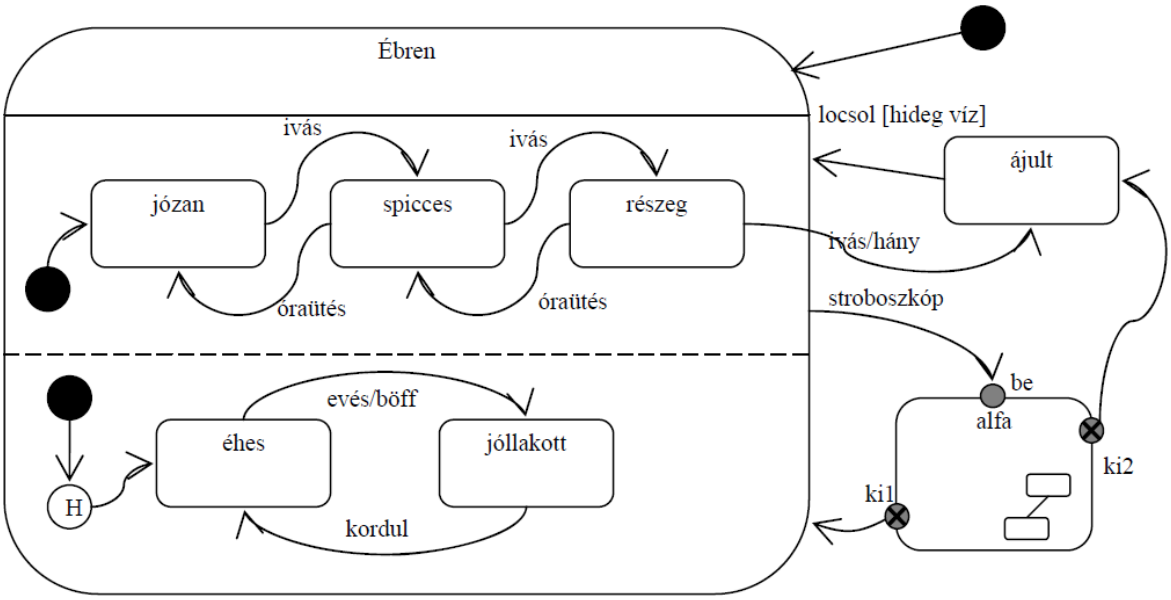
2010.01.05 (B) – 9. Feladat

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 szüneteltetni 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.



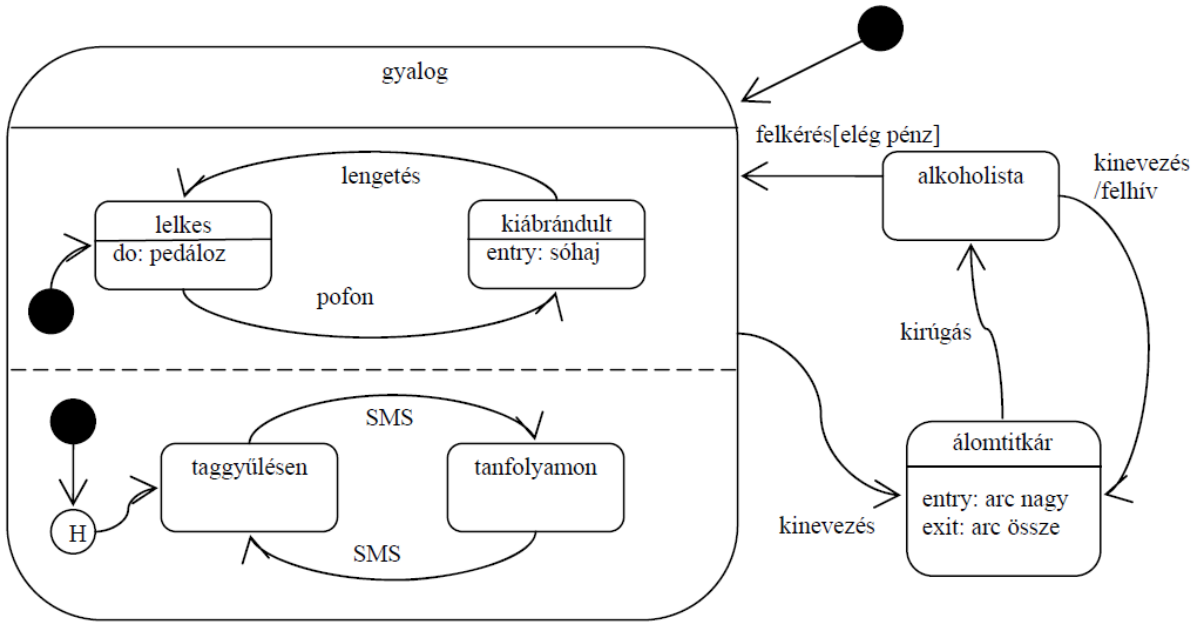
2010.01.12 (A) – 7. Feladat

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 visszafele változik. Ha éhes és eszik, akkor elböffenti magát és jóllakott lesz. Akkor éhezik 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. Mindezeken 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.



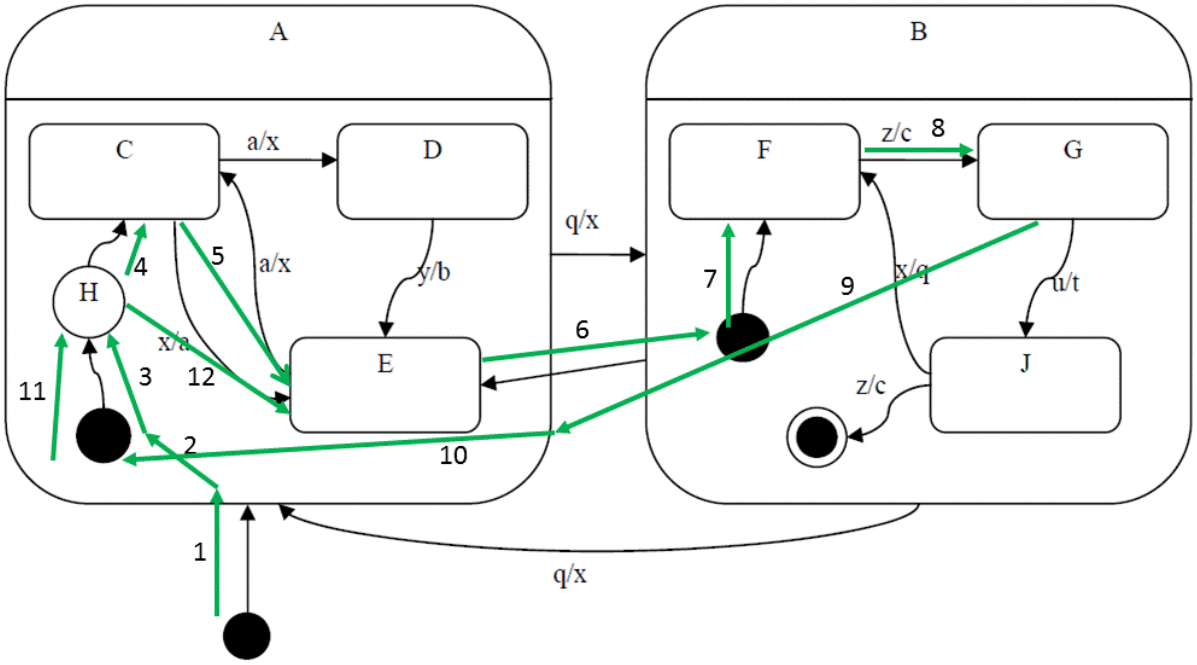
2010.01.26 – 7. Feladat

Rajzoljon UML 2.0 állapotábrát (state chart) az alábbi történet alapján!
A Stupiditas nevű szervezet tagja *gyalog*ként (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 összejövetelen lehet megtalálni: *taggyűlésen* és *tanfolyamon* (a taggyűlés az első). Ha az egyiken SMS-t kap, átmegy a másikra. Mikor kinevezik *álomtitkár*nak, 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 összejövetelen folytatja, ahol utoljára gyalogként megfordult. Az alkoholizmusból egy újabb álomtitkári kinevezés is kigyógyítja. Ilyenkor felhívja anyukáját.



2010.05.26 – 1. Feladat

A következő UML2 állapotdiagram alapján minősítse az állításokat!



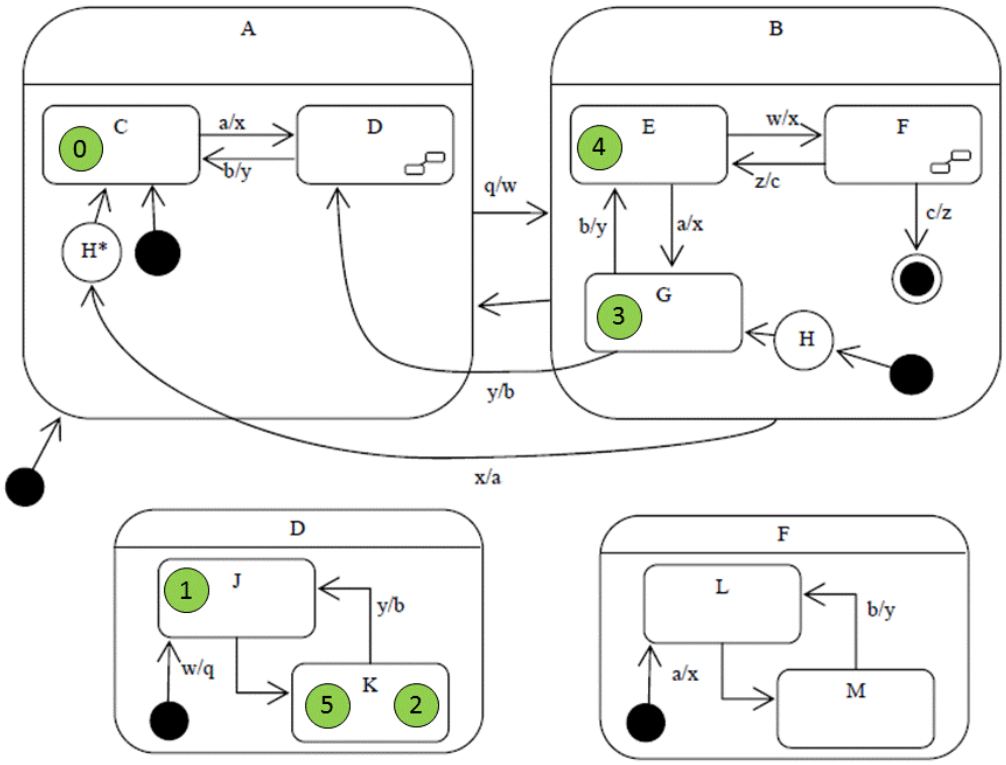
IGAZ	HAMIS	Állítás
x		D állapotból 2 lépésben visszaérhet D-be *History miatt: D-ből q-val F-be, majd egy újabb q-val vissza D-be.*
	x	F állapotból „q” esemény hatására H állapotba kerül *A H az „nem egy állapot”, ott nem állhatunk meg.*
x		B-ből A-ba való váltáskor végrehajtódhat a „c” tevékenység *Ha J állapotban „z” esemény jön.*
	x	E állapotból egyetlen esemény hatására csak a C állapot következhet *Hamis, „q” esemény hatására F-be kerülünk.*
x		J állapotból egyetlen esemény hatására E állapot következhet *Ha a Historyban az E állapot került mentésre, akkor „q” eseményre E jön.*

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

IGAZ	HAMIS	Állítás
	x	C állapotba kerülünk *E állapotba kerülünk.*
x		Kétszer lefut az „x” tevékenység *A 2 „q” eseménynél futnak le.*
	x	Érintjük a J állapotot *Ábra alapján látszik, hogy nem.*

2010.06.01 – 1. Feladat

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



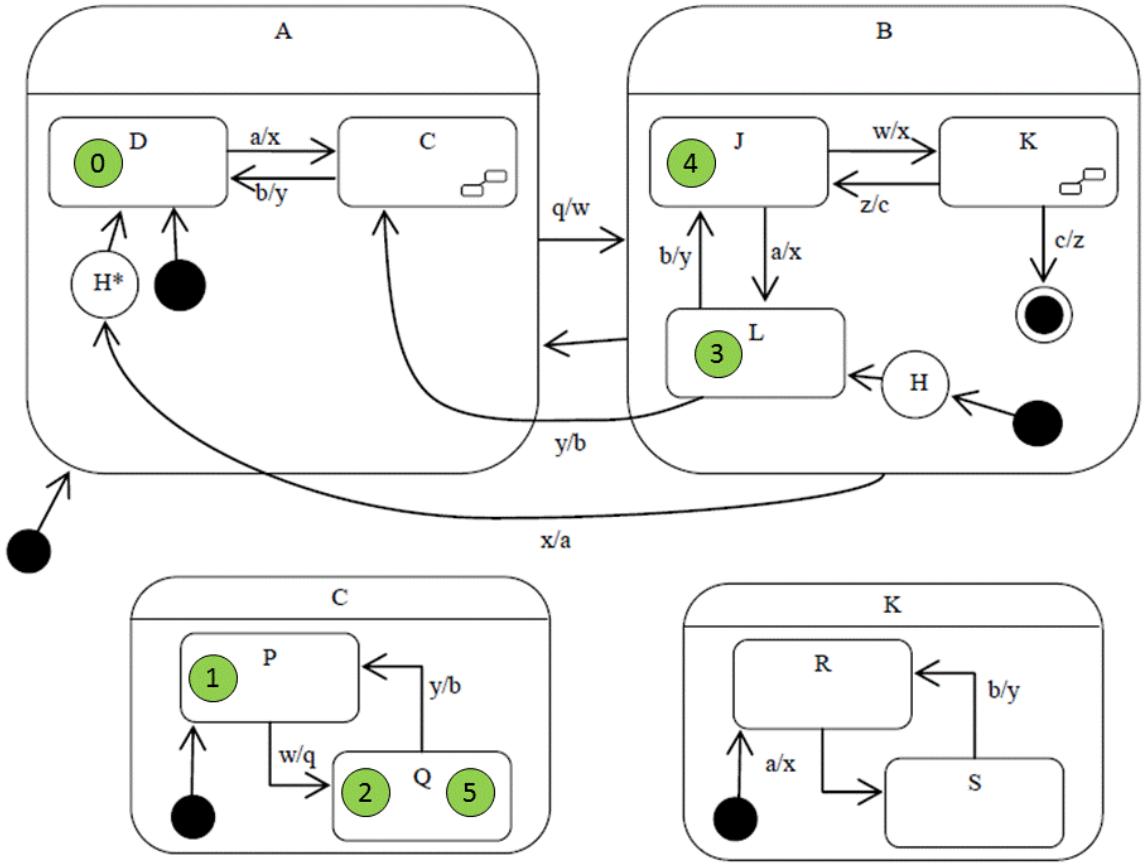
IGAZ	HAMIS	Állítás
	x	G állapot után csak D és E következhet egy lépésben *Következhet C is „x” esemény hatására.*
	x	D állapotból elérhető egy lépésben M *B állapotban csak H van, nem H*, így az a belső állapotot nem jegyzi meg, így F-ben az első állapot mindig az L lesz.*
x		G állapot után közvetlenül következhet K állapot *Mivel A-ban H* van, így az megjegyzi a belső állapokat is, így ha A-n belül utoljára K állapotban voltunk, akkor G-ből „x” esemény hatására K-ba kerülünk.*
	x	K állapotból „y” esemény hatására átlépünk C-be *Nope, J-be kerülünk.*
x		F állapotból csak „c” és „x” esemény hatására léphetünk át A állapotba *Egyedül a „z” esemény jöhet még F-ben, az meg E-be visz, így ez igaz.*

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

IGAZ	HAMIS	Állítás
x		K állapotba kerülünk
	x	érintettük az F állapotot
	x	pontosan kétszer fut le a „q” tevékenység *Csak 1x, a w esemény lefutásának hatására.*

2010.12.21 – 1. Feladat

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



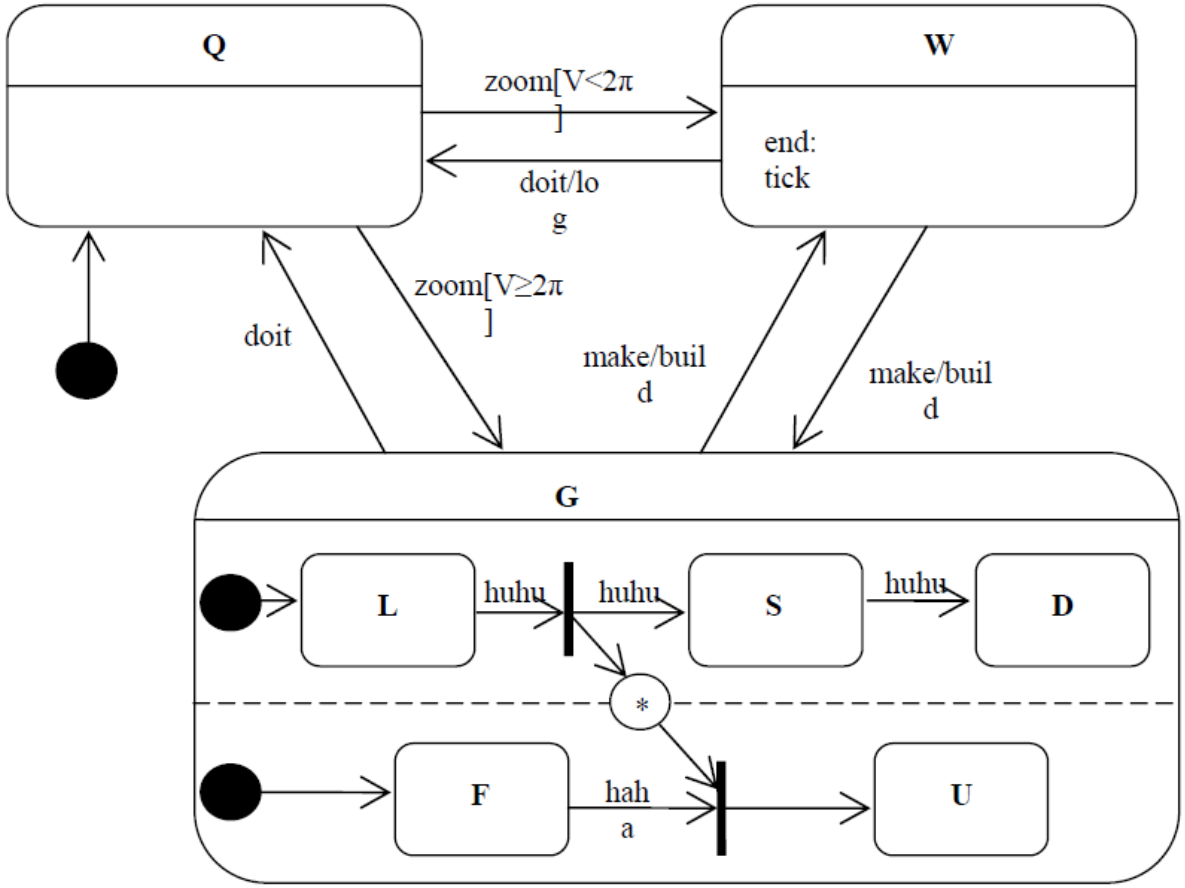
IGAZ	HAMIS	Állítás
X		L állapot után közvetlenül következhet Q állapot *A H* elmenti C belső állapotát is, így „x” esemény hatására Q-ba kerülhetünk.*
	X	C állapotból elérhető egy lépésben S *Mivel B-ben csak H-van, így nem (K-n belül mindig R-be jutunk először).*
X		K állapotból csak „c” és „x” esemény hatására léphetünk át A állapotba *Csak „z” esemény jöhet, azzal meg J-be jutunk, így ez igaz.*
	X	Q állapotból „y” esemény hatására átlépünk D-be *P-be lépünk, nem a D-be.*
	X	L állapot után csak C és J következhet egy lépésben *„x” eseény hatására a D-be is juthatunk.*

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

IGAZ	HAMIS	Állítás
	X	pontosan kétszer fut le a „q” tevékenység *Csak 1x fut le, „w” esemény hatására.*
X		Q állapotba kerülünk
	X	érintettük az K állapotot

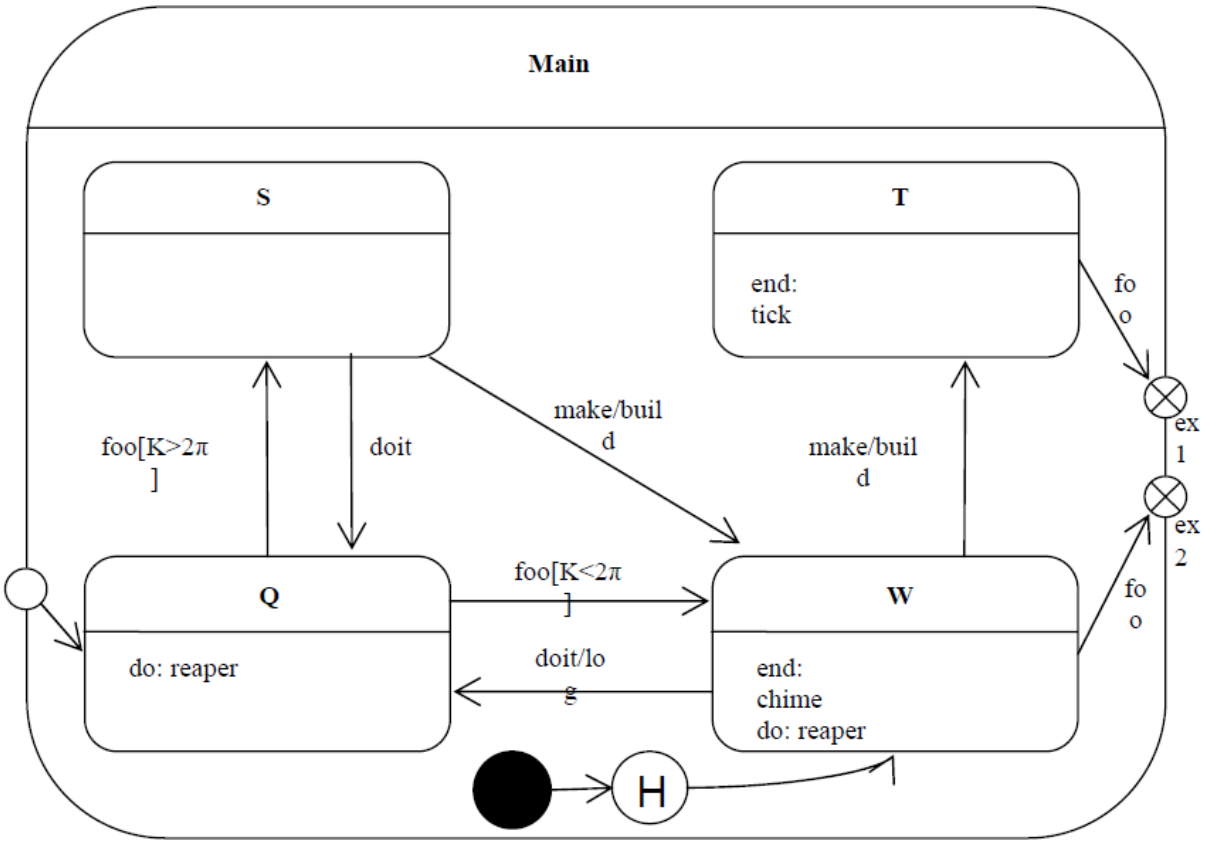
2011.01.04 (A) – 6. Feladat

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 2π 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.



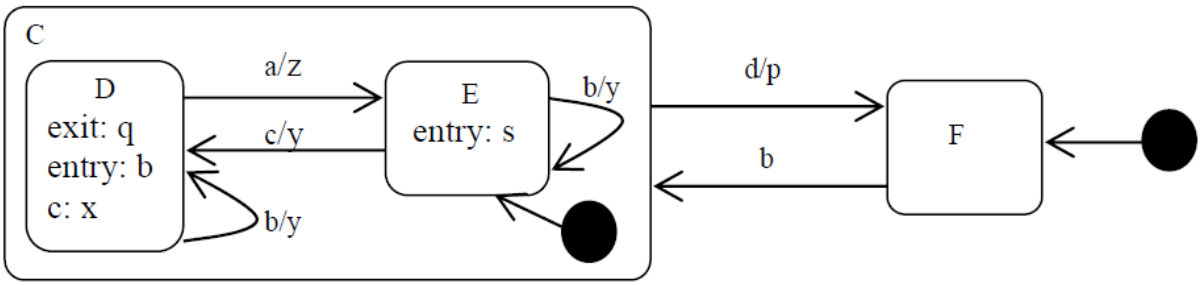
2011.01.04 (B) – 6. Feladat

Egészítse ki az alábbi UML 2 állapotdiagramot (state chart) a következő leírás alapján!
Egy objektum **Main** állapotában 4 alállapot található (**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 2π 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.



2011.01.18 – 5. Feladat

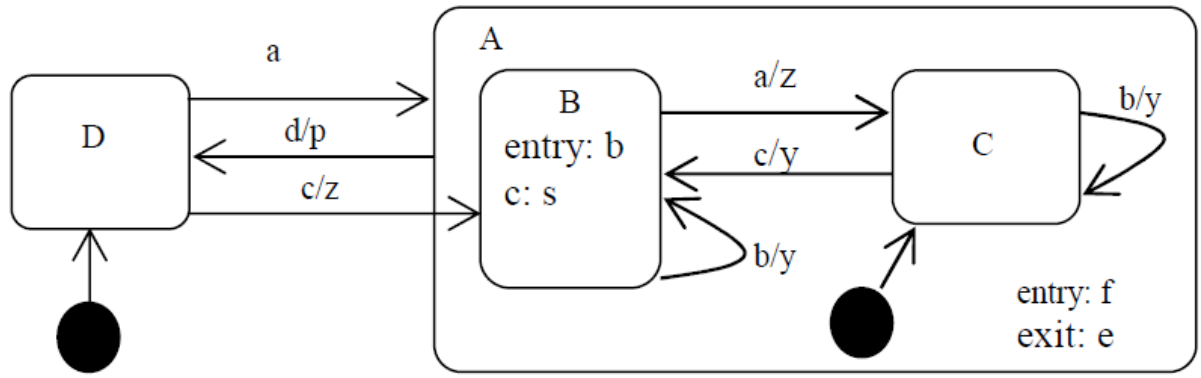
Rajzolja fel az alábbi UML2 state-chart-nak megfelelő állapottáblát!



	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	-	-

2011.05.24 – 5. Feladat

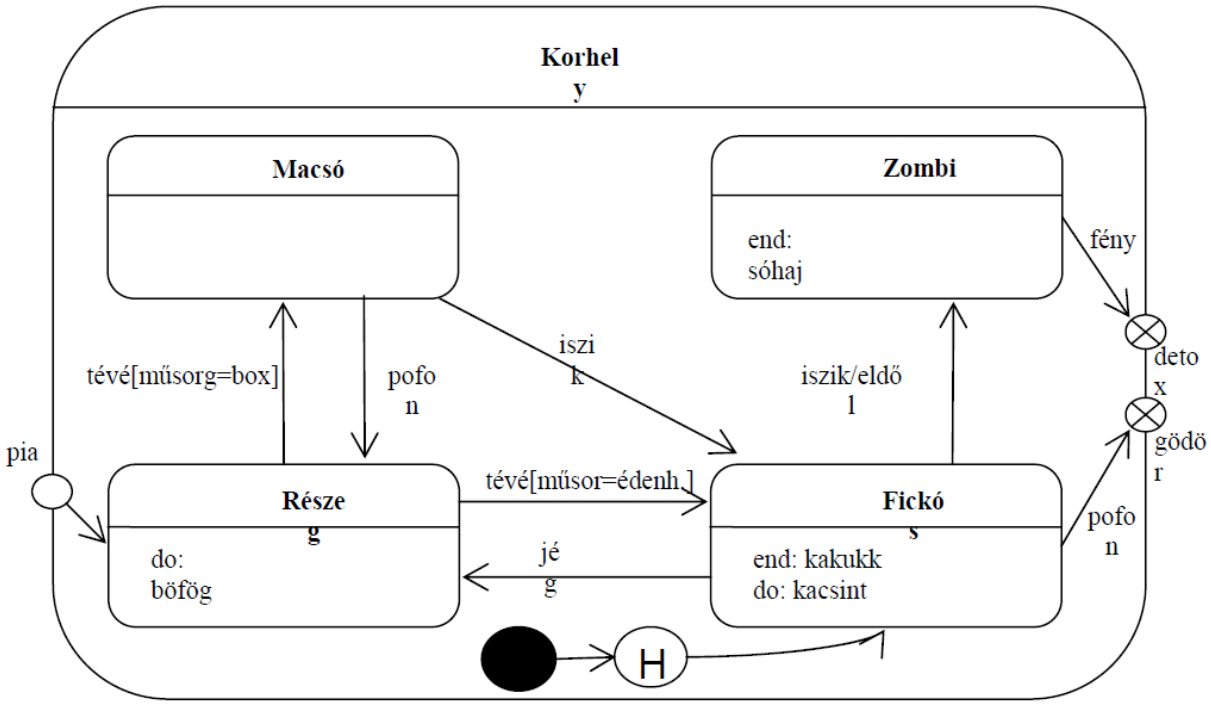
Rajzolja fel az alábbi UML2 state-chart-nak megfelelő állapottáblát!



	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

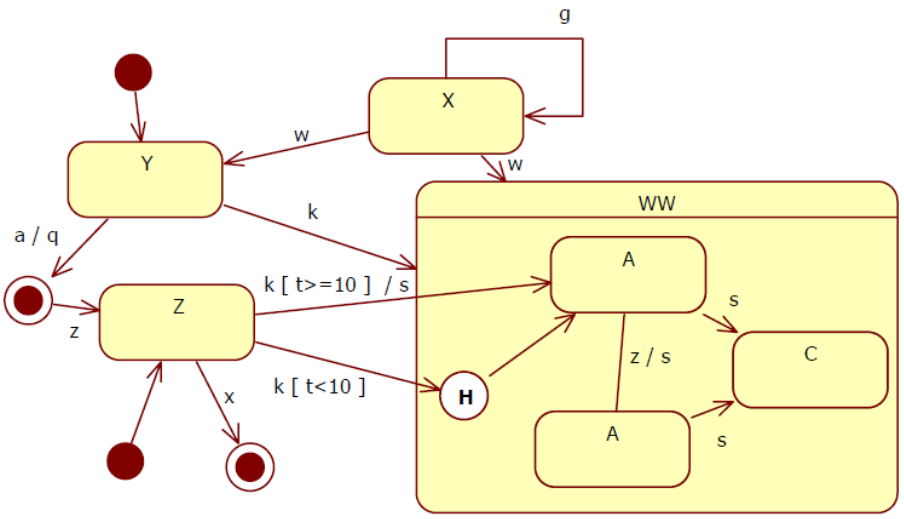
2011.06.07 – 10. Feladat

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 boksz 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 sóhajt. Zombiból fény hatására a detox-ba kerül.



2012.01.03 – 8. Feladat

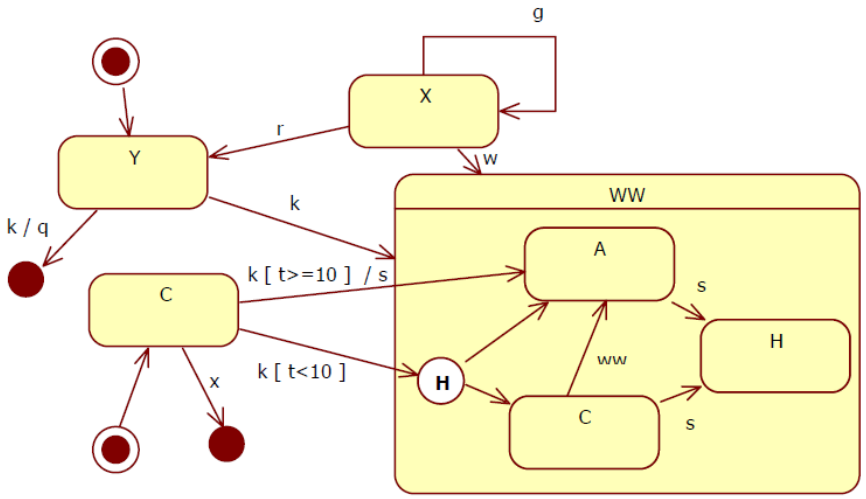
Milyen szintaktikai és szemantikai hibák találhatók az alábbi UML2 állapot-diagramon (state-chart)? /Segítségképp: 8 hiba van benne./



- Két kezdőállapot
- Ugyanarra az eseményre két állapotba is mehet
- Végállapotból kilépünk
- Nincs kezdőállapot
- Ugyanolyan nevű állapotok
- Nincs nyíl
- Csak kimenő állapot
- Csak bemenő állapot

2012.05.22 – 7. Feladat

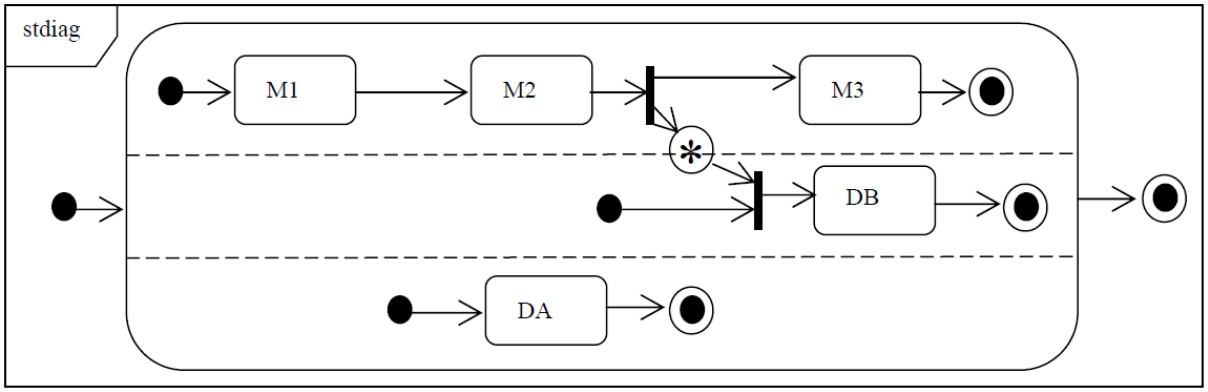
Milyen szintaktikai és szemantikai hibák találhatók az alábbi UML2 állapot-diagramon (state-chart)? /Segítségképp: 8(7) hiba van benne./



- Végállapotból kilépünk
- Kezdőállapotba belépünk
- *(avagy: 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 historyból kétfele is megyünk
- X forrás (nincs belépő átmenet)

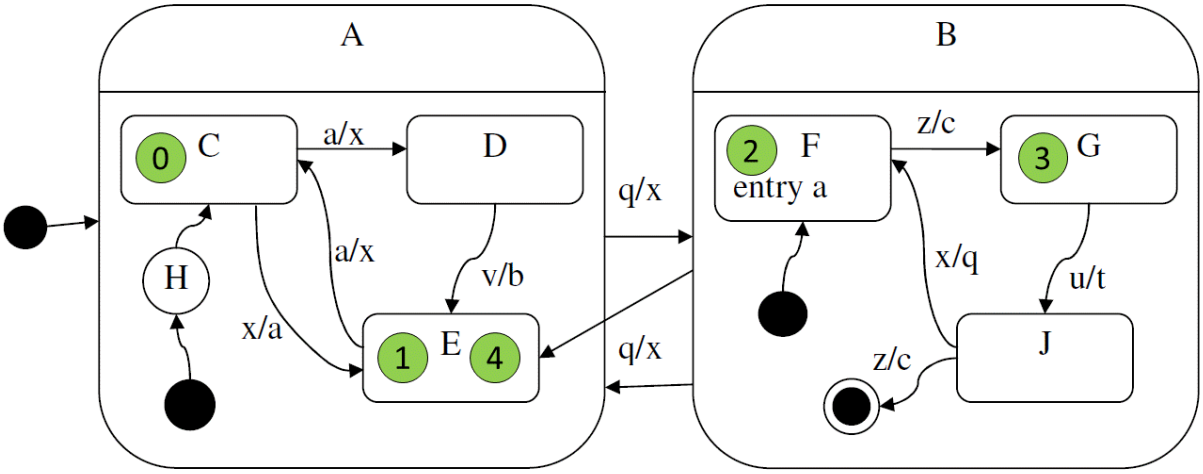
2012.06.05 – 8. Feladat

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!



2013.01.08 – 9. Feladat

A következő UML2 állapotdiagram alapján minősítse az állításokat!



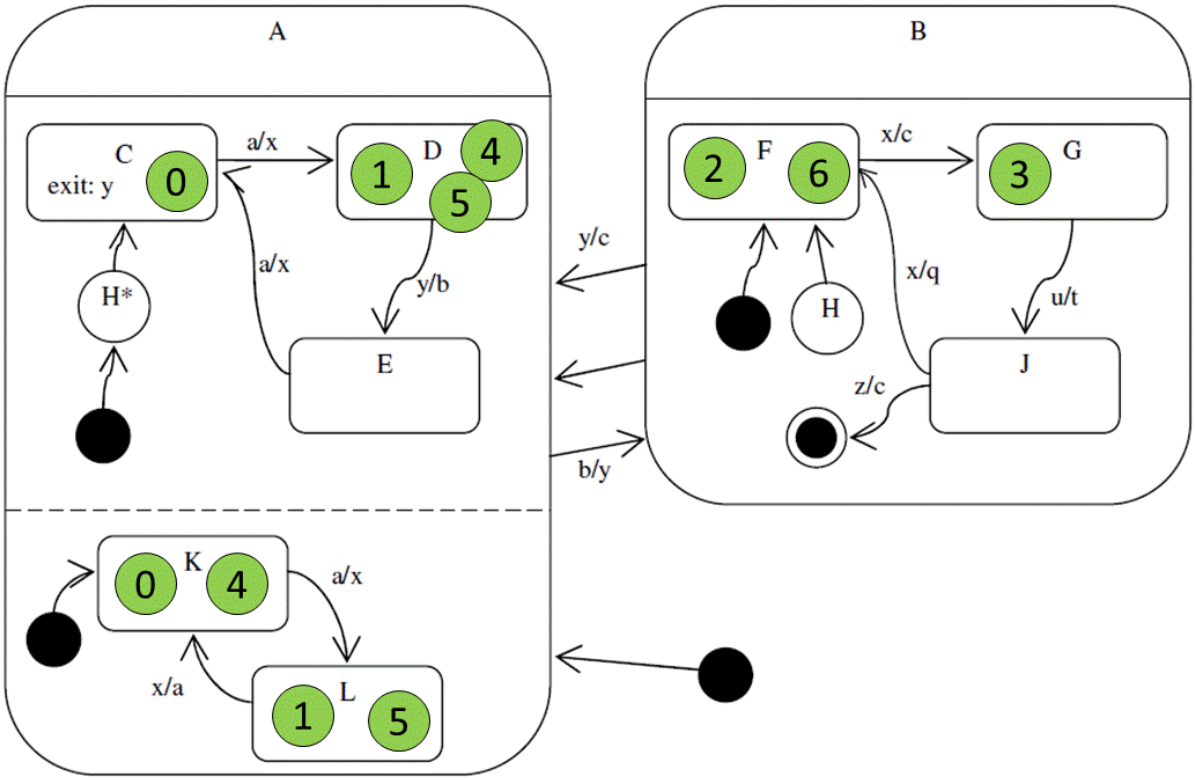
IGAZ	HAMIS	Állítás
x		D állapotból 2 lépésben visszaérhet D-be *D-ből „q”-val F-be, majd egy újabb „q”-val H-nak köszönhetően vissza D-be.*
x		J állapotból egyetlen esemény hatására D állapot következhet *„z” esemény hatására H segítségével D-be kerülhetünk.*
	x	F állapotból „q” esemény hatására H állapotba kerül *A H az „nem egy állapot”, ott nem állhatunk meg.*
	x	B-ből A-ba való váltáskor végrehajtódhat az „a” tevékenység

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

IGAZ	HAMIS	Állítás
x		Kétszer lefut az „x” tevékenység *A 2 „q” eseménykor.*
x		E állapotba kerülünk
x		Kétszer lefut az „a” tevékenység *Az „x” eseménykor, ill. amikor az F-be lépünk.*

2013.06.11 – 4. Feladat

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



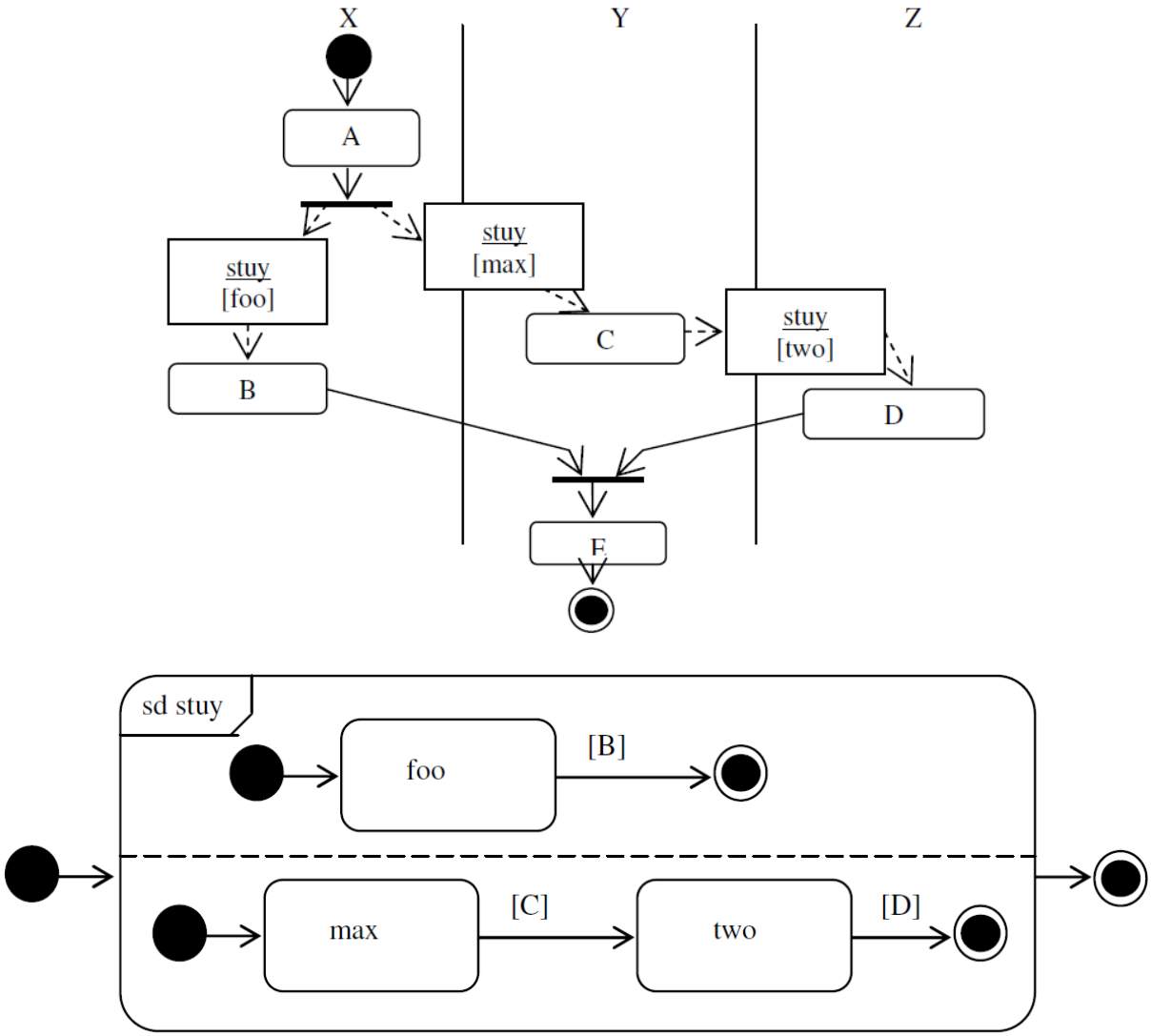
IGAZ	HAMIS	Állítás
	X	H állapotból bármely esemény bekövetkeztekor F állapotba jutunk. *Kerülhetünk G-be, és J-be is.*
X		B állapot elhagyásakor a 'c' tevékenység pontosan egyszer hajtódik végre. *Vagy J-ből „z" esemény hatására, vagy bármely B-s állapotból „y" hatására.*
X		G-ből két lépésben eljuthatunk L-be. *„y" esemény, majd „a" esemény.*
	X	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
	X	pontosan kétszer fut le az 'x' tevékenység. *3x (2x az 1. „a"-nál, és 1x a 2. „a"-nál amikor K-ból L-be megyünk).*
	X	a végén G állapotba kerülünk. *F állapot a nyerő.*
X		pontosan kétszer fut le a 'c' tevékenység. *1x az „x", majd az „y" eseménynél.*
X		pontosan kétszer érintettük az L állapotot.

2013.06.11 – 6. Feladat

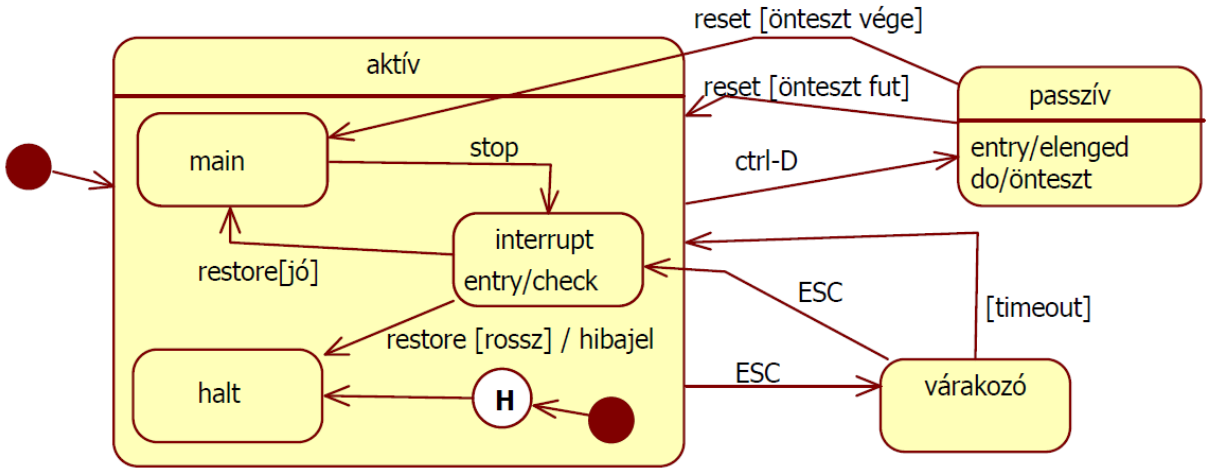
Adott a mellékelt – object flow-val kiegészített – aktivitás-diagram (activity diagram)! Rajzolja meg azon objektumok UML2 state-chartját, amelyeknek az ábra alapján több állapota is van!



2013.06.18 – 9. Feladat

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 checket. 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 véget ér 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.

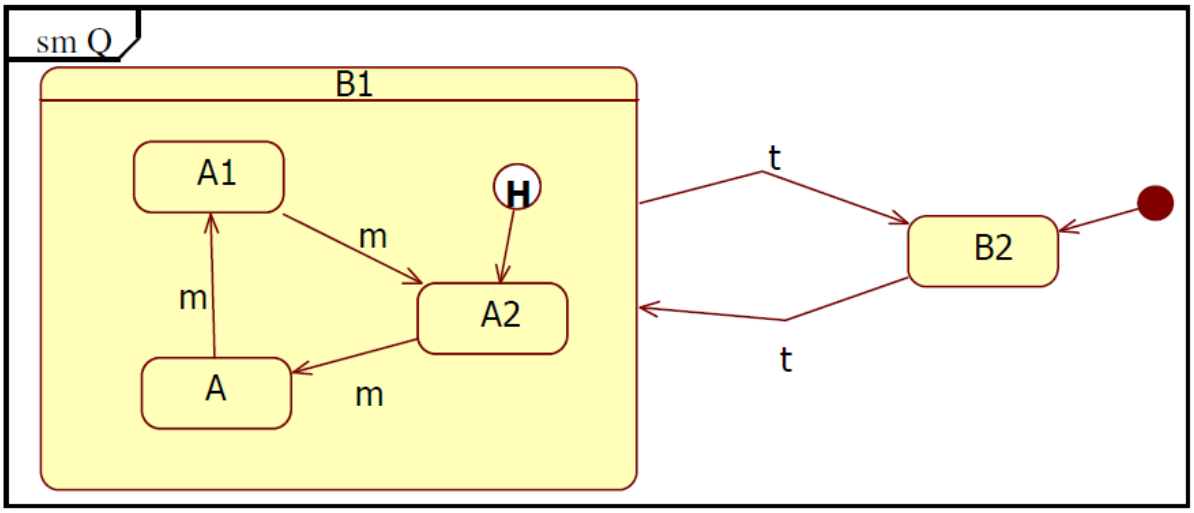


2014.01.07 – 6. Feladat

Rajzoljon UML2 állapotdiagramot a Q osztályhoz!

```
enum AS { A1, A2, A3 }
enum BS { B1, B2 }

class Q {
  private AS as = AS.A2;
  private BS bs = BS.B2;

  public void t() {
    if (bs == BS.B1) {bs = BS.B2;}
    else {bs = BS.B1;}
  }
```

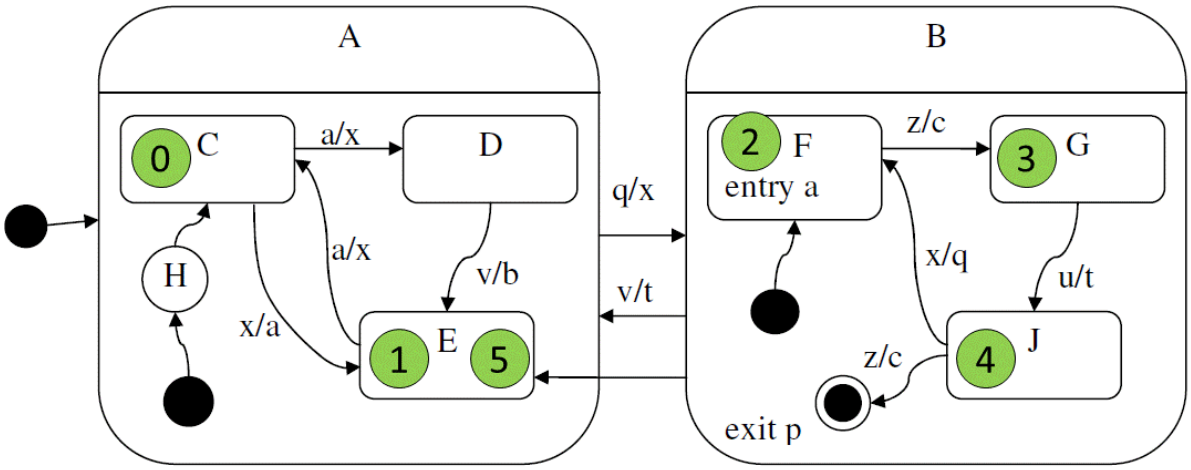
```
  public void m() {
    if (bs == BS.B1) {
      switch (as) {
        case A1: as = AS.A2; break;
        case A2: as = AS.A3; break;
        case A3: as = AS.A1; break;
      }
    }
  }

}
```



*/*Én a B-be azért rajzolnék egy kezdőállapotot (ami a H-ba vezet), legalábbis korábbi feladatok alapján ezt így szokás...*/*

2014.01.07 – 7. Feladat

A következő UML2 állapotdiagram alapján minősítse az állításokat!



IGAZ	HAMIS	Állítás
X		D állapotból 2 lépésben visszaérhet D-be *D-ben jön egy „q”, majd „v” eseménnyel visszajutunk H miatt D-be.*
	X	J állapotból „p” esemény hatására E állapotba kerül *„p” esemény nincs is, az „p” tevékenység (B-t elhagyva mindig le kell futtatni a „p” tevékenységet).*
	X	F állapotból „v” esemény hatására H állapotba kerül *A H az „nem egy állapot”, ott nem állhatunk meg.*
X		A-ból B-be való váltáskor mindig végrehajtódik az „a” tevékenység *Mivel B-ben mindig az F állapot a kezdő, aminek entry-je az „a” tevékenység, így igaz.*

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

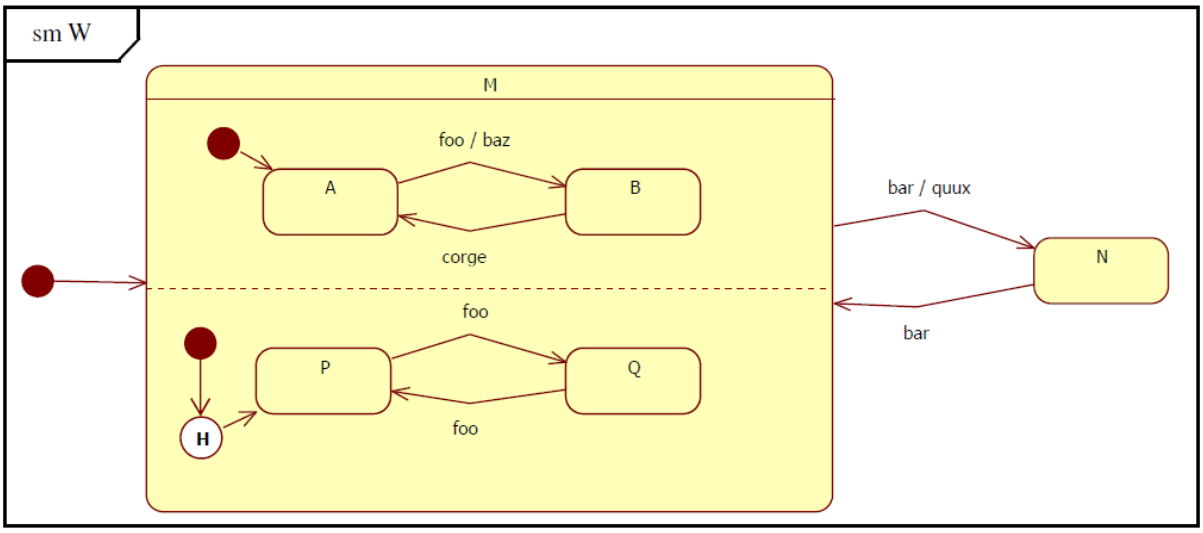
IGAZ	HAMIS	Állítás
X		Egyszer lefut a „p” tevékenység *Amikor elhagyjuk B-t az utolsó „z” esemény hatására.*
	X	C állapotba kerülünk *E állapotba kerülünk.*
X		Az E állapotot kétszer érintjük
X		Kétszer lefut az „a” tevékenység *1x az „x” esemény bekövetkezésekkor, majd utána, amikor F-be kerülünk.*

2013.01.21 – 6. Feladat

Rajzoljon UML2 állapotdiagramot a W osztályhoz!

```
enum XS { A, B }
enum YS { P, Q }
enum ZS { M, N }

class W {
  private XS xs = XS.A;
  private YS ys = YS.P;
  private ZS zs = ZS.M;

  public void foo() {
    if (zs == ZS.M) {
      if (xs == XS.A) {
        xs = XS.B;
        baz();
      }
      if (ys == YS.P) ys = YS.Q;
      else ys = YS.P;
    }
  }
```

```
  public void bar() {
    if (zs == ZS.M) {
      zs = ZS.N;
      quux();
    } else {
      zs = ZS.M;
      xs = XS.A;
    }
  }

  public void corge() {
    if (zs == ZS.M) {
      if (xs == XS.B) xs = XS.A;
    }
  }

  private void baz() { }
  private void quux() { }
}
```

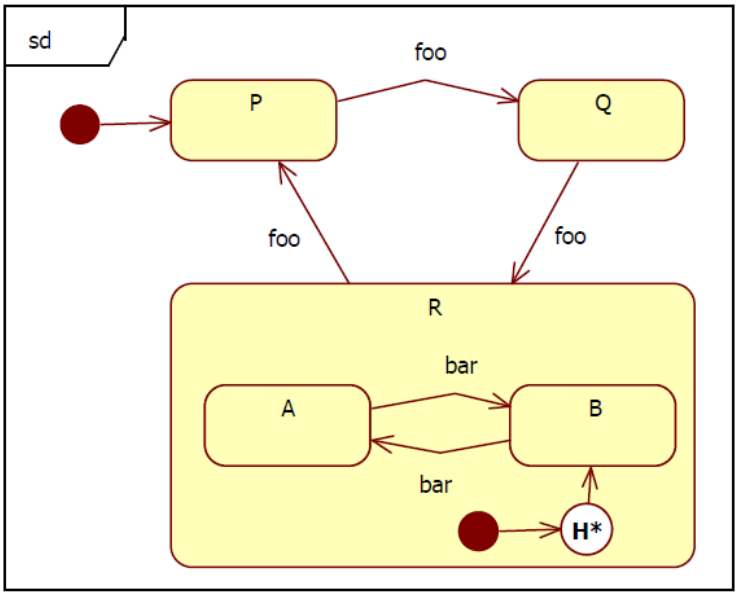


2014.05.27 – 5. Feladat

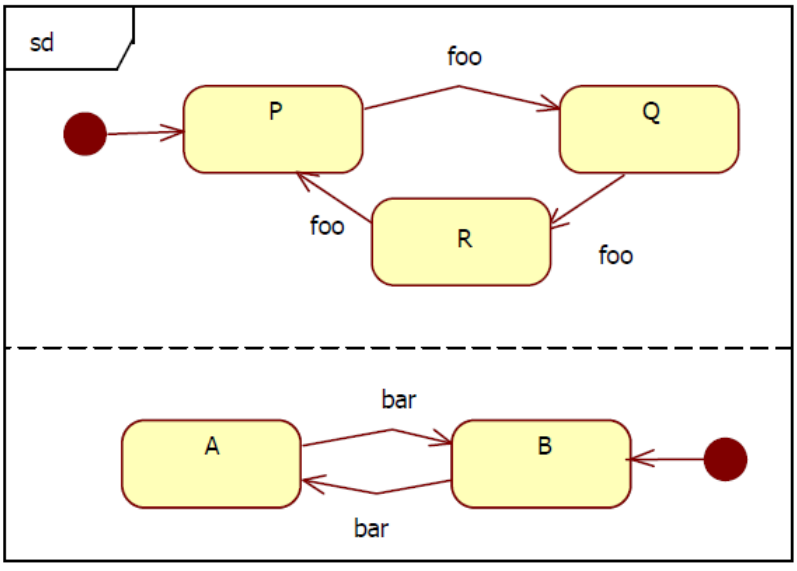
Rajzoljon UML2 állapotdiagramot a Garply osztályhoz!

```
enum PQR {P, Q, R}
enum AB {A, B}

public class Garply {
  private PQR pqr;
  private AB ab;

  public Garply() {
    pqr = PQR.P; ab = AB.B;}

  public void bar() {
(1) if (pqr == PQR.R) {
      if (ab == AB.A) ab = AB.B;
      else ab = AB.A;
(1) }
  }
```

```
  public void foo() {
    switch (pqr) {
    case P:
      pqr = PQR.Q; break;
    case Q:
      pqr = PQR.R; break;
    case R:
      pqr = PQR.P; break;
    }
  }
}
```

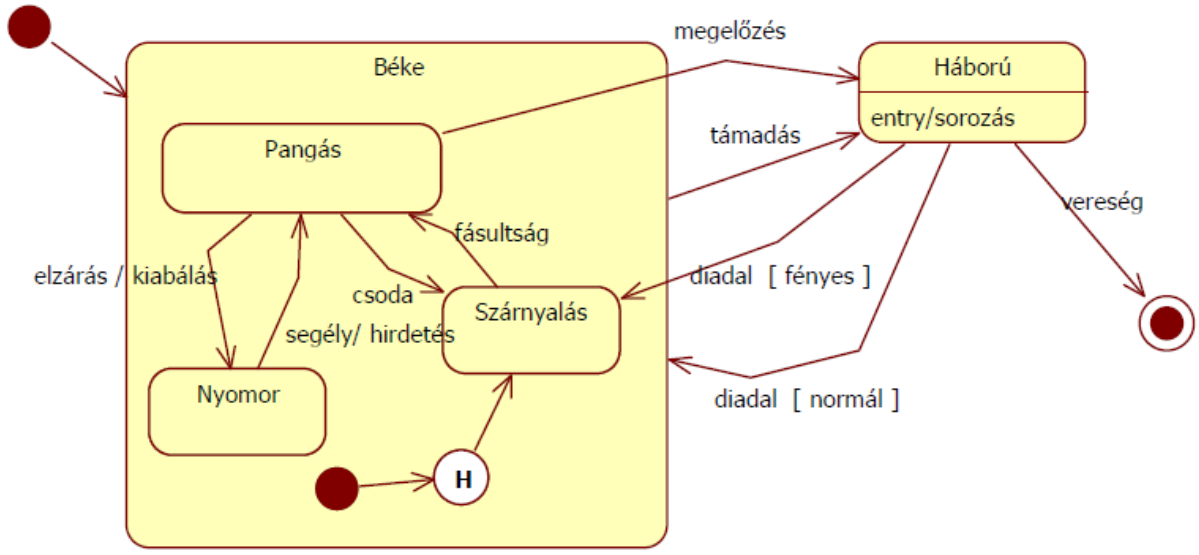


Rajzolja fel az UML2 állapotdiagramot, ha a programból töröljük a két (1)-val jelzett sort!



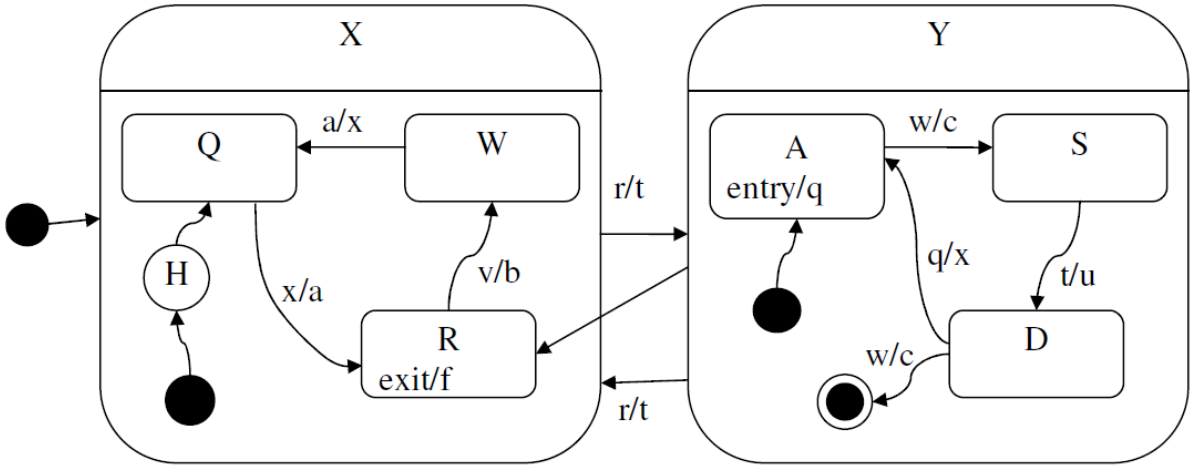
2014.06.03 – 6. Feladat

Rajzoljon UML2 állapotdiagramot az alábbi leírás alapján! (4 pont)
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 abbahagyta, fényes diadal esetén azonban szárnyalás jön.



2014.06.17 – 4. Feladat

A következő UML2 állapotdiagram alapján minősítse az állításokat! (7 pont)



IGAZ	HAMIS	Állítás
	x	R állapotból 2 lépésben nem lehet visszaérni R-be. *R-ből „r”-re megyünk A-ba, majd egy újboli „r” eseményre a History miatt vissza R-be.*
x		A állapotból egyetlen esemény hatására W állapot következhet. *A History miatt lehetséges (szintén „r” esemény hatására).*
	x	S állapotból „r” esemény hatására H állapotba kerül. *A H az nem egy állapot, az a History.*
x		X-ből Y-ba való váltáskor végrehajtódhat az „f” tevékenység. *Az R-ben jön egy „r”, ekkor R-t elhagyva lefut az f.*

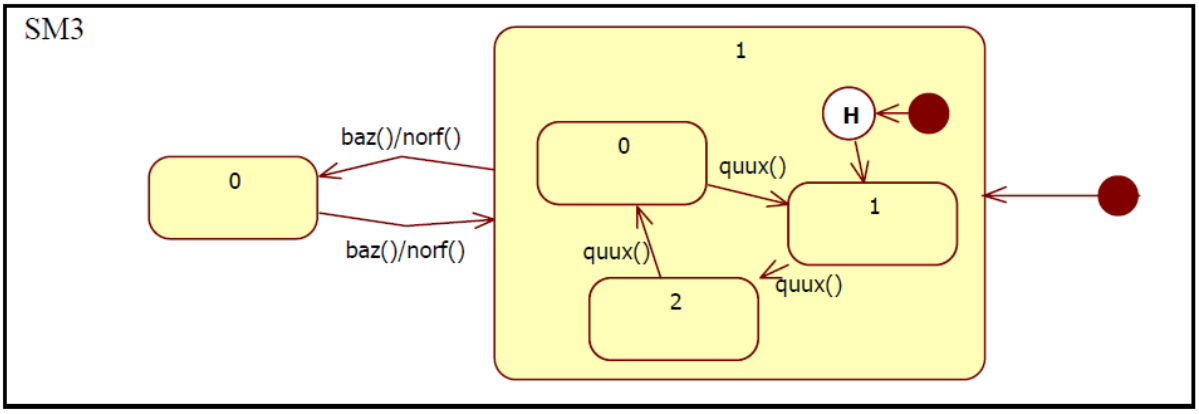
A kezdés után az **x, r, w, r** esemény-szekvencia hatására:

IGAZ	HAMIS	Állítás
x		Kétszer lefut a „t” tevékenység. *A 2 „r” esemény hatására.*
	x	Q állapotba kerülünk. *R állapotba kerülünk.*
x		Összesen 6 tevékenység fut le. *a,f,t,q,c,t → 6db.*

2015.01.13 – 8. Feladat

Rajzoljon UML2 állapotgépet az alábbi Java osztályhoz! (6 pont)

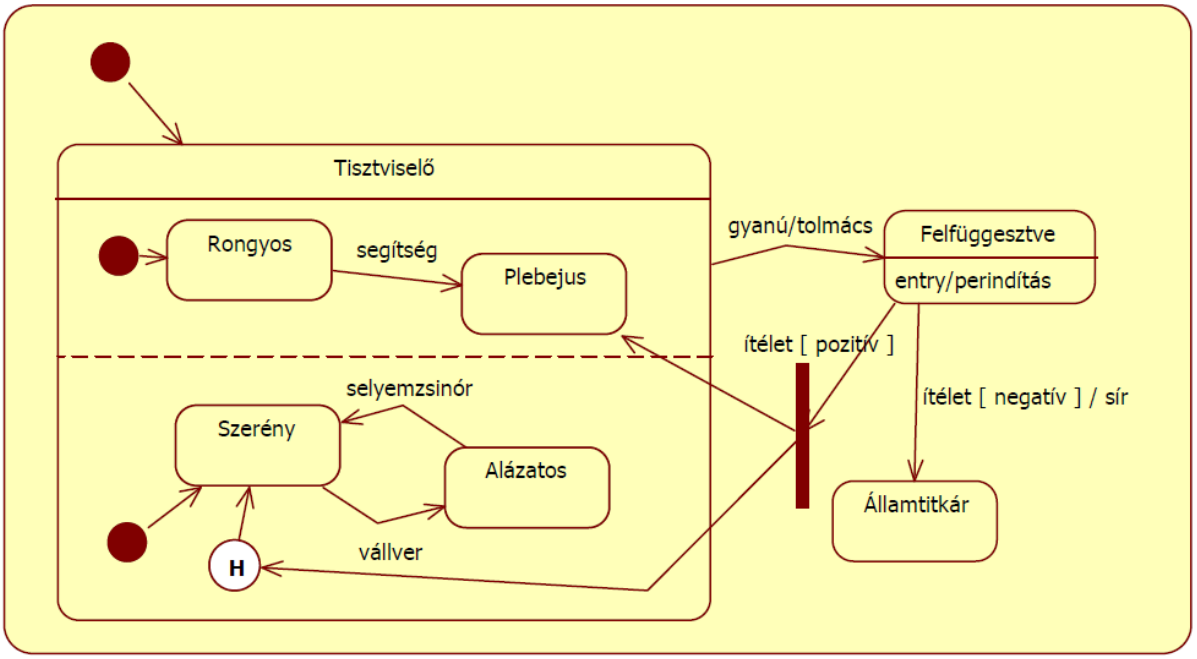
```
public class SM3 {
   private int x = 1;
   private int y = 1;
   private static final int p[] = { 1, 0 };

   private int s() { return (x+y)%3; }
   public void baz() { y = p[y]; norf(); }
   public void quux() { if (y == 1) x = s(); }
   public void norf() {}
}
```



2015.01.20 – 5. Feladat

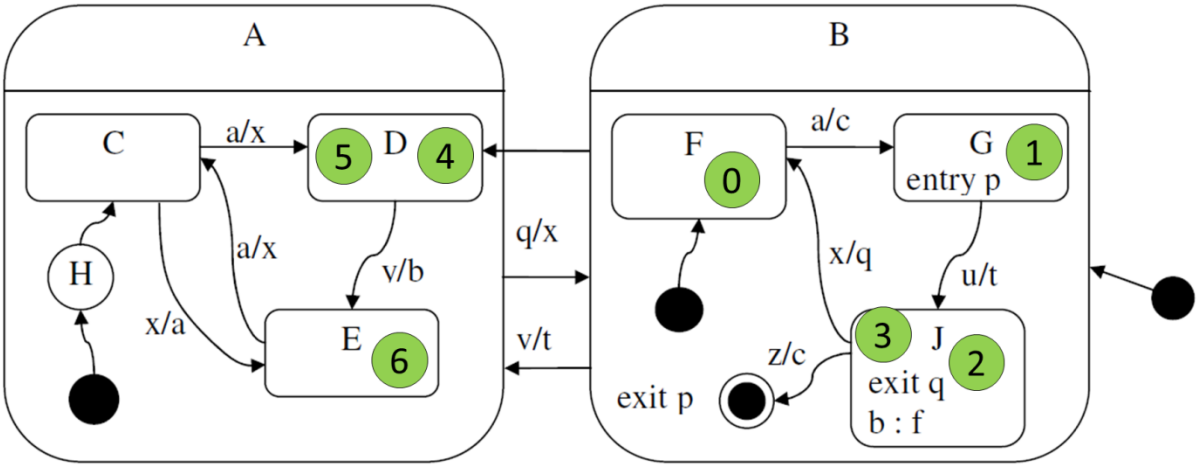
Készítsen UML2 állapotdiagramot (statechart) az alábbi történet alapján! (8 pont)

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.



2015.01.20 – 6. Feladat

A következő UML2 állapotdiagram alapján minősítse az állításokat! (6 pont)



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

IGAZ	HAMIS	Állítás
X		összesen 5 különböző állapotot érintünk
	X	„b” tevékenységet nem végzünk *v esemény hatására végzünk*
	X	C lesz a végállapot
X		van olyan esemény, amelynek hatására nem végzünk tevékenységet *A 2. 'a' esemény hatására nincs hova mennünk (maradunk D-ben), így ott nem végzünk tevékenységet.*
	X	az E állapotot kétszer érintjük
	X	egyszer lefut az „a” tevékenység *Egyszer se fut le.*

asd

asd

asd