VIZSGA FELADATSOR

SZOFTVERTECHNOLÓGIA

c. tárgyból

2013. június 11.

Az első lapon található feladatok megoldására 30 perc áll rendelkezésére. Az elérhető 24 pontból minimum 14 pontot kell kapnia ahhoz, hogy a második lapon szereplő feladatokra adott megoldásait értékeljük.

A tesztkérdésekre adott rossz válasz esetében pontot veszít, de feladatonként a total pontszám >= 0

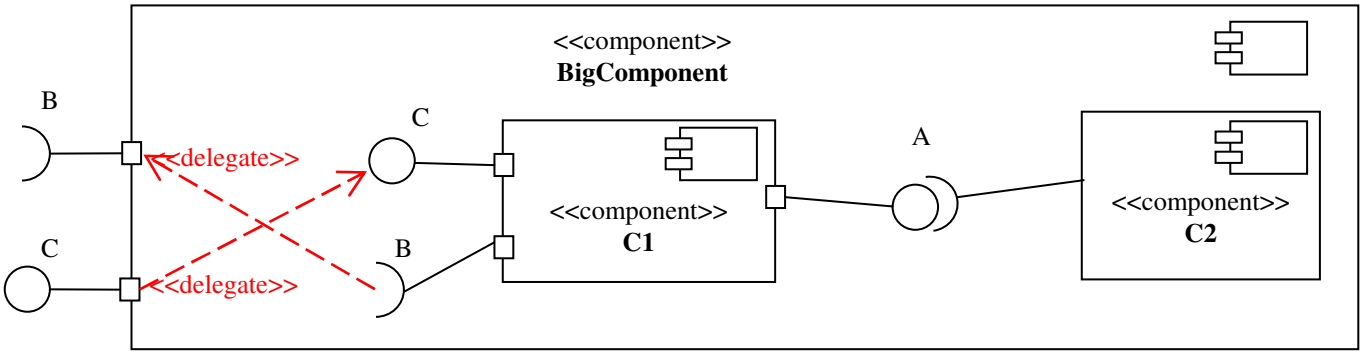
1. Készítse el azt az állapottáblát, amely megfelel az alábbi DTD-vel definiált adatszerkezetnek ! A tábla az 1. állapotban kezdődik, és segítségül megadtunk két blokkot. Kötőjellel (-) jelölje, ha egy adatelem, egy állapotban nem fogadható el! (5 pont)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE x [
  <!ELEMENT x    ((a,b+)|(c,b?))*>
  <!ELEMENT a    (#PCDATA)>
  <!ELEMENT b    (#PCDATA)>
  <!ELEMENT c    (#PCDATA)>
]>
```

	a	b	c
1	2	–	3
2	–	4	–
3	2	1	3
4	2	4	3

---

2. Rajzolja be alábbi UML2 diagramba a hiányzó kapcsolatokat! (3 pont)



---

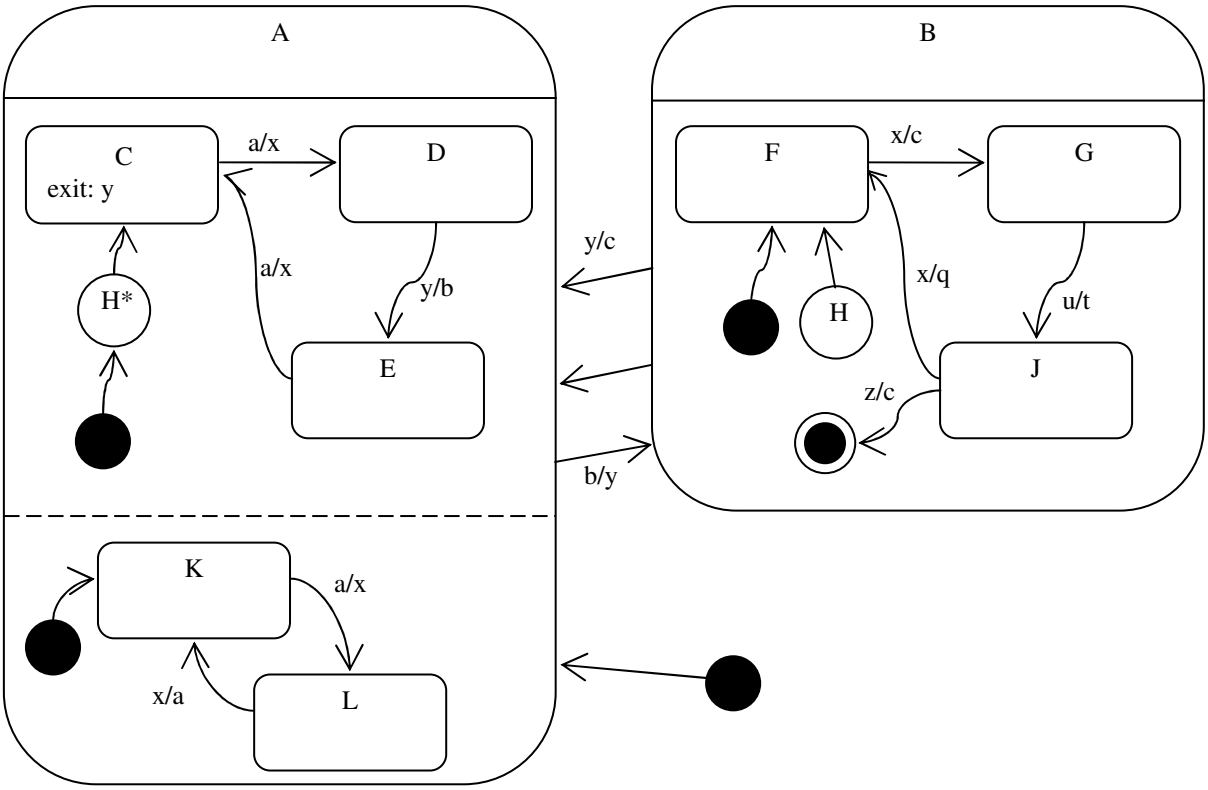
3. Töltse ki a táblázatot a Java gyűjtemény-keretrendszer osztály és interfészneveivel, amelyre igazak a táblázat peremén található állítások! Egy dobozba egy interfész és egy (az interfészt megvalósító) osztály nevét írja be! (3 pont)

	Elemek egyediek (unique)	**Elemek nem egyediek (non unique)**
Elemek rendezettek (ordered)	**SortedSet, TreeSet**	**List, ArrayList, Vector, LinkedList**
Elemek nem rendezettek (unordered)	**Set, HashSet**	**---**

Húzza alá azt a metódust, amelyik a fenti osztályok példányain meghívható! (1pont)

join, sleep, wait, interrupt, notify

4. A következő UML állapotdiagram alapján minősítse az állításokat! Csak a rubrikába tett jelzést vesszük figyelembe! (8 pont)



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

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

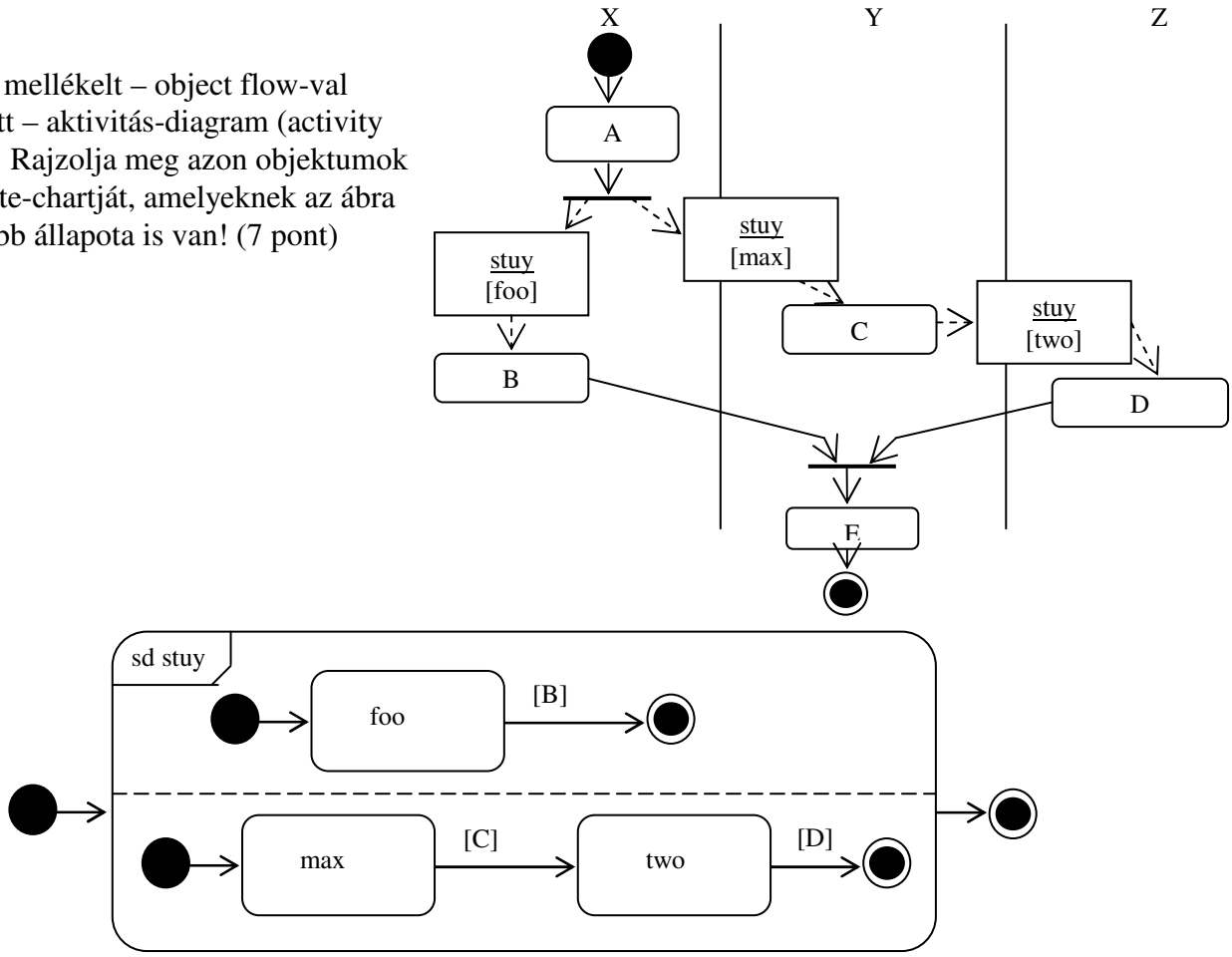
Igaz	Hamis	Állítás
☐	☐	pontosan kétszer fut le az 'x' tevékenység.
☐	☐	a végén G állapotba kerülünk.
☐	☐	pontosan kétszer fut le a 'c' tevékenység.
☐	☐	pontosan kétszer érintettük az L állapotot.

---

5. Jelölje az állítások mellett 1-5-ig, hogy minimálisan melyik CMM szinttől igazak ! Ha az állítás nem értelmezhető, akkor tegyen X-et ! (4 pont)

2	Fejlett projekt-menedzsment technikákat és eszközöket alkalmaznak a napi gyakorlatban.
X	Koreográfia (choreography) elvén szervezik a technológiai folyamatokat.
X	A projektekben rendszeresen alkalmazzák az agilis programozást (pl. Scrum).
X	A vezetők képesek a termékek minőségének közvetlen, számszerű ellenőrzésére.

6. 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! (7 pont)



---

7. Jellemezzünk egy stringet az alábbi műveletekkel!

END(s, x)	az **s** string végére rakja az **x** karaktert.
CHR(s, x)	az **s** stringben található **x** karakterek előfordulásának számát adja.
LGTH(s)	az **s** string karaktereinek számát adja.
CRT( )	új (üres) stringet hoz létre.
TAIL(s)	az **s** string legrégebbi karakterének levágása után maradó stringet adja.

Az alábbi kifejezésekhez adja meg, hogy azok algebrai axiómák BAL oldalán állhatnak-e vagy sem! (5 pont)

Igen	**Nem**		**Igen**	**Nem**	
☐	☐	CHR(TAIL(s), x)	☐	☐	END(CRT(), x)
☐	☐	TAIL(CHR(s, x))	☐	☐	LGTH(TAIL(s))
☐	☐	CHR(END(s, x), x)	☐	☐	CHR(CRT(), 0)
☐	☐	CHR(END(s, y), x)	☐	☐	LGTH(CHR(s, x))
☐	☐	TAIL(CHR(s, x))	☐	☐	LGTH(TAIL(s))
☐	☐	LGTH(END(CRT(), x))	☐	☐	END(s, LGTH(s))

---

8. Sorolja fel a tesztelés során végrehajtandó „kiértékelés” (evaluation) folyamatnak a be és kimeneteit! (4 pont)

Bemenetek: test results, expected results.....

Kimenetek: errors, error statistics.....

9. Jelölje be az alábbi táblán, hogy az egyes szerződéses feltételek megszegése esetén melyik fél a hibás! (2 pont)

	kliens	szerver
invariáns (invariant)		X
utófeltétel (postcondition)		X

---

10. Adott az alábbi **Student** generikus Java osztály, amelyet tárgy (C) típussal lehet paraméterezni.

```
public class Student<C> implements Cloneable {
    private String name; // név
    private String neptun; // neptunkód
    private Set<C> courses; // felvett tárgyak
    public Student(String na, String ne) {
        name = na;
        neptun = ne;
        courses = new HashSet<C>();
    }
    public void setName(String s) { name = s; }
    public void addCourse(C c) { courses.add(c); }
    public void delCourse(C c) { courses.remove(c); }}
```

Implementálja meg Javaban a fenti **Student** osztály következő metódusait! Nem használható ciklus és a paraméterek állapotát nem módosíthatja!

```
public boolean equals(Object o) {
//Két hallgató akkor egyezik, ha a nevük és a neptunkódjuk is azonos. Feltételezheti,
//hogy a metódus csak Student dinamikus típusú objektumot kap paraméterül. (2 pont)
```

Student<C> s = (Student<C>)o;
return (s.name.equals(name)&&s.neptun.equals(neptun));

```
}
public Object clone() {
//visszatér egy olyan Student példánnyal, amelynek tartalma azonos az eredetivel, de a
//set, add és del kezdetű metódusok hívásakor csak a hívott objektum tartalma változik
//(deep clone). (3 pont)
```

Student<C> s = new Student<C>(name, neptun);
s.courses = new HashSet<C>(courses);
return s;

```
}
static <C> int numOfSharedCourses(Student<C> s1,
                                  Student<C> s2) {
//visszatérési értéke a két hallgató közös tárgyainak darabszáma. (3 pont)
```

HashSet<C> hs = new HashSet<C>(s1.courses);
hs.retainAll(s2.courses);
return hs.size();

```
}
```

Eredmények értékelése:

Pontszám	Osztályzat
21 -	2
28 -	3
35 -	4
42 -	5