VIZSGA FELADATSOR

SZOFTVERTECHNOLÓGIA

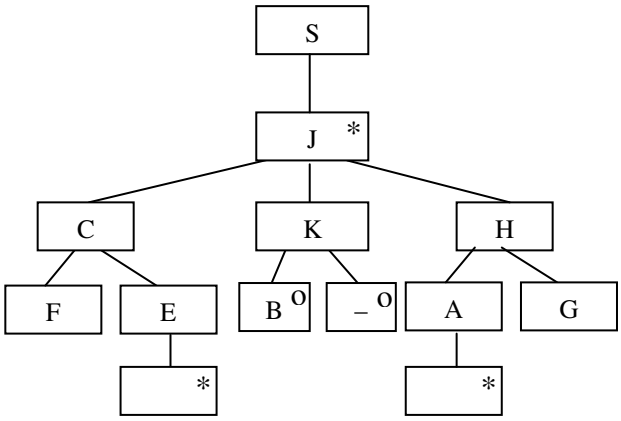
c. tárgyból

2013. június 18.

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. Egészítse ki az alábbi ELH-t és állapottáblát úgy, hogy mindkettő ugyanazt a szerkezetet írja le! A kiegészítés során az **ábrák szerkezetén**, az azokba **beírt, berajzolt elemeken változtani tilos**! Az ELH-ban csak a két üresen maradt blokkot kell kitölteni! Az állapottáblát egészítse ki! Az induló állapot legyen az ① ! (ELH 2 pont, állapottábla 3 pont)



	F		B	D
①	③			
②				
③	③			

---

2. Mire használjuk a JUnit-ban a ***@AfterClass*** annotációval megjelölt metódust? (2 pont)

Mit történt, ha a JUnit teszt eredménye ***error*** ? (2 pont)

---

3. A szabványos Java nyelvben az alábbiak közül mely hívások hatására hagyja el biztosan a szál a futó állapotát? (4 pont)

- ☐ yield()
- ☐ wait()
- ☐ notify()
- ☐ notifyAll()
- ☐ sleep(1000)
- ☐ stop()
- ☐ egy másik élő szálra történő join() hívás
- ☐ Thread.killThread()

---

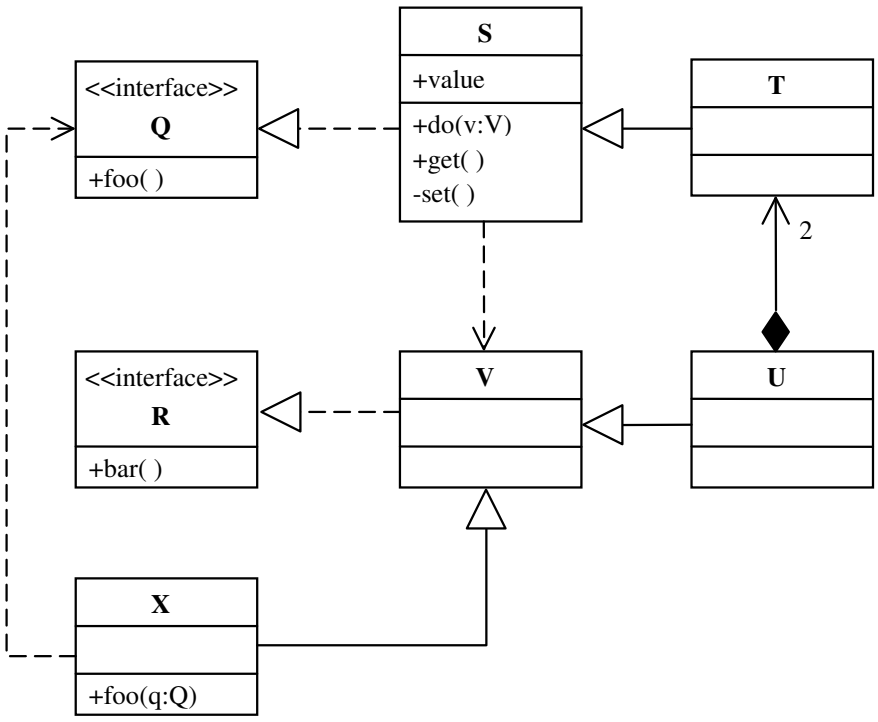
4. Jelölje (karikázza be) az állítások igazságtartalmát, ha feltesszük, hogy szabványos Java nyelvet használunk! (3 pont)

I H egy szál egyszerre csak egy objektum monitorában tartózkodhat.

I H szálak nem képesek saját magukat közvetlenül *waiting* állapotból notify-jal felébreszteni.

I H egy változó statikus típusa nem lehet a változó dinamikus típusának leszármazottja.

5. Az alábbi UML2 diagram alapján - a kulcs felhasználásával - jellemezze az állításokat ! (8 pont)



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 (- -)

[ ] **T** létrehozhat **U** osztályú objektumot, mert **S** létrehozhat **V**-t és **T** az **S**-nek, **U** a **V**-nek leszármazottja.

[ ] **X foo(q:Q)** metódusa kaphat paraméterül **T**-t, mert **T**-nek is van **foo()** metódusa.

[ ] **X foo(q:Q)** metódusa meghívhatja a paraméterül kapott **S get()** metódusát, mert **S** megvalósítja a **Q** interfészt.

[ ] **V** törlésekor törölni kell két **T**-t is, mert egy **U**-nak két **T** komponense van és **U** a **V** leszármazottja.

[ ] **T** nem függ **U**-tól, mert **T** nem függ **V**-től sem.

[ ] **X** meghívhatja egy **Q** interfészű objektum **foo( )** metódusát, mert **X** implementálja **Q**-t.

[ ] **X bar()** metódusából meghívhatjuk egy **Q** interfészű objektum **foo( )** metódusát, mert **X foo(q:Q)** metódusából is hívhatjuk egy **Q** interfészű objektum **foo( )** metódusát.

[ ] **S set( )** metódusa nem módosíthatja a **value** attribútumot, mert a láthatóságuk különböző.

6. A kockázat tervezése során milyen stratégiákat választunk ? (3 pont)

---

7. Mit jelent a CMM ? (3 pont)

Jelölje meg az igaz állításokat ! (4 pont)

- ☐ a CMM egy szervezet által készített összes szoftver minőségét értékeli.
- ☐ a CMM szint kifejezi a szervezet vezetésének minőségét is.
- ☐ CMM minősítést csak jó szoftverek kaphatnak.
- ☐ a CMM egy adott szoftver termék fejlettségét, érettségét vizsgálja.
- ☐ a CMM egy szervezetben zajló szoftver fejlesztési folyamatot értékeli.
- ☐ egy jó szoftvertől elvárható a magas CMM szint.

---

8. Rajzoljon UML 2 osztálydiagramot az alábbi Java kódrészlet alapján! (6 pont)

```
class A {
  private static int x = 0;
  public int bar() { return ++x; }
  protected A baz() {
              return new A(); }}

class E extends Exception {
  private A a;
  public E(A a) { this.a = a; }
  public A getA() { return a; } }
```

```
interface B {
  void foo(A a) throws E; }

class C extends A implements B {
  boolean flag;
  public void foo(A a) throws E {
    flag = !flag;
    if(a.bar()>5&&flag) throw new E(baz());
  }
  protected A baz() { return new C(); }}
```

9. Rajzoljon UML2 állapot diagramot! (8 pont)
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.

---

10. Mi a “fan-out”? (2 pont)

- ☐ egy adott modul (osztály) döntési hatáskörébe tartozó modulok (osztályok) száma
- ☐ egy adott modulban (osztályban) használt paraméterek száma
- ☐ egy adott modul (osztály) vezérlési hatáskörébe tartozó modulok (osztályok) száma
- ☐ egy adott modul (osztály) által használt modulok (osztályok) száma
- ☐ egy adott modult (osztályt) használó más modulok (osztályok) száma

---

Eredmények értékelése:

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