Základy objektovo-orientovaného programovania **A**
Ing. Ján Lang, PhD., UISI FIIT STU
Test - 13. novembra 2014

Priezvisko:
Meno:

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Test trvá 35 minút. V uzavretých otázkach s ponúknutými odpoveďami je vždy správna iba jedna možnosť. Do tabuľky uveďte písmeno pod ktorým je označená odpoveď, ktorú vyberáte. Hodnotia sa len odpovede v tabuľke. V prípade opravy jasne vyznačte odpoveď, ktorá platí. Každá správna odpoveď má hodnotu vyznačenú v otázke. Nesprávna odpoveď, alebo nejednoznačné vyznačenie má hodnotu 0 bodov. Postup riešenia sa nehodnotí. Akceptovaný bude len odovzdaný celistvý list.

1. (2b) Daný je nasledujúci kód v Jave:

```
public class Prva {
	public Prva() {
		System.out.print("P");
	}
}
public class Druha extends Prva{
	public Druha() {
		System.out.print("D");
	}
}
public class Tretia extends Prva {
	public Tretia() {
		System.out.print("T");
	}
}
public class Stvrta extends Druha {
	public Stvrta() {
		System.out.print("S");
	}
}
```

Čo sa vypíše po vykonaní príkazov:

```
new Tretia();
new Druha();
new Stvrta();
new Prva();
```

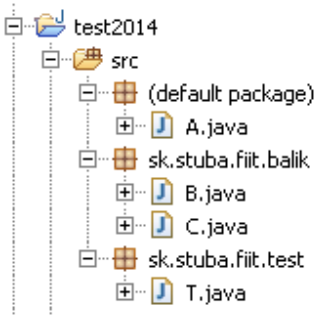
(a) PPDSPDPT
(b) PTPDPDSP
(c) PDSPPTPD
(d) PTPDSPPD
(e) PDPTPDSP
(f) PPDPTPDS

2. (1b) Trieda A v zmysle obr. 1
(a) Môže byť importovaná triedou B, C a T
(b) Môže byť importovaná iba triedou B
(c) Môže byť importovaná iba triedou C
(d) Môže byť importovaná iba triedou T
(e) Môže byť importovaná triedou ktorá ju rozširuje
(f) nemôže byť importovaná

3. (1b) Ktoré z nasledovných tvrdení je pravdivé?

Implicitný konštruktor v Jave:
(a) musí mať vždy argumenty
(b) nemôže byť preťažený
(c) má návratovú hodnotu
(d) sa nezachová vytvorením explicitne uvedeného konštruktora
(e) je priamo použiteľný na tvorbu klonov existujúcich inštancií
(f) neexistuje

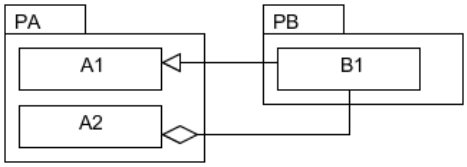
4. (2b) Daný je projekt test2014 v Jave ako vidno na obr. 1:



Obr. 1 (Obrázok pre úlohy č. 2, 4 a 10)

Atribút triedy `T` bude v triede `B` viditeľný v prípade, že:
(a) daný atribút bude `protected` a triede T sprístupníme menný priestor sk.stuba.fiit.balik a zároveň trieda B bude rozširovať triedu T
(b) daný atribút bude `private` a triede T sprístupníme menný priestor sk.stuba.fiit.test a zároveň trieda T bude rozširovať triedu B
(c) daný atribút bude `private` a triede T sprístupníme menný priestor sk.stuba.fiit.balik a zároveň trieda B bude rozširovať triedu T
(d) daný atribút bude `protected` a triede B sprístupníme menný priestor sk.stuba.fiit.test a zároveň trieda B bude rozširovať triedu T
(e) daný atribút bude `protected` a triede T sprístupníme menný priestor sk.stuba.fiit.test a zároveň trieda T bude rozširovať triedu B
(f) daný atribút bude `protected` a triede B sprístupníme menný priestor sk.stuba.fiit.balik a zároveň trieda T bude rozširovať triedu B

5. (2b) Daný je nasledujúci diagram v jazyku UML:



Obr.2 (Obrázok pre úlohy č. 5 a 6)

Znázornený vzťah na úrovni implementácie v jazyku Java znamená, že:
(a) trieda B1 dedí vlastnosti triedy A1 a trieda A2 agreguje inštanciu triedy B1
(b) trieda A1 dedí vlastnosti triedy B1 a trieda A2 agreguje inštanciu triedy B1
(c) trieda B1 dedí vlastnosti triedy A1 a trieda B1 agreguje inštanciu triedy A2
(d) trieda B1 agreguje vlastnosti triedy A1 a trieda A2 dedí inštanciu triedy B1
(e) trieda A1 agreguje vlastnosti triedy B1 a trieda A2 dedí inštanciu triedy B1
(f) trieda A1 agreguje vlastnosti triedy B1 a trieda A2 agreguje inštanciu triedy B1

6. (1b) Daný je nasledujúci kód v Jave:

```
public static B1 m() {
      return new B1();
}
```

K statickej metóde `m()` triedy `B1` v zmysle obr.2 nemôžeme z triedy A2 pristupovať:
(a) `B1 b = new B1().m();`
(b) `B1.m();`
(c) `new B1().m();`
(d) `new B1.m();`
(e) `B1 bb = new B1(); bb.m();`

7. (1b) Daný je nasledujúci kód v Jave:

```
public class Clovek {
      void Clovek(int i) {    }
      void Clovek(int i, int j) {   }
      void m(int i) {   }
      void m(int i, int j) {  }
}

public class Obcan extends Clovek {
      void Obcan(int i) {     }
      void Obcan(int i, int j) {    }
      void n(int i) {   }
      void m(int i, int j) {  }
}
```

Uvedená konštrukcia predstavuje:
(a) hierarchiu dedenia
(b) hierarchiu agregácie
(c) nepredstavuje hierarchiu
(d) hierarchiu kompozície
(e) hierarchiu zapuzdrenia

8. (2b) V zmysle Java kódu uvedeného v úlohe č. 7:

(a) metóda `n(int i)` triedy `Obcan` preťažuje metódu `m(int i)` triedy `Clovek`
(b) metóda `n(int i)` triedy `Obcan` prekonáva metódu `m(int i)` triedy `Clovek`
(c) metóda `n(int i)` triedy `Obcan` preťažuje metódu `m(int i, int j)` triedy `Clovek`
(d) metóda `n(int i)` triedy `Obcan` prekonáva metódu `m(int i, int j)` triedy `Clovek`
(e) žiadne z uvedeného

9. (1b) V zmysle Java kódu uvedeného v úlohe č. 7 volanie konštruktora `Clovek(int i)` z konštruktora `Obcan(int i, int j)`

(a) je možné príkazom `super.Clovek(int i);`
(b) je možné príkazom `super.Obcan(i);`
(c) je možné príkazom `super.Clovek(i);`
(d) je možné príkazom `super.Obcan(int i);`
(e) je možné príkazom `super.Obcan(int i, j);`
(f) nieje možné

10. (2b) Daný je nasledujúci kód v Jave a rozloženie tried do balíkov v zmysle obr.1:

//...súbor C.java
```
public class C {
      private int ci=1;
      protected int cj=2;
      public int ck=3;
      int cl=4;
}
```

//…súbor A.java
```
import sk.stuba.fiit.balik.C;
public class A extends C {
  public int testDostupnosti() {
      ...todo...*
  }
}
```

*Korektný prístup k hodnote atribútu `cj` triedy `C` z metódy `testDostupnosti()` triedy `A` je:

```
(a) System.out.println(new T().cj);
(b) System.out.println(new C().cj);
(c) System.out.println(A().cj);
(d) System.out.println(C().cj);
(e) System.out.println(cj);
```
(f) žiaden z uvedených

Spolu 15 bodov
Riešenie:

1	b
2	f
3	d
4	d
5	a
6	d
7	a
8	e
9	c
10	e