Objektovo-orientované programovanie **A**

Ing. Valentino Vranić, PhD., ÚISI FIIT STU

Semestrálny test — 4. apríl 2008

Priezvisko:		1b	
Meno:		2b	

	a	b	c	d	e
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					

Test trvá 50 minút.

V otázkach je len jedna možnosť správna. Vyznačte svoju odpoveď krížikom do veľkej tabuľky (malú tabuľku nevypĺňajte). Hodnotia sa len odpovede vyznačené v tabuľke.

V prípade opravy jasne vyznačte odpoveď ktorú vyberáte. Každá správna odpoveď má hodnotu vyznačenú v otázke. Nesprávna odpoveď, vyznačenie viac odpovedí alebo nejednoznačné vyznačenie má hodnotu 0 bodov. Postup riešenia sa nehodnotí.

1. (1 b) Objekt v objektovo-orientovanom programovaní predstavuje

(a) typ
(b) modul
(c) inštanciu triedy
(d) inštanciu triedy alebo rozhrania
(e) triedu

2. (1 b) Príkaz

import java.util.∗;

(a) fyzicky pripojí typy balíka java.util k programu bez typov v podbalíkoch
(b) fyzicky pripojí len skutočne použité typy balíka java.util k programu bez typov v podbalíkoch
(c) fyzicky pripojí všetky typy balíka java.util k programu vrátane typov v podbalíkoch
(d) sprístupní priestor názvov všetkých typov balíka java.util, ale bez typov v podbalíkoch
(e) sprístupní priestor názvov všetkých typov balíka java.util vrátane typov v podbalíkoch

3. (1 b) V triede, ktorá implementuje rozhranie, je možné zadefinovať

(a) len ďalšie polia
(b) ľubovoľné ďalšie polia a metódy
(c) len ďalšie metódy
(d) len metódy deklarované v rozhraní
(e) len metódy nedeklarované v rozhraní

4. (2 b) Daný je kód v Jave na obr. 1. Čo sa vypíše po vykonaní týchto príkazov:

```
A b = new B();
A c = new C();

b.m();
c.m();
((B)c).m();
```

(a) aab
(b) bcc
(c) bcb
(d) aaa
(e) aac

```
class A {
   public void m() { System.out.print("a"); }
}

class B extends A {
   public void m() { System.out.print("b"); }
}

class C extends B {
   public void m() { System.out.print("c"); }
}
```

Obrázok 1: Kód pre otázky 4 a 5.

5. (1 b) Daný je kód v Jave na obr. 1. Dá sa z metódy m() triedy C zavolať rovnomenná metóda triedy A (bez vytvárania ďalšieho objektu)?

(a) áno, príkazom super.m();
(b) áno, príkazom super.super.m();
(c) nie
(d) áno, príkazom A.m();
(e) áno, príkazom super.this.m();

6. (1 b) Zapuzdrenie v objektovo-orientovanom programovaní

(a) umožňuje znížiť závislosť klientskeho kódu
(b) predstavuje spôsob tvorenia hierarchie
(c) umožňuje spájanie objektov
(d) umožňuje, aby sa objekt uplatnil namiesto objektu jeho nadtypu
(e) predstavuje kritérium pre použitie agregácie

7. (1 b) Prístup **protected** je vhodné použiť pri takých prvkoch triedy, ku ktorým chceme pristupovať len

(a) v odvodených triedach toho istého balíka
(b) v danej triede
(c) v triedach toho istého balíka
(d) v odvodených triedach
(e) v odvodených triedach a v triedach toho istého balíka

8. (2 b) Daný je nasledujúci kód:

```
class MyException extends Exception {}

class A {
    void a() throws MyException {
        if (...) {
            ...
        }
        else
            throw new MyException();
    }
}

class B {
    void b() throws MyException {
        new A().a();
    }
}
```

Metóda b() triedy B

(a) musí zachytávať výnimku typu MyException
(b) nesmie obsahovať klauzulu **throws**
(c) musí ošetrovať výnimku typu MyException
(d) je korektná
(e) musí vyhadzovať výnimku typu MyException

9. (1 b) Dá sa urobiť inštancia abstraktnej triedy?

(a) áno, ako hociktorej inej triedy
(b) nie
(c) áno, ale len ak trieda neobsahuje abstraktné metódy
(d) áno, ale nebudú sa dať zavolať abstraktné metódy
(e) áno, ale bude abstraktná

10. (1 b) V triede, ktorá dedí od rozhrania je možné zadefinovať

(a) len ďalšie polia
(b) len konkrétne metódy
(c) len metódy ktoré predpisuje rozhranie
(d) len to čo predpisuje rozhranie
(e) aj iné metódy než predpisuje rozhranie

11. (1 b) Daný je nasledujúci kód:

```
if (o.class == "A")
    ((A)o).ma();
else if (o.class == "B")
    ((B)o).mb();
else
    ...
}
```

Tento kód porušuje

(a) princíp otvorenosti a uzavretosti
(b) princíp zapuzdrenia
(c) Liskovej princíp substitúcie
(d) pravidlá dedenia
(e) pravidlá polymorfizmu

12. (2 b) Čím treba nahradiť označené časti kódu z obr. 2, aby metóda add() pripájala ďalší prvok so zadaným údajom data k aktuálnemu prvku pre rôzne typy údajov, ako ukazuje metóda main().

(a) `**1**`: `**2**`: T d `**3**`: El(d)
(b) `**1**`: <data> `**2**`: data d `**3**`: El<data>(d)
(c) `**1**`: <T> `**2**`: T d `**3**`: El<T>(d)
(d) `**1**`: <T> `**2**`: <T> d `**3**`: El(<T> d)
(e) `**1**`: `**2**`: d `**3**`: El<d>(d)

```
public class El<T> {
    private T data;
    private El**1** next;

    public El(**2**) {
        data = d;
    }
    public void add(**2**) {
        next = new **3**;
    }

    public static void main(String[] args) {
        El<Double> e1 = new El<Double>(8.4);
        e1.add(10.2);

        El<Integer> e2 = new El<Integer>(10);
        e1.add(12);
    }
}
```

Obrázok 2: Kód pre otázku 12.

Objektovo-orientované programovanie **A**
Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Semestrálny test — 4. apríl 2008

15 b

1 c

2 d

3 b

4 b

5 c

6 a

7 e

8 d

9 b

10 e

11 a

12 c