Objektovo-orientované programovanie **A**

Ing. Valentino Vranić, PhD., ÚISI FIIT STU

Semestrálny test — 9. apríl 2010

Priezvisko:

Meno:

1b	
2b	

Test trvá 45 minút.

V otázkach je len jedna možnosť správna. Vyznačte svoju odpoveď krížikom do tabuľky. Hodnotia sa len odpovede v tabuľke.

V prípade opravy jasne vyznačte odpoveď, ktorá platí. Každá správna odpoveď má hodnotu vyznačenú v otázke. Nesprávna odpoveď, vyznačenie viac odpovedí alebo nejednoznačné vyznačenie má hodnotu 0 bodov. Postup riešenia sa nehodnotí. Len celistvý list bude akceptovaný.

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

1. (1 b) Daná je trieda

```
public class A {
   private int i;
   public void m(int i) { this.i = i; }
}
```

Daný je objekt `o` triedy `A`. Príkaz

```
o.m(5);
```

(a) mení stav objektu
(b) mení správanie objektu
(c) mení stav a správanie objektu
(d) mení identitu objektu
(e) inicializuje objekt

2. (1 b) Zapuzdrenie v objektovo-orientovanom programovaní

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

3. (2 b) K triedam z obr. 1 je daný nasledujúci kód:

```
List<A> list = new ArrayList<A>();
list.add(new B());
list.get(0).m();
```

Tento kód

(a) po vykonaní vypíše `a`
(b) sa vôbec nepreloží, lebo typ referencie `list` nezodpovedá typu priradeného objektu
(c) po vykonaní vypíše `b`
(d) sa vôbec nepreloží, lebo do zoznamu `list` sa dajú vkladať len objekty typu `A`
(e) padne počas vykonávania s výnimkou `ClassCastException`

4. (1 b) V triede, ktorá dedí od rozhrania je možné zadefinovať

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

5. (1 b) Princíp otvorenosti a uzavretosti hovorí, že

(a) kód má byť otvorený pre rozšírenie, ale uzavretý pre zmeny
(b) kód má byť otvorený pre zmeny, ale uzavretý pre rozšírenie
(c) kód je počas úpravy otvorený, a po úprave uzavretý
(d) kód má byť zároveň aj otvorený, aj uzavretý pre úpravy
(e) súbor s kódom treba po otvorení aj zavrieť

6. (2 b) Daný je kód v Jave na obr. 1. Čo sa vypíše po vykonaní týchto príkazov:

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

M.r(a);
M.r(b);
M.r(c);
M.r((A)a);
M.r((A)b);
M.r((A)c);
```

(a) abcaaa
(b) aaaaaa
(c) abaaba
(d) abcaba
(e) abcabc

```
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"); }
}
class M {
   public static void r(A o) { o.m(); }
}
```

Obr. 1: Kód pre otázky 3 a 6.

7. (1 b) Agregácia v objektovo-orientovanom programovaní

(a) stanovuje kritéria pre použitie abstraktných tried
(b) znamená skrytie implementácie objektu
(c) znamená spájanie objektov do väčších celkov
(d) predstavuje kritérium pre použitie dedenia
(e) umožňuje, aby sa objekt uplatnil namiesto objektu jeho nadtypu

8. (1 b) Daný je nasledujúci kód:

```
for (Object o : l) {
   if (o.class == "Frog")
      ((Frog)o).jump();
   else if (o.class == "Bird")
      ((Bird)o).fly();
   else
      ;
}
```

Tento kód porušuje

(a) princíp abstrakcie
(b) princíp otvorenosti a uzavretosti kódu
(c) Liskovej princíp substitúcie
(d) princíp generalizácie a špecializácie
(e) princíp polymorfizmu

9. (1 b) Daná je trieda:

```
public abstract class T {
   abstract void a();
   void b() { ... };
}
```

Inštancia tejto triedy sa

(a) dá vytvoriť
(b) nedá vytvoriť, lebo metóda `a()` je abstraktná
(c) dá vytvoriť, ale nebude sa dať zavolať metóda `a()`
(d) dá vytvoriť, ale bude ukazovať na **null**
(e) nedá vytvoriť

10. (1 b) Rozšírená slučka **for** v Jave pri aplikácii na zoskupenie

(a) potrebuje explicitné zadanie iterátora zoskupenia
(b) automaticky získa inkrementátor zoskupenia
(c) potrebuje explicitné zadanie inkrementátora zoskupenia
(d) automaticky získa iterátor zoskupenia
(e) pracuje cez iteračnú premennú

11. (2 b) Daný je nasledujúci kód:

```
class MyException extends Exception {}

class A {
   void a(int i) throws MyException {
      if (i > 0) {
         i = i / 2;
      }
      else
         throw new MyException();
   }
}

class B {
   void b(int i) {
      try {
         new A().a(i);
      } catch(MyException e) {
         System.out.println("chyba");
      }
   }
}
```

Metóda `b()` triedy `B`

(a) musí v bloku **finally** vyhadzovať výnimku typu `MyException`
(b) je korektná
(c) musí obsahovať klauzulu **throws**
(d) musí v bloku **catch** vyhadzovať výnimku typu `MyException`
(e) musí v bloku **try** vyhadzovať výnimku typu `MyException`

12. (1 b) Návrhový vzor Strategy je vhodné použiť na

(a) vysporiadanie sa s veľkým počtom objektov
(b) realizáciu viacnásobného polymorfizmu
(c) implementáciu variability v uskutočňovaní určitého procesu
(d) zabezpečenie viacnásobného dedenia
(e) dodržanie Liskovej princípu substitúcie

Objektovo-orientované programovanie **A**
Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Semestrálny test — 9. apríl 2010

15 b

1 a

2 b

3 c

4 d

5 a

6 e

7 c

8 b

9 e

10 d

11 b

12 c