**Objektovo-orientované programovanie** **B**

Ing. Valentino Vranić, PhD., ÚISI FIIT STU

Semestrálny test — 9. apríl 2010

**Priezvisko:**

**Meno:**

| | |
|---|---|
| 1b | |
| 2b | |

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

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.

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

**2. (1 b)** Návrhový vzor Strategy je vhodné použiť na

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

**3. (1 b)** Agregácia v objektovo-orientovanom programovaní

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

**4. (1 b)** Daná je trieda:

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

Inštancia tejto triedy sa

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

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

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

**7. (1 b)** V triede, ktorá dedí od rozhrania je možné zadefinovať

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

**8. (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

**9. (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) aaaaaa
(b) abcabc
(c) abaaba
(d) abcaba
(e) abcaaa

```
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 9 a 12.

**10. (1 b)** Princíp otvorenosti a uzavretosti hovorí, že

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

**11. (1 b)** Rozšírená slučka **for** v Jave pri aplikácii na zoskupenie

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

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

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

15 b

**1** e

**2** d

**3** a

**4** a

**5** e

**6** d

**7** b

**8** a

**9** b

**10** c

**11** b

**12** a