Programovacie jazyky Ing. Valentino Vranić, PhD.
Test — 16. mája 2005 ÚISI FIIT STU

Priezvisko:	
Meno:	
Počet listov:	

Pri otázkach 1–6 je len jedna možnosť správna. Vyznačte svoju odpoveď krížikom do tabuľky uvedenej nižšie. V prípade opravy, jasne vyznačte ktorú odpoveď vyberáte. Nesprávna odpoveď, vyznačenie viac odpovedí alebo nejednoznačné vyznačenie má hodnotu 0 bodov. Postup riešenia sa nehodnotí.

Pri otázkach 7 a 8 uveďte úplnú odpoveď podľa zadania.

Otázky 9 a 10 boli dopredu zverejnené. Odovzdajte predvypracovanú odpoveď pri overovaní identity. Ak ste odpoveď nevypracovali, môžete to ešte urobiť v priebehu testu.

Všetky listy ktoré odovzdávate (okrem tohto) očíslujte a označte svojim menom. Na vyššie vyznačené miesto uveďte počet odovzdaných listov.

	a	b	c	d	e
1					
2					
3					
4					
5					
6					

1. (5 b) Jeden z rozdielov medzi abstraktnou triedou a rozhraním v Jave je:

(a) Je možné vytvárať inštancie rozhraní, ale nie aj abstraktných tried.

(b) Je možné vytvárať inštancie abstraktných tried, ale nie aj rozhraní.

(c) Abstraktná trieda môže dediť od rozhrania, ale nie naopak.

(d) Rozhranie môže dediť od abstraktnej triedy, ale nie naopak.

(e) Konkrétne triedy môžu dediť od abstraktnej triedy, ale nie od rozhrania.

```
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 X {
   public void m() { System.out.print("X"); }
}
```

Obrázok 1: Kód pre otázky 2 a 7.

2. (5 b) Daný je kód v Jave z obr. 1. Čo sa vypíše po vykonaní týchto príkazov:

```
A a = new C();
a.m();
((B)a).m();
((A)a).m();
```

(a) ABC **(b)** AAA **(c)** CCC **(d)** CBA **(e)** ABA

3. (5 b) Daná je nasledujúca trieda:

```
public class Uzol {
   Uzol dalsi;
   String hodnota;
}
```

Pomocou triedy `Uzol` sa dá vytvoriť zreťazený zoznam. Pre použitie takého zoznamu stači uchovávať referenciu na jeho vrch.

Predpokladajme že je vytvorený takýto zoznam obsahujúci len jeden uzol, ktorý je teda vrchom zoznamu:

```
Uzol vrch = new Uzol();
```

Vytvoríme potom ďalší prvok:

```
Uzol p = new Uzol();
```

Ako pridáme prvok `p` na vrch zoznamu (tak aby premenná `vrch` naďalej ukazovala na vrch zoznamu)?

(a)
```
p = vrch;
vrch = p;
```

(b)
```
p.dalsi = vrch;
vrch = p;
```

(c)
```
p.dalsi = vrch.dalsi;
vrch = p.dalsi;
```

(d)
```
p = vrch.dalsi;
vrch = p.dalsi;
```

(e)
```
p = vrch.dalsi;
vrch = p;
```

4. (5 b) Daný je program v Jave pozostávajúci z dvoch balíkov. V jednom z nich je nasledujúci kód:

```
package balik1;

class Trieda1 {
   private int a;
   protected int b;
   String c;
}

public class Trieda2 extends Trieda1 {
   public String d;
   public void m1() {
      a = 1;      // 1
      b = 2;      // 2
      c = "a";    // 3
   }
}
```

Druhý balík obsahuje tento kód:

```
package balik2;
import balik1;

public class Trieda3 {
   public static void main(String[] args) {
      Trieda2 t = new Trieda2();

      t.b = 1;     // 4
      t.c = "a";   // 5
      t.d = "b";   // 6
   }
}
```

Všetky príkazy označené komentárom ktoré porušujú prístupnosť prvkov tried sú:

(a) *1*, *4* a *5* **(b)** *1* **(c)** *1*, *2*, *4* a *5* **(d)** *1*, *4*, *5* a *6* **(e)** *1* a *4*

5. (5 b) Doplňte tri chýbajúce fragmenty v nasledujúcom programe tak aby sa po jeho vykonaní vypísal text *vo vnutri*.

```
**1** B {
   void m();
}

class A {
   **2** fun() {
      return new **3** {
         public void m() {
            System.out.println("vo vnutri");
            }
      };
   }
   public static void main(String[] args) {
      new A().fun().m();
   }
}
```

(a) `**1**: class  **2**: B  **3**: B()`

(b) `**1**: interface  **2**: B  **3**: B()`

(c) `**1**: class  **2**: A  **3**: A()`

(d) `**1**: class  **2**: A  **3**: B()`

(e) `**1**: interface  **2**: A  **3**: B()`

6. (5 b) Čo sa vypíše po vykonaní nasledujúceho kódu:

```
class Vynimka1 extends Exception {}

class A {
   int mv(int i) throws Vynimka1 {
      if (i >= 0)
         return i * i;
      else
         throw new Vynimka1();
   }
}

class B {
   public static void main(String[] args) {
      try {
         System.out.println(new A().mv(1));
         System.out.println(new A().mv(-1));
      } catch (Vynimka1 e) {
         System.out.println("Vynimka1");
      }
      System.out.println("Koniec");
   }
}
```

(a)
```
1
Koniec
```

(b)
```
1
```

(c)
```
1
Vynimka1
```

(d)
```
Vynimka1
```

(e)
```
1
Vynimka1
Koniec
```

7. (12 b) Doplňte kód z obr. 1 bez narušenia jestvujúcich vzťahov dedenia tak aby bolo možné uskutočniť nasledujúce príkazy:

```
I[] i = {new A(), new B(), new C(), new X()};

for (int c = 0; c < 4; c++)
   i[c].m();
```

8. (16 b) Pre určitú počítačovú hru pripravujete model postáv. Postavy sa nachádzajú v miestnostiach ktoré tvoria priestor hry. V hre je viac druhov postáv. Druh postavy určuje ako postava reagujú na dotyk (ostatné vlastnosti môžete ignorovať). Jeden druh postavy je čarodejník, ktorý na dotyk zmizne. Ďalší druh je drak, ktorý odletí. Sú však dva špeciálne druhy drakov — zelený drak a čierny drak — ktoré sa správajú ako obyčajný drak, ale ešte predtým zelený drak zareve, kým čierny vypľuje oheň.

(a) (6 b) Pre daný problém nakreslite diagram tried v UML.

(b) (10 b) Napíšte zodpovedajúci kód v Jave aj s implementáciou reakcie na dotyk (jednoduchým hlásením uskutočnenej reakcie).

9. (7 b) Vysvetlite ako by ste aplikovali niektorý z návrhových vzorov v rámci svojho projektu:

- vysvetlite kontext v ktorom aplikujete vzor a
- identifikujte účastníkov a operácie príznačné pre daný vzor v kontexte svojho projektu.

10. (5 b) Ako by ste uplatnili AspectJ v rámci svojho projektu — čo by ste mohli vyčleniť ako aspekt alebo čo by ste mohli pridať v tvare aspektu?