Objektovo-orientované programovanie **B**

doc. Ing. Valentino Vranić, PhD., ÚISI FIIT STU

Skúška – 21. máj 2012

Priezvisko:

Meno:

1 b	
2 b	
3 b	

	a	b	c	d	e
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					

Skúška trvá 75 minút.

V otázkach 1–12 je len jedna možnosť správna. Vyznačte svoju odpoveď krížikom do tabuľky. Odpovede na otázky 13–15 vpíšte do tabuľky.

Pri otázkach 1–15 sa hodnotia len odpovede v tabuľke (bez postupu). V prípade opravy jasne vyznačte odpoveď, ktorá platí. Nesprávna odpoveď, vyznačenie viac odpovedí alebo nejednoznačné vyznačenie má hodnotu 0 bodov. Poškodený list nebude uznaný.

Odpoveď na otázku 18 píšte výlučne na list, na ktorom sa nachádza jej znenie.

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

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

2. (1 b) Atribút triedy, ktorému predchádza kľúčové slovo **private**

(a) sa nezapíše do súboru pri serializácii objektu
(b) je dostupný len v danej hierarchii tried
(c) je dostupný len v danej triede
(d) je dostupný len v rámci jednej nite
(e) je chránený pred zápisom

3. (1 b) Import balíka do programu v Jave

(a) zväčší tento program len v preloženom tvare o kód z balíka
(b) nezväčší tento program
(c) zväčší tento program aj v nepreloženom, aj v preloženom tvare o kód z balíka
(d) zväčší tento program len v preloženom tvare o skutočne použitý kód z balíka
(e) zväčší tento program len v nepreloženom tvare o kód z balíka

4. (2 b) Kód

```
BufferedReader stdin =
    new BufferedReader(
        new InputStreamReader(System.in));
```

zabezpečuje

(a) pohodlnejšiu prácu so štandardným vstupom
(b) čítanie zo štandardného vstupu
(c) vytvorenie a otvorenie štandardného vstupu
(d) otvorenie štandardného vstupu
(e) vytvorenie štandardného vstupu

5. (2 b) Videnia v jazyku AspectJ

(a) neposkytujú vlastnú funkcionalitu, len zachytávajú body spájania
(b) sa musia zavolať explicitne na miestach zachytených zodpovedajúcimi bodovými prierezmi
(c) sa vykonajú ihneď po vzniku inštancie aspektu
(d) sa vyvolajú automaticky na miestach zachytených zodpovedajúcimi bodovými prierezmi
(e) slúžia na tvorbu inštancií aspektov

6. (2 b) Daný je nasledujúci kód:

```
class A {
    void a() throws XException {
        if (...) { ... }
        else throw new XException();
    }
}
class B {
    void b() throws XException { new A().a(); }
}
```

Metóda b() triedy B

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

7. (2 b) Agregácia v objektovo-orientovanom programovaní

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

8. (2 b) V rámci Swing sa kliknutie na tlačidlo v okne sleduje

(a) volaním metódy actionPerformed() príslušného tlačidla v slučke
(b) prijímačom udalosti kliknutia registrovaným pre dané tlačidlo
(c) metódou onClick() tlačidla implementovanou pri odvodení od všeobecného tlačidla JButton
(d) automaticky po pridaní tlačidla do okna metódou add()
(e) zavolaním statickej metódy EventQueue.onClick() a následným zistením, či sa kliknutie vzťahuje na dané tlačidlo

9. (2 b) V programe v jazyku C++ trieda GumovaLopta dedí od triedy Lopta a implementuje špeciálnu verziu metódy skoc(). Daný je nasledujúci kód:

```
Lopta* o;
o = new GumovaLopta();
o->skoc();
```

Zavolá sa týmto správna metóda skoc()?

(a) áno
(b) nie
(c) áno, ale jedine ak je metóda skoc() statická
(d) áno, ale jedine ak je trieda Lopta virtuálna
(e) áno, ale jedine ak je metóda skoc() virtuálna

10. (2 b) Synchronizácia nestatickej metódy

(a) znamená uzamknutie objektu triedy pre hocijaký iný prístup
(b) znamená uzamknutie objektu **this** pre hocijaký iný synchronizovaný prístup
(c) nie je možná
(d) znamená uzamknutie objektu triedy pre hocijaký iný synchronizovaný prístup
(e) znamená uzamknutie objektu **this** pre hocijaký iný prístup

11. (2 b) V diagrame sekvencií v jazyku UML obdĺžnik na zvislej prerušovanej čiare, do ktorého vstupuje orientovaná hrana, označuje

(a) volanie operácie
(b) vytvorenie operácie
(c) prepnutie operácie
(d) vykonávanie operácie
(e) zmenu operácie

12. (2 b) Ak sa v triede nachádza kód vo forme:

```
m(I o) {
   o.op(this);
}
```

je pravdepodobné, že táto trieda je súčasťou vzoru? Ak áno, tak ktorého?

```
abstract class A {
   static int i = 0;
   public abstract void m();
}

class X extends A {
   public void m() { i--; }
}

class Y extends X {
   public void m() { i++; }
}
```

Obr. 1: Kód pre otázky 13 a 14.

---

13. (3 b) Ku kódu v Jave na obr. 1 je daný nasledujúci kód v Jave:

```
A o1 = new X();
Y o2 = new Y();
o1.m();
((X)o1).m();
((A)o1).m();
o2.m();
((A)o2).m();
((X)o2).m();
```

Akú hodnotu nadobudne atribút i vykonaním týchto príkazov?

14. (3 b) Ku kódu v Jave na obr. 1 je daná nasledujúca trieda:

```
class M {
   static void m(Class<? extends A> T, A... o) {
      int i = 0;
      for (A e : o) {
         if (T.isInstance(e))
            i++;
         else
            i--;
      }
      System.out.print(i);
   }

   public static void main(String[] args) {
      m(X.class, new A[]{new X(), new Y(), (X)new Y()});
   }
}
```

Aká hodnota sa vypíše po jej vykonaní?

15. (3 b) Čo sa vypíše po spustení nasledujúceho programu?

```
interface I {
   void op();
}
class A implements I {
   public void op() { System.out.print("a"); }
}
class B extends A {
   public void op() { System.out.print("b"); }
}
class C implements I {
   protected A a;
   public C(A a) { this.a = a; }
   public void op() {
      System.out.print("c");
      a.op();
   }
}
class D extends C {
   public D(A a) { super(a); }
   public void op() {
      System.out.print("d");
      a.op();
   }
}
class M {
   public void opi(I... i) {
      for (I x : i)
         x.op();
   }
   public static void main(String[] args) {
      A o1 = new A();
      A o2 = new B();
      C o3 = new C(o1);
      C o4 = new C(o2);
      C o5 = new D(o1);
      C o6 = new D(o2);
      new M().opi(o1, o2, o3, o4, o5, o6);
   }
}
```

Priezvisko:

Meno:

16. (10 b) Aplikácia slúži na evidenciu úloh. Úlohy môžu byť elementárne a zložené, pričom zložené úlohy pozostávajú z jednej alebo viac podúloh (aj elementárnych, aj zložených). Pre každú úlohu sa eviduje začiatok, trvanie a opis. Úlohy sa uchovávajú v poradí, v akom vznikajú. Je možné ich vypísať, pričom výpis zloženej úlohy zahŕňa aj výpisy úloh, z ktorých táto úloha pozostáva.

Nakreslite diagram tried s najvýznamnejšími vzťahmi, operáciami a atribútmi, ktoré vyplývajú z uvedeného opisu. Napíšte zodpovedajúci kód v Jave. Aplikujte mechanizmy objektovo-orientovaného programovania. Ak je to vhodné, aplikujte niektorý z návrhových vzorov.

Objektovo-orientované programovanie **B**
doc. Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Skúška – 21. máj 2012

1 e

2 c

3 b

—

4 a

5 d

6 c

7 a

8 b

9 e, ale uznávaná bude aj odpoveď b, lebo v otázke je chyba: premenná o mala byť typu Lopta∗

10 b

11 d

12 Visitor

—

13 0

14 3

15 abcacbdadb

40