Objektovo-orientované programovanie **B**

Ing. Valentino Vranić, PhD., ÚISI FIIT STU

Opravná skúška — 25. jún 2008

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					
16					

Skúška trvá najviac 100 minút.

V otázkach 1–16 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 ktorú odpoveď vyberáte. Každá správna odpoveď má hodnotu vyznačenú v otázke. Nesprávna odpoveď, vyznačenie viac odpovedí alebo nejednoznačné vyznačenie má hodnotu 0 bodov. Postup riešenia sa pre otázky 1–16 nehodnotí.

Odpovede na otázky 17 a 18 píšte na prídavný list. Na obidvoch listoch paličkovým písmom napíšte svoje priezvisko a meno.

1. (1 b) Abstraktná trieda v Jave

(a) nemôže dediť
(b) môže mať statické metódy
(c) môže mať len abstraktné metódy
(d) nemôže mať polia
(e) nemôže mať prekonávajúce metódy

2. (1 b) Pri dedení triedy od inej triedy v podtriede je možné zadefinovať

(a) len ďalšie polia
(b) len ďalšie metódy
(c) len metódy, ktorých názvy nejestvujú v nadtriede
(d) ďalšie polia a metódy
(e) len metódy, ktorých názvy jestvujú v nadtriede

3. (1 b) Iterátory v Java API uľahčujú

(a) prechádzanie zoskupeniami
(b) rušenie prvkov v zoskupeniach
(c) pridávanie prvkov do zoskupení
(d) opakovanie vykonávania ľubovoľného kódu
(e) volania abstraktných metód

4. (1 b) Objekt v objektovo-orientovanom programovaní predstavuje

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

5. (2 b) Zapuzdrenie v objektovo-orientovanom programovaní

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

6. (2 b) Diagram objektov v jazyku UML predstavuje jeden

(a) z pohľadov rozloženia
(b) z pohľadov správania
(c) z dynamických pohľadov
(d) zo štrukturálnych pohľadov
(e) z pohľadov interakcie

7. (2 b) Aký prístup je vhodné použiť pri metóde, ktorá má byť prekonaná, ale nemá byť prístupná triedam mimo jej hierarchie dedenia?

(a) **private**
(b) **protected**
(c) **public**
(d) inherited
(e) implicitný prístup (bez modifikátora)

8. (2 b) Na rozdiel od objektovo-orientovaného programovana aspektovo-orientované programovanie umožňuje

(a) oddelenie pretínajúcich záležitostí
(b) rýchlejšie vykonávanie programu
(c) rozdelenie kódu do komponentov
(d) tvorbu modulov
(e) prepletenie pretínajúcich záležitostí

9. (2 b) Kliknutie na tlačidlo vo Swingu zachytáva a spracúva

(a) príslušné okno, v ktorom sa tlačidlo nachádza
(b) prijímač registrovaný pre dané tlačidlo
(c) samotné tlačidlo svojou metódou listen()
(d) alternátor tlačidla
(e) metóda main() v nekonečnej slučke

10. (2 b) Generickým triedam v Jave v jazyku C++ zodpovedajú

(a) iterátory
(b) abstraktné triedy
(c) šablóny (template)
(d) virtuálne triedy
(e) štruktúry (struct)

11. (2 b) Nech o je objekt triedy, ktorá poskytuje verejnú metódu **int** m(). Pole r je definované takto:

Object r[] = **new** Object[o.m()];

Táto definícia je

(a) korektná
(b) korektná jedine ak je metóda m() statická
(c) nekorektná
(d) korektná jedine ak je metóda m() synchronizovaná
(e) korektná jedine ak je metóda m() finálna

12. (3 b) Ku kódu v Jave na obr. 1 je daná nasledujúca trieda:

```
class M {
   static int m(Class<? extends I> T, I... o) {
      int i = 10;
      for (I e : o)
         if (T.isInstance(e))
            i--;
      return i;
   }
```

```
    public static void main(String[] args) {
        System.out.println(
            m(B.class, new I[]{new B(), new A(), new B()}));
    }
}
```

Pri jej vykonaní

(a) vznikne výnimka
(b) vypíše sa 10
(c) vypíše sa 9
(d) vypíše sa 7
(e) vypíše sa 8

13. (3 b) Návrhový vzor Observer slúži na

(a) pridávanie operácií nad objektmi daných tried bez ich zmeny
(b) definovanie závislosti stavu viacerých objektov od ďalšieho objektu
(c) pridávanie vzťahov medzi triedami bez ich zmeny
(d) zabránenie vytváraniu viac než jednej inštancie danej triedy
(e) zabránenie rozširovania kódu

14. (3 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`
(b) `1`
    `Vynimka1`
(c) `1`
    `Vynimka1`
    `Koniec`
(d) `Vynimka1`
(e) `1`
    `Koniec`

15. (3 b) Daný je kód v Jave na obr. 1. Vykonaním týchto príkazov:

```
I i = new B();
i.m();
((A)i).m();
((I)i).m();
```

(a) vznikne chyba v poslednom riadku
(b) nevypíše sa nič
(c) vypíše sa bbb
(d) vypíše sa aaa
(e) vypíše sa bb

```
interface I {
    void m();
}
class A implements I {
    public void m() { System.out.print("a"); }
}
class B extends A {
    public void m() { System.out.print("b"); }
}
```

Obrázok 1: Kód pre otázky 12, 15 a 17.

16. (3 b) Daný je nasledujúci program v Jave:

```
class C {
    C x;
    void m() {
        x = new SubC();
        **1**.f();
    }
    public static void main(String[] args) {
        **2**.m();
    }
}
class SubC **3** {
    void f() {
        System.out.println("f");
    }
}
```

Ktoré fragmenty kódu treba v tomto programe doplniť, aby sa pri jeho vykonaní vypísalo f?

(a) `**1**`: x `**2**`: **new** C()
    `**3**`: **extends** C
(b) `**1**`: SubC `**2**`: C
    `**3**`: **extends** C
(c) `**1**`: ((SubC)x) `**2**`: **new** C()
    `**3**`: nič
(d) `**1**`: ((SubC)x) `**2**`: **new** C()
    `**3**`: **extends** C
(e) `**1**`: x `**2**`: **new** C()
    `**3**`: **implements** C

17. (5 b) Nakreslite diagram tried v UML pre kód na obr. 1.

18. (12 b) Aplikácia na prácu s obrázkami podporuje rôzne formáty obrázkov a umožňuje rôzne operácie nad nimi. Medzi týmito operáciami sú zrkadlový obraz a inverzia farieb. Implementácia týchto operácii závisí od formátu obrázku. Aplikácia v súčasnosti podporuje formáty BMP a JPG, ale v budúcnosti bude potrebné podporiť aj iné formáty.

Napíšte relevantný kód v Jave, ktorý zodpovedá týmto požiadavkám. Aplikujte pritom vhodné mechanizmy objektovo-orientovaného programovania a vysvetlite ich úlohu. Uveďte príklad použitia. Vysvetlite ako by ste pridali nový formát obrázku a novú operáciu. Spôsob uchovánia obrázku a samotná implementácia operácií nad obrázkami nie sú predmetom tejto otázky.

Objektovo-orientované programovanie **B**
Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Opravná skúška — 25. jún 2008

1 b

2 d

3 a

4 e

—

5 e

6 d

7 b

8 a

9 b

10 c

11 a

—

12 e

13 b

14 c

15 c

16 d

50