Objektovo-orientované programovanie **B**

Ing. Valentino Vranić, PhD., ÚISI FIIT STU

Skúška — 4. 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 ňom tiež uveďte svoje priezvisko a meno.

1. (1 b) Prúd údajov (stream) v Java API sa obaľuje, aby

(a) bolo možné prúd údajov presmerovať
(b) bolo ľahšie zachytiť výnimku IOException
(c) nedošlo k strate údajov v dôsledku vysokej rýchlosti ich prenosu
(d) prístup k prúdu údajov mohol byť realizovaný operáciami vyššej úrovne
(e) bolo vôbec možné pracovať s prúdom údajov

2. (1 b) Na rozdiel od Javy jazyk C++

(a) automaticky vytvára objekty
(b) má explicitné deštruktory
(c) má implicitné konštruktory
(d) automaticky ruší nereferencované objekty
(e) má explicitné konštruktory

3. (1 b) Dá sa v Jave niť vytvoriť implementáciou rozhrania Runnable?

(a) nie
(b) áno, ale len atomická
(c) áno, ale len ako démon
(d) áno, ale na spustenie nite sa musí použiť trieda Thread
(e) áno, ale len synchronizovaná

4. (1 b) Prúd údajov (stream) v Java API sa zatvára

(a) príkazom System.close()
(b) prečítaním posledného údaja v ňom
(c) zavolaním jeho deštruktora
(d) príkazom IOStream.close()
(e) jeho metódou close()

5. (2 b) V diagrame sekvencií jazyka UML horizontálne šípky s plnou čiarou označujú

(a) vyvolanie operácie
(b) prenos parametrov operácie v smere šípky
(c) návrat hodnoty
(d) agregáciu
(e) tok údajov medzi objektmi

6. (2 b) Agregácia v objektovo-orientovanom programovaní

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

7. (2 b) V jazyku AspectJ je pomocou videnia (advice) možné

(a) pridať novú metódu do aspektu
(b) pridať novú metódu do triedy
(c) definovať nové závislosti medzi triedami
(d) definovať bod spájania
(e) zmeniť vykonávanie metódy

8. (2 b) Okno v rámci Swing sa dá vytvoriť

(a) zavolaním konštruktora triedy SwingWindow
(b) zavolaním konštruktora triedy JFrame
(c) zavolaním statickej metódy EventQueue.newWindow()
(d) zavolaním statickej metódy SwingUtils.newWindow()
(e) zavolaním metódy newWindow() triedy JFrame

9. (2 b) Prístup **protected** je vhodné použiť pri takých prvkoch triedy, ku ktorým chceme pristupovať len

(a) v odvodených triedach a v triedach toho istého balíka
(b) v odvodených triedach
(c) v danej triede
(d) v odvodených triedach toho istého balíka
(e) v triedach toho istého balíka

10. (2 b) Implementácii rozhrania v Jave v jazyku C++ zodpovedá:

(a) **private** dedenie od **abstract** triedy
(b) **public** dedenie od triedy, ktorá má len virtuálne metódy bez tela
(c) **public** dedenie od **abstract** triedy
(d) **private** dedenie od triedy, ktorá má len virtuálne metódy bez tela
(e) **protected** dedenie od **abstract** triedy

11. (2 b) Princíp otvorenosti a uzavretosti hovorí, že

(a) kód má byť otvorený pre zmeny, ale uzavretý pre rozšírenie
(b) každý prúd údajov okrem štandardného vstupu a výstupu je potrebné po otvorení aj zatvoriť
(c) kód má byť zároveň aj otvorený, aj uzavretý pre úpravy
(d) kód má byť otvorený pre rozšírenie, ale uzavretý pre zmeny
(e) každý prúd údajov je potrebné po otvorení aj zatvoriť

12. (3 b) V programe je každý druh geometrického útvaru reprezentovaný triedou. Kód mimo týchto tried využíva rozmery útvarov na výpočet ich povrchu. Je vhodné odvodiť triedu, ktorá reprezentuje štvorec, od triedy, ktorá reprezentuje obdĺžnik?

(a) áno
(b) áno, ale musíme zabezpečiť, aby sa pri štvorci šírka a výška rovnali
(c) nie, lebo to porušuje princíp otvorenosti a uzavretosti
(d) nie, lebo to porušuje princíp zapuzdrenia
(e) nie, lebo to porušuje Liskovej princíp substitúcie

13. (3 b) Daný je kód v Jave na obr. 2. Vykonaním týchto príkazov pre objekt m triedy M:

```
A o1 = (A)m.f(0);
A o2 = (A)m.f(1);
A o3 = m.f(0);
```

(a) počas vykonávania vznikne výnimka na druhom a treťom riadku
(b) kompilátor hlási chybu na druhom a treťom riadku
(c) kompilátor hlási chybu na treťom riadku; po jej odstránení počas vykonávania vznikne výnimka na druhom riadku
(d) nevzniknú žiadne chyby
(e) kompilátor hlási chybu na druhom riadku; po jej odstránení počas vykonávania vznikne výnimka na treťom riadku

14. (3 b) Daný je kód na obr. 1. Ktoré fragmenty treba do kódu doplniť, aby bol správny?

(a)
```
**1**: NOperation **2**: PlainInt
**3**: this; **4**: n.calc(new NSquare())
```
(b)
```
**1**: PlainInt **2**: int
**3**: p.op(this); **4**: n.calc(n.get())
```
(c)
```
**1**: NOperation **2**: PlainInt
**3**: p.get() **4**: n.calc(new PlainInt());
```
(d)
```
**1**: PlainInt **2**: NOperation
**3**: p.op(this); **4**: n.calc(new NSquare())
```
(e)
```
**1**: PlainInt **2**: NOperation
**3**: p; **4**: n.calc(this)
```

```
public interface NOperation {
    int op(**1** n);
}
public interface Num {
    int calc(**2** p);
    void set(int i);
    int get();
}
public class PlainInt implements Num {
    private int i;
    public void set(int i) { this.i = i; }
    public int get() { return i; }
    public int calc(**2** p) { return **3** }
}
public class NSquare implements NOperation {
    public int op(**1** n) { return n.get() * n.get(); }
}
public class M {
    public static void main(String[] args) {
        Num n = new PlainInt();
        n.set(3);

        System.out.println(**4**);
    }
}
```

Obrázok 1: Kód pre otázky 14 a 15.

15. (3 b) Kód na obr. 1 predstavuje implementáciu (riešte najprv otázku 14)

(a) vzoru Observer, pričom NOperation je pozorovateľ, a Num predmet
(b) vzoru Visitor, pričom NOperation je návštevník, a Num predmet
(c) vzoru Observer, pričom Num je pozorovateľ, a NOperation predmet
(d) vzoru Visitor, pričom Num je návštevník, a NOperation predmet
(e) idiómu double dispatch, pričom Num je double dispatch, a NOperation predmet

```
interface I { }
class A implements I { }
class B implements I { }
class M {
    I f(int a) {
        if (a == 0)
            return new A();
        else
            return new B();
    }
}
```

Obrázok 2: Kód pre otázky 13 a 17.

16. (3 b) K typom

```
abstract class A { }
interface I { }
```

je daný nasledujúci kód (umiestnený v metóde v tom istom balíku):

```
A[] a = new A[5];
I[] i = new I[5];
```

Tento kód sa

(a) preloží a vykoná korektne
(b) preloží, ale vznikne chyba pri vykonávaní prvého riadku
(c) nepreloží, lebo prekladač hlási chybu v druhom riadku
(d) preloží, ale vznikne chyba pri vykonávaní druhého riadku
(e) nepreloží, lebo prekladač hlási chybu v prvom riadku

17. (5 b) Nakreslite diagram tried v UML pre kód na obr. 2.

18. (12 b) Stopky umožňujú spustenie merania času, zastavenie merania a jeho vynulovanie. Merajú s presnosťou na sekundu. Implementujte ich ako triedu, v ktorej čas bude uchovaný v sekundách. Predpokladajte, že sa posun času uskutočňuje volaním metódy, ktorá aktualizuje uchovanú hodnotu. Táto metóda bude zaradená na vykonávanie každú sekundu do plánovača, čo nie je predmetom tejto úlohy.

Čas je potrebné zobraziť súčasne vo viacerých verziách, z ktorých každá má definovaný časový posun vzhľadom na samotné stopky (napr. + 10 s alebo – 255 s). Každé zobrazenie je v tvare hh:mm:ss alebo len v sekundách. V budúcnosti bude potrebné pridať nové druhy zobrazenia (napr. klasickými hodinkami).

Napíšte relevantný kód v Jave, ktorý zodpovedá týmto požiadavkám. Aplikujte pritom mechanizmy objektovo-orientovaného programovania v maximálnej miere a vysvetlite ich úlohu. Samotný výpis času stačí realizovať metódou println().

Objektovo-orientované programovanie **B**
Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Skúška — 4. jún 2008

1 d

2 b

3 d

4 e

—

5 a

6 c

7 e

8 b

9 a

10 b

11 d

—

12 e

13 c

14 d

15 b

16 a

50