**Objektovo-orientované programovanie** **A**

Ing. Valentino Vranić, PhD., ÚISI FIIT STU

Skúška — 23. jún 2006 (opravný termín)

| | |
|---|---|
| **Priezvisko:** | |
| **Meno:** | |

| | |
|---|---|
| 1b | |
| 2b | |
| 3b | |

| | a | b | c | d | e |
|---|---|---|---|---|---|
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |
| 6 | | | | | |
| 7 | | | | | |
| 8 | | | | | |
| 9 | | | | | |
| 10 | | | | | |
| 11 | | | | | |
| 12 | | | | | |
| 13 | | | | | |
| 14 | | | | | |
| 15 | | | | | |
| 16 | | | | | |
| 17 | | | | | |
| 18 | | | | | |

Test trvá 120 minút.

V otázkach 1–18 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 1–18 nehodnotí.

Odpovede na otázky 19 a 20 píšte na list s poslednými tromi otázkami. Na ňom tiež uveďte meno.

**1. (1 b)** Abstraktná metóda v Jave

(a) nemôže mať argumenty
(b) nemôže byť volaná
(c) môže mať telo
(d) nemôže mať návratovú hodnotu
(e) môže sa vyskytovať v hocijakej triede

**2. (1 b)** V triede ktorá dedí od rozhrania je možné zadefinovať

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

**3. (1 b)** V jazyku C++ inline funkcie

(a) predstavujú výlučne jednoriadkové funkcie
(b) prekladač nahrádza kódom na mieste ich volaní
(c) umožňujú polymorfizmus
(d) nemajú telo
(e) nemôžu obsahovať volania iných funkcií

**4. (1 b)** Iterátory v Java API uľahčujú

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

**5. (2 b)** Aspekty v aspektovo-orientovanom programovaní

(a) urychľujú vykonávanie programu
(b) definujú miesta vo vykonávaní programu
(c) definujú rozdelenie kódu do komponentov
(d) modifikujú štruktúru a vykonávanie programu
(e) robia kód zapleteným a roztrúseným

**6. (2 b)** Aby bolo možné využiť polymorfizmus v jazyku C++, zodpovedajúce metódy určené na prekonávanie musia byť

(a) virtuálne
(b) abstraktné
(c) verejné
(d) konštantné
(e) statické

**7. (2 b)** Nech A je trieda ktorá poskytuje verejnú metódu **int** m(). Trieda C je definovaná takto:

```
class C {
   int i = (new A()).m();
}
```

Táto definícia je

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

**8. (2 b)** Prístup **protected** je vhodné použiť pri takých prvkoch triedy ku ktorým chceme pristupovať len

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

**9. (2 b)** Diagram prípadov použitia v jazyku UML znázorňuje

(a) funkcionalitu z pohľadu používateľa alebo iného systému
(b) vzťahy medzi inštanciami tried v určitom okamihu vykonávania programu
(c) triedy a vzťahy medzi nimi
(d) štruktúru systému
(e) postupnosť správ prenášaných medzi objektmi

**10. (2 b)** Agregácia v objektovo-orientovanom programovaní

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

**11. (2 b)** V jazyku AspectJ bodové prierezy (pointcuts) slúžia na

(a) zachytenie bodov spájania
(b) doplnenie nových prvkov do tried
(c) pretínanie tried
(d) modifikáciu vykonávania programu
(e) definovanie nových metód

**12. (3 b)** Návrhový vzor Observer slúži na

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

**13. (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 = 0;
        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 A(), new B()}));
    }
}
```

Pri jej vykonaní

(a) vypíše sa 1
(b) vypíše sa 2
(c) vznikne výnimka
(d) vypíše sa 0
(e) vypíše sa 3

**14. (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 13 a 14.

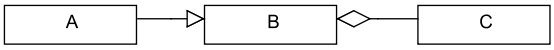
**15. (3 b)** Daný je nasledujúci kód v Jave:

```
class **1** extends **2** { . . . }

class **3** {
    **4** o;
    . . .
}
```

Aby kód zodpovedal diagramu na obr. 2, fragmenty kódu označené číslami 1–4 v danom poradí treba nahradiť identifikátormi:

(a) B, A, A, C
(b) A, B, B, C
(c) B, A, C, A
(d) C, B, B, A
(e) B, C, A, B




Obrázok 2: Diagram pre otázku 15.

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

**17. (3 b)** V jazyku C++ implementujete operácie nad komplexnými číslami. V čase implementácie ešte neviete či zložky komplexných čísel budú môcť byť len celočíselné alebo aj decimálne. Na vyriešenie tohto problému použijete

(a) preťaženie operátorov
(b) virtuálne funkcie
(c) statické funkcie
(d) preťaženie funkcií
(e) šablóny

**Objektovo-orientované programovanie** **A**

Ing. Valentino Vranić, PhD., ÚISI FIIT STU

Skúška — 23. jún 2006 (opravný termín)

| **Priezvisko:** | |
|---|---|
| **Meno:** | |

**18. (3 b)** Vytvorili ste tlačidlo t ako komponent rámca Swing a zviditeľneli ste okno, ktoré ho obsahuje. Potrebujete zmeniť označenie (label) tlačidla na text Abc. Urobite to volaním

(a) t.changeText("Abc")
(b) t.invokeLater(t.changeText("Abc");}});
(c) (**new** Runnable(SwingUtilities.invokeLater(
t.changeText("Abc");}}))).run();
(d) SwingUtilities.invokeLater(**new** Runnable() {
**public void** run() {w.changeText("Novy text");}});
(e) SwingUtilities.run(**new** Runnable() {
**public void** invokeLater() {
w.changeText("Novy text");}});

**19. (6 b)** Vysvetlite princíp otvorenosti a uzavretosti kódu. Uveďte príklad (slovne a so základom kódu).

**20. (10 b)** V programe na prácu s grafickými útvarmi zobrazenie útvarov môže byť v rôznej kvalite (napr. len hrany, s výplňou apod.). Naznačte hierarchiu útvarov. Aplikujte idióm double dispatch na zobrazenie útvarov: uveďte základ kódu s vysvetlením. Uveďte príklad použitia idiómu double dispatch pri vyvolaní zobrazenia.

**Objektovo-orientované programovanie** **A**
Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Skúška — 23. jún 2006 (opravný termín)

**1** b

**2** d

**3** b

**4** c

—

**5** d

**6** a

**7** b

**8** b

**9** a

**10** c

**11** a

—

**12** e

**13** a

**14** c

**15** b

**16** c

**17** e

**18** d

55