Objektovo-orientované programovanie **A**
Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Semestrálny test — 3. apríl 2006

Priezvisko:		**Body**
Meno:		

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

V otázkach je len jedna možnosť správna. Vyznačte svoju odpoveď krížikom do tabuľky uvedenej nižšie. Hodnotia sa len odpovede vyznačené v tabuľke. V prípade opravy jasne vyznačte ktorú odpoveď vyberáte.
Každá správna odpoveď má hodnotu 1 bod. Nesprávna odpoveď, vyznačenie viac odpovedí alebo nejednoznačné vyznačenie má hodnotu 0 bodov. Postup riešenia sa nehodnotí.

1. Konštruktor v Jave

(a) nemôže mať argumenty
(b) musí mať vždy len jeden argument
(c) nemusí byť explicitne poskytnutý **X**
(d) môže vraciať ľubovoľnú hodnotu
(e) nemôže byť preťažený

2. Objekt v objektovo-orientovanom programovaní predstavuje

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

3. Nech o je objekt triedy ktorá poskytuje metódu **int** m(). Reťazec r je definovaný takto:

// Integer r[] = new r[o.m()]; // syntakticka chyba
Integer r[] = **new** Integer[o.m()]; *// takto to malo byt*

Táto definícia je

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

V skupine B je to otázka 5 a odpovede (a) a (c).

4. Príkaz

import java.util.∗;

(a) sprístupní priestor názvov všetkých typov balíka java.util, ale bez typov v podbalíkoch **X**
(b) sprístupní priestor názvov všetkých typov balíka java.util vrátane typov v podbalíkoch
(c) fyzicky pripojí typy balíka java.util k programu bez typov v podbalíkoch
(d) fyzicky pripojí len skutočne použité typy balíka java.util k programu bez typov v podbalíkoch
(e) fyzicky pripojí všetky typy balíka java.util k programu vrátane typov v podbalíkoch

5. Tok údajov (stream) v Java API sa otvára

(a) príkazom System.open()
(b) jeho konštrukciou **X**
(c) jeho prvým použitím
(d) jeho metódou open()
(e) príkazom IOStream.open()

6. Daný je kód v Jave na obr. 1. Čo sa vypíše po vykonaní týchto príkazov:

```
C o = new X();
o.m();
o.s();
((X)o).s();
```

(a) Cm Cs Xs
(b) Xm Xs Xs
(c) Cm Xs Xs
(d) Cs Xm Cs
(e) Xm Cs Xs **X**

```
class C {
    void m() { System.out.print("Cm "); }
    static void s() { System.out.print("Cs "); }
}

class X extends C {
    void m() { System.out.print("Xm "); }
    static void s() { System.out.print("Xs "); }
}
```

Obrázok 1: Kód pre otázky 6, 7 a 10.

7. Daný je kód v Jave na obr. 1. Dá sa z metódy s() triedy X zavolať rovnomenná metóda triedy C?

(a) nie
(b) áno, príkazom X.s(); -> C.s(); (**X**)
(c) áno, príkazom s();
(d) áno, príkazom super.s();
(e) áno, príkazom this.s();

Za otázku 7 sa prideli bod bez ohľadu na odpoveď.

8. 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 **X**
(c) v triedach toho istého balíka
(d) v odvodených triedach toho istého balíka
(e) v danej triede

9. Daný je nasledujúci kód:

```
for (Object o : l) {
    if (o.class == "Frog")
        ((Frog)o).jump();
    else if (o.class == "Bird")
        ((Bird)o).fly();
    else
        ;
}
```

Tento kód porušuje

(a) princíp zapuzdrenia
(b) pravidlá dedenia
(c) Liskovej princíp substitúcie
(d) princíp otvorenosti a uzavretosti **X**
(e) pravidlá polymorfizmu

10. K triedam z obr. 1 je daný nasledujúci kód:

```
List<C> list = new ArrayList<C>();
list.add(new X());
```

Tento kód sa

(a) nepreloží, lebo typ referencie list nezodpovedá typu priradeného objektu
(b) nepreloží, lebo do zoznamu list sa dajú vkladať len objekty typu C
(c) preloží, ale padne počas vykonávania s výnimkou ClassCastException
(d) preloží a vykoná korektne **X**
(e) nepreloží, lebo trieda ArrayList nie je generická

11. Dá sa urobiť inštancia abstraktnej triedy?

(a) áno, ako hociktorej inej triedy
(b) nie **X**
(c) áno, ale len ak trieda neobsahuje abstraktné metódy
(d) áno, ale nebudú sa dať zavolať abstraktné metódy
(e) áno, ale bude abstraktná

12. Daný je nasledujúci kód:

```
1  abstract class A { }
2  interface I { }
3  A[] a = new A[5];
4  I[] i = new I[5];
```

Tento kód sa

(a) preloží a vykoná korektne **X**
(b) nepreloží, lebo prekladač hlási chybu na riadku 3
(c) nepreloží, lebo prekladač hlási chybu na riadku 4
(d) preloží, ale vznikne chyba pri vykonávaní riadku 3
(e) preloží, ale vznikne chyba pri vykonávaní riadku 4

13. Kľúčové slovo **synchronized** slúži na

(a) zmenu priority vykonávania nite
(b) zmenu poradia vykonávania nití
(c) spojenie dvoch niti
(d) určenie metód ktoré sa môžu vykonávať súčasne
(e) uzamknutie objektu pre vyhradený prístup k jeho poliam **X**

14. Daný je nasledujúci program:

```
**1** B {
    **2**
}

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

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

(a) `**1**`: **class** `**2**`: **void** f(); `**3**`: B
(b) `**1**`: **interface** `**2**`: **void** m(); `**3**`: B **X**
(c) `**1**`: **class** `**2**`: **void** m(); `**3**`: A
(d) `**1**`: **class** `**2**`: **void** m(); `**3**`: B
(e) `**1**`: **interface** `**2**`: **void** f(); `**3**`: B

15. Daný je nasledujúci kód:

```
class MyException extends Exception {}

class A {
    void m() throws MyException {
        . . .
    }
}

class B {
    void m() {
        new A().m();
    }
}
```

Metóda m() triedy B

(a) je korektná
(b) musí deklarovať že vyhadzuje výnimku typu MyException
(c) musí deklarovať alebo ošetrovať výnimku typu MyException **X**
(d) musí ošetrovať výnimku typu MyException
(e) musí vyhadzovať výnimku typu MyException

Objektovo-orientované programovanie **A**
Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Semestrálny test — 3. apríl 2006

1 c

2 b

3 e

4 a

5 b

6 e

7 b

8 b

9 d

10 d

11 b

12 a

13 e

14 b

15 c