

Základy objektovo-orientovaného programovania

B

Ing. Ján Lang, PhD., UISI FIIT STU
Skúška - 20. januára 2015

Priezvisko:

Meno:

1		Test trvá 75 minút.
2		V uzavretých otázkach 1-15 s ponúknutými
3		odpoveďami je vždy správna iba jedna
4		možnosť. Do tabuľky uveďte písmeno pod
5		ktorým je označená odpoveď, ktorú vyberá-
6		te. Hodnotia sa len odpovede v tabuľke.
7		V prípade opravy jasne vyznačte odpoveď,
8		ktorá platí. Každá správna odpoveď má
9		hodnotu vyznačenú v otázke. Nesprávna
10		odpoveď, alebo nejednoznačné vyznačenie
11		má hodnotu 0 bodov. Postup riešenia sa
12		v otázkach 1-15 nehodnotí. Akceptovaný
13		bude len odovzdaný celistvý list.
14		Riešenie úlohy 16 píšete do prázdneho mies-
15		ta na liste na ktorom sa nachádza jej znenie.
		Poškodený list nebude uznaný.

1. (2b) Daný je nasledujúci kód v Jave:

```
class A {
    A() { System.out.print("A"); }
    static void sf() { System.out.print("A"); }
}

class B extends A {
    B() { System.out.print("B"); }
    static void sf() { System.out.print("B"); }
}

class C extends B {
    C() { System.out.print("C"); }
    static void sf() { System.out.print("C"); }
}

A b = new C();
b.sf();
```

Kód uvedený vyššie reprezentuje:

- (a) upcasting, s inštanciou triedy C bude zaobchádzané ako s inštanciou triedy B
- (b) upcasting, s inštanciou triedy C bude zaobchádzané ako s inštanciou triedy A
- (c) downcasting, s inštanciou triedy C bude zaobchádzané ako s inštanciou triedy B
- (d) downcasting, s inštanciou triedy C bude zaobchádzané ako s inštanciou triedy A
- (e) žiadne z uvedených

2. (1b) Implementácia metódy rovnakej signatúry v podtype:

- (a) nemá zmysel ale mala by byť identická
- (b) nemá zmysel
- (c) má zmysel a mala by byť identická
- (d) má zmysel a nemala by byť špecifická
- (e) má zmysel a mala by byť špecifická
- (f) žiadne z uvedených

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

```
class A {
    public void f() { System.out.print("A"); }
    public void af() { System.out.print("Af"); }
}

class B extends A {
    public void f() { System.out.print("B"); }
    public void af() { System.out.print("ABf"); }
    public void bf() { System.out.print("Bf"); }
}

A a = new B();
B b = new B();

a.f();
(new B()).bf();
b.af();
((B) a).af();
(new B()).f();
```

Po kompilácii kódu uvedeného vyššie sa vypíše:

- (a) BBfABfABfB
- (b) AAfBfABfAf
- (c) BBfBBfABfB
- (d) BAFBfABfAf
- (e) BBfABfBABf
- (f) chyba pri kompilácii

4. (2b) Daný je nasledujúci kód v Jave:

```
class A {
    public void f() { System.out.print("A"); }
    public void af() { System.out.print("Af"); }
}

class B extends A {
    public void f() { System.out.print("B"); }
    public void bf() { System.out.print("Bf"); }
}

class C extends A {
    public void f() { System.out.print("C"); }
    public void cf() { System.out.print("Cf"); }
}

A[] alphabet = new A[2];
alphabet[0] = new B();
alphabet[1] = new C();

for (A a : alphabet) {
    a.f();
}
```

O tom, ktorá verzia metódy `f()` sa spustí:

- (a) sa rozhodne pri kompilácii
- (b) sa nerozhodne
- (c) sa rozhodne až pri zavolaní metódy v bežiacom programe
- (d) sa rozhodne pri kompilácii za predpokladu, že sú vytvorené explicitné konštruktory
- (e) žiadne z uvedených

5. (3b) Použitie polymorfizmu je vtedy:

- (a) keď v referencii na triedu potomka používame odkaz na objekty triedy rodiča
- (b) keď v referencii na triedu rodiča používame odkaz na objekty triedy potomka
- (c) keď v referencii na triedy rodičov používame odkaz na objekt triedy potomka
- (d) keď v referencii na triedu potomka používame odkaz na objekt triedy rodiča
- (e) žiadne z uvedených

6. (1b) Daný je nasledujúci kód v Jave:

```
public class Potraviny {  
    void doplnVitaminy(Obyvatel o);  
  
    public String toString() {  
        return "Ja som Potravina s nazvom: ";  
    }  
}
```

Metóda toString()

- (a) prekonáva java.lang.Object.toString metódu
- (b) prekonáva java.lang.toString metódu
- (c) preťažuje java.lang.toString metódu
- (d) preťažuje java.lang.Object.toString metódu
- (e) preťažuje doplnVitaminy(Obyvatel o) metódu
- (f) žiadne z uvedených

7. (2b) V prípade kódu z príkladu uvedeného vyššie výpis:

Ja som Potravina s nazvom: dosiahneme:

- (a) konštrukciou: System.out.println(new Potraviny);
- (b) explicitným volaním metódy toString()
- (c) implicitným volaním metódy doplnVitaminy();
- (d) volaním metódy doplnVitaminy(Obyvatel o);
- (e) vždy po vytvorení inštancie triedy Potraviny
- (f) žiadne z uvedených

8. (2b) Daný je nasledujúci kód v Jave:

```
public class Seno extends Potrava {  
    public void nakrm(Zviera z) {  
        z.zjedz(this);  
    }  
}
```

Kľúčové slovo this

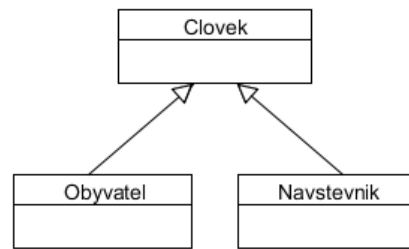
- (a) reprezentuje inštanciu triedy Zviera
- (b) volá metódu nakrm() nadtypu triedy Seno
- (c) volá metódu nakrm() nadtypu triedy Zviera
- (d) volá konštruktor triedy Seno
- (e) volá konštruktor triedy Zviera
- (f) reprezentuje inštanciu triedy Seno
- (g) žiadne z uvedených

9. (1b) Viacnásobná implementácia rozhraní v jave je:

- (a) nepovolená
- (b) nepodporovaná
- (c) neštandardná
- (d) použiteľná
- (e) žiadne z uvedených

10. (2b) Dané je nasledovné:

```
Obyvatel o = new Obyvatel();
```



Ktoré z tvrdení o objekte/inštancii o triedy Obyvatel nie je pravdivé:

- (a) objekt o sa dá bezpečne pretypovať na objekt Navstevnik použitím kľúčového slova instanceof
- (b) kompilátor Java pracuje s inštanciou o ako s inštanciou triedy Object
- (c) objekt o je typu Obyvatel a má k dispozícii všetky metódy z triedy Clovek aj Obyvatel
- (d) objekt o sa dá pretypovať na objekt Clovek
- (e) žiadne z uvedených

11. (2b) Atribút s prístupom protected je prístupný:

- (a) len v rámci balíka
- (b) v rámci balíka pokiaľ sa nejedná o default-ný balík
- (c) len v rámci hierarchie tried, do ktorej daná trieda patrí
- (d) v rámci zdrojového súboru, do ktorej daná trieda patrí pričom súbor a trieda musia mať rovnaký názov
- (e) v rámci balíka a v rámci hierarchie tried, do ktorej daná trieda patrí
- (f) v rámci zdrojového súboru, do ktorej daná trieda patrí pričom súbor a trieda nemusia mať rovnaký názov

12. (2b) Daný je nasledujúci kód v Jave:

```
public class SampleClass {  
    protected int intField = 0;  
    protected double doubleField = 0;  
  
    public void setIntField(int intField) {  
        this.intField = intField;  
    }  
  
    public static void main(String[] args) {  
        SampleClass so = new SampleClass();  
        so.doubleField = so.intField++;  
        System.out.println(++so.doubleField +  
            so.intField);  
        int intVariable = so.intField;  
        so.setIntField(100);  
        intVariable = so.intField++;  
    }  
}
```

Ktorý z nasledujúcich príkazov nemení stav objektu so?

- (a) so.doubleField = so.intField++;
- (b) System.out.println(++so.doubleField + so.intField);
- (c) so.setIntField(100);
- (d) int intVariable = so.intField;
- (e) intVariable = so.intField++;

Základy objektovo-orientovaného programovania

B

Ing. Ján Lang, PhD., UIIS FIIT STU
Skúška - 20. januára 2015

Priezvisko:

Meno:

13. (3b) Daný je nasledujúci kód v Jave:

```
class A {
    public void f() { System.out.print("A"); }
    public void af() { System.out.print("Af"); }
}

class B extends A {
    public void f() { System.out.print("B"); }
    public void bf() { System.out.print("Bf"); }
}

class C extends B {
    public void f() { System.out.print("C"); }
    public void cf() { System.out.print("Cf"); }
}

public static void main(String[] args) {
    A a = new C();
    B b = new B();
    B c = new A();

    a.f();
    a.af();
    b.f();
    b.bf();
    c.f();
    c.af();
}
```

Po kompilácii kódu uvedeného vyššie sa vypíše:

- (a) BBfAfBBfA
- (b) CAfBBfCAf
- (c) BAfBBfBAf
- (d) CAfBBfBAf
- (e) ABfAfBBfA
- (f) chyba pri kompilácii

14. (2b) Ktoré z tvrdení vo vzťahu k vykonaniu kódu uvedenému v otázke 8. je pravdivé:

- (a) inštancia triedy Zviera vie „zjesť“ výhradne inštanciu triedy Seno a inštancia triedy Seno vie nakŕmiť ľubovoľné Zviera
- (b) inštanciu triedy Potrava nevie „zjesť“ ľubovoľné Zviera ale inštancia triedy Seno vie nakŕmiť ľubovoľné Zviera
- (c) inštancia triedy Zviera vie „zjesť“ inštanciu triedy Potrava a inštancia triedy Potrava vie nakŕmiť ľubovoľné Zviera
- (d) inštanciu triedy Potrava nevie „zjesť“ ľubovoľné Zviera ani inštancia triedy Seno nevie nakŕmiť ľubovoľné Zviera
- (e) inštanciu triedy Seno nevie „zjesť“ ľubovoľné Zviera ale inštancia triedy Potrava vie nakŕmiť ľubovoľné Zviera
- (f) žiadne z uvedených

15. (2b) Daný je nasledujúci kód v Jave:

```
public interface I {
    void m();
}

public abstract class A implements I { ... }

public class C extends A { ... }
```

V kóde vyššie uvedenom - implementovať predpísanú metódu `m()` v triede `A`:

- (a) musím ako abstraktnú
- (b) musím ale už priamo v rozhraní
- (c) neviem ju implementovať
- (d) nesiem
- (e) nemusím

16. (10 b) Obyvateľ nášho simulovaného mesta má rôzne možnosti uplatnenia sa vzhľadom na jeho kvalifikáciu. Počas svojho života sa môže vzdelávať a tým nadobúdať istú mieru vedomostí, zručností a kompetencií pre danú kvalifikáciu. Rôznym spôsobom kvalifikovaní obyvatelia môžu pracovať u rôznych zamestnávateľov a podávať rôzny výkon. Zamestnaní sa ponúka široká škála. Škálu zamestnávateľov ako aj kvalifikácií je možné vždy ďalej rozširovať. Napíšte zodpovedajúci kód v Jave. Mapujte reálne entity virtuálneho sveta a aplikujte adekvátne mechanizmy objektovo-orientovaného programovania. Špeciálne uplatnite polymorfizmus. Napokon nakreslite diagram identifikovaných tried s uvedením vzťahov medzi triedami. Uplatnené mechanizmy OOP v kóde viditeľne vyznačte.

Spolu 40 bodov

Riešenie:

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