

Základy objektovo-orientovaného programovania

A

Ing. Ján Lang, PhD., UIIS FIIT STU

Skúška - 10. januára 2014

Priezvisko:

Meno:

1		Test trvá 75 minút.
2		V uzavretých otázkach 1-16 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 1-16 nehodnotí. Akceptovaný
13		bude len odovzdaný celistvý list.
14		Riešenie úlohy 17 píšete do prázdneho mies-
15		ta na liste na ktorom sa nachádza jej znenie.
16		Poškodený list nebude uznaný

1. (1b) Triedy odvodené od triedy, ktorá implementuje nejaké rozhranie

- (a) Musia implementovať rozhraním predpísané metódy
- (b) Môžu implementovať rozhraním predpísané metódy
- (c) Nesmú implementovať rozhraním predpísané metódy
- (d) Nevedia implementovať rozhraním predpísané metódy
- (e) Žiadna z uvedených odpovedí

2. (1b) Preťaženie metód vzniká

- (a) Keď sa program snaží alokovať viac pamäte ako je možné
- (b) Keď majú dve triedy rovnaký názov metód ale s rôznym zoznamom parametrov
- (c) Keď majú dve triedy rovnaký názov metód s rovnakým zoznamom parametrov
- (d) Keď dve metódy tej istej triedy nesú rovnaký názov, ale líšia sa v zozname parametrov
- (e) Nevzniká nikdy

3. (1b) Zapuzdrenie v jave je dobré používať

- (a) Lebo umožňuje lepšie triediť triedy do balíkov
- (b) Lebo každý objekt sa môže uplatniť na mieste objektu hociktorého z jeho nadtypov
- (c) Aby implementácia objektu zostala skrytá
- (d) Na priame sprístupnenie privátnych premenných
- (e) Nie je dobré a ani žiadane používať

4. (1b) Daný je kód v Jave

```
class Number {  
    int i;  
}
```

```
Number n1 = new Number();  
Number n2 = new Number();  
n1.i = 1;
```

```
n2.i = 2;  
System.out.print(n1.i + n2.i);  
n1 = n2;  
System.out.print(n1.i - n2.i);  
n1.i = 3;  
System.out.print(n1.i + " " + n2.i);
```

Aký je výstup programu?

- (a) 302null
- (b) 3430
- (c) 3022
- (d) 3033
- (e) 12033

5. (2b) Daný je kód v Jave

```
class Utvar {  
    int farba = 12;  
    void nakresli(int farba) {  
    }  
}  
  
class Kruh extends Utvar {  
    void nakresli(int f) {  
        System.out.println("Kruh farby " + f);  
    }  
}
```

```
Kruh k = new Kruh();  
k.nakresli(21);
```

Vyberte z nasledovných možností to čo spôsobí kód uvedený vyššie

- (a) vypíše na konzolu: Kruh farby 12
- (b) vypíše na konzolu: Kruh farby 21
- (c) zahlásí chybu pri preklade nakoľko metóda nakresli triedy Kruh bez parametra nie je deklarovaná
- (d) nevypíše nič
- (e) Zahlásí chybu v počas behu programu

6. (2b) Daný je kód v Jave

```
class Soup {  
    Soup() {}  
    private static Soup ps1 = new Soup();  
    public static Soup access() {  
        return ps1;  
    }  
    public void f() {  
        System.out.println("It tastes good...");  
    }  
}
```

Aby sme dosiahli výpis textu: It tastes good... použijeme príkaz:

- (a) Soup.f(new S());
- (b) Soup.access(f());
- (c) Soup s = new S(); s.f();
- (d) Soup.f().access();
- (e) Soup s = new S(f());
- (f) Soup.access().f();

7. (1b) Upcasting predstavuje:

- (a) Implicitnú zmenu podtypu referencie na objekt z typu nadtriedy na typ podtriedy
- (b) Explicitnú zmenu typu referencie na objekt z typu nadtriedy na typ podtriedy
- (c) Implicitnú zmenu podtypu referencie na objekt z typu nadtriedy na typ podtriedy
- (d) Explicitnú zmenu podtypu referencie na objekt z typu nadtriedy na typ podtriedy
- (e) Implicitnú zmenu typu referencie na objekt z typu podtriedy na typ nadtriedy
- (f) Explicitnú zmenu typu referencie na objekt z typu podtriedy na typ nadtriedy

8. (2b) Daný je kód v Jave

```
public class A {  
    void zarad(A a) { }  
}  
  
public class B extends A { }  
  
public static void main(String[] args) {  
    A a = new A();  
    B b = new B();  
    a.zarad(b);  
}
```

Vyberte z nasledovných možností to čo spôsobí kód uvedený vyššie

- (a) Zahlási chybu pri kompilácii
- (b) Metóda bude akceptovať objekt podtriedy
- (c) Metóda nebude akceptovať objekt podtriedy
- (d) Zahlási chybu v počas behu programu
- (e) Metóda bude očakávať výlučne objekt nadtriedy
- (f) Metóda nebude očakávať referenciu na prázdny objekt

9. (1b) Pre polymorfizmus s výnimkou finálnych metód je príznačné:

- (a) Výber tela metódy sa uskutoční až v čase vykonávania programu
- (b) Dve metódy tej istej triedy môžu niesť rovnaký názov ak sa líšia v zozname parametrov
- (c) Návrátová hodnota sa nedá použiť na rozlíšenie medzi preťaženými metódami
- (d) Promócia primitívnych typov
- (e) Pri preťažených metódach sa vyberie metóda, ktorej veľkosť typu formálneho parametra je najbližšia skutočnému
- (f) Inicializácia statických atribútov prebehne pri načítaní triedy, inak pri vytvorení objektu

10. (2b) Daný je kód v Jave

```
public class A {  
    void fa(A a) { }  
}  
  
public class B extends A {  
    void fb(B b) { }  
}  
  
public static void main(String[] args) {  
    A a = new A();  
    A b = new B();  
}
```

```
B c = new B();  
}
```

Volanie metódy fb prostredníctvom referencie b:

- (a) Je možné príkazom ((B) b).fb(c);
- (b) Je možné príkazom (B) b.fb(c);
- (c) Je možné príkazom ((B) b).fb(b);
- (d) Je možné príkazom b.fb(c);
- (e) Nie je možné

11. (3b) Daný je nasledujúci kód v Jave:

```
abstract class A {  
    void A() {  
        System.out.print("A");  
    }  
  
    abstract void nakresli();  
}  
  
public class B extends A {  
    void nakresli() {  
        System.out.print("B");  
    }  
}  
  
A a = new A();  
a.nakresli();
```

Vyberte z nasledovných možností to čo spôsobí kód uvedený vyššie

- (a) Vypíše: A
- (b) Vypíše: B
- (c) Vypíše: AB
- (d) Vypíše: BA
- (e) Zahlási chybu pri kompilácii
- (f) Zahlási chybu počas behu programu

12. (3b) Daný je nasledujúci kód v Jave:

```
class A {  
    static void sf() { System.out.print("A"); }  
}  
  
class B extends A {  
    static void sf() { System.out.print("B"); }  
}  
  
class C extends B {  
    static void sf() { System.out.print("C"); }  
}  
  
B b = new C();  
b.sf();
```

Model uvedený vyššie vypíše:

- (a) nič
- (b) A
- (c) B
- (d) C
- (e) CB
- (f) BC
- (g) ABC
- (h) AC

Základy objektovo-orientovaného programovania

A

Ing. Ján Lang, PhD., UISI FIIT STU
Skúška - 10. januára 2014

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"); }
}

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

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

Kód uvedený vyššie vypíše:

- (a) nič
- (b) nič lebo kompilátor zahľási chybu
- (c) AAfAfABf
- (d) BBfAfBBf
- (e) ABfAfABf
- (f) BAfAfBBf

14. (3b) Daný je nasledujúci kód v Jave:

```
public class A {
    protected void f() {
    }
}

public class B {
    public static void main(String[] args)
    {
        A pc = new A();
        pc.f();
        B f = new B();
        f.f();
        A pcf = new B();
        pcf.f();
        B fpc = new A();
        fpc.f();
    }
}
```

Korektné bude volanie metódy f objektu:

- (a) pc
- (b) f
- (c) pcf
- (d) fpc
- (e) žiadneho z uvedených

15. (1b) Rozhranie umožňuje:

- (a) Definovať správanie s implementáciou
- (b) Definovať správanie bez implementácie
- (c) Implementovať správanie priamo v rozhraní
- (d) Predpisovať správanie, ktoré nie je záväzné
- (e) Predpisovať správanie implementované priamo v abstraktných triedach
- (f) Implementovať správanie až v definícii abstraktných tried

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

```
public interface Interfaceble {
    void printI();
}

class A implements Interfaceble {
    void printI(){
        System.out.println("class A implements
        Interfaceble");
    }
}
```

Pre uvedený kód kompilátor zahľási chybu pretože:

- (a) Implementácia metódy printI() rozhrania v triede A musí byť static
- (b) Implementácia metódy printI() rozhrania v triede A musí byť private
- (c) Implementácia metódy printI() rozhrania v triede A musí byť protected
- (d) Implementácia metódy printI() rozhrania v triede A nemusí mať uvedený modifikátor prístupu
- (e) Implementácia metódy printI() rozhrania v triede A musí byť public
- (f) Implementácia metódy printI() rozhrania v triede A musí byť final

17. (10 b) Dane sú neodmysliteľnou súčasťou života každého obyvateľa v našom simulátore mesta. Rôzne typy obyvateľov uhrádzajú rôzne typy daní. Dane sa časom vyvíjajú a je silný predpoklad, že budú pribúdať nové. 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	d
3	c
4	d
5	b
6	f
7	e
8	b
9	a
10	a
11	e
12	c
13	f
14	a
15	b
16	e