Základy objektovo-orientovaného programovania **A**
Ing. Ján Lang, PhD., UISI FIIT STU
Skúška - 20. januára 2015

Priezvisko:
Meno:

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

Test trvá 75 minút.
V uzavretých otázkach 1-15 s ponúknutými odpoveďami je vždy správna iba jedna možnosť. Do tabuľky uveďte písmeno pod ktorým je označená odpoveď, ktorú vyberáte. Hodnotia sa len odpovede v tabuľke. V prípade opravy jasne vyznačte odpoveď, ktorá platí. Každá správna odpoveď má hodnotu vyznačenú v otázke. Nesprávna odpoveď, alebo nejednoznačné vyznačenie má hodnotu 0 bodov. Postup riešenia sa v otázkach 1-15 nehodnotí. Akceptovaný bude len odovzdaný celistvý list.
Riešenie úlohy 16 píšte do prázdneho miesta na liste na ktorom sa nachádza jej znenie. Poškodený list nebude uznaný.

1. (3b) Použitie polymorfizmu je vtedy:
(a) keď v referencii na triedu potomka používame odkaz na objekty triedy rodiča
(b) keď v referencii na triedy rodičov používame odkaz na objekt triedy potomka
(c) keď v referencii na triedu potomka používame odkaz na objekt triedy rodiča
(d) keď v referencii na triedu rodiča používame odkaz na objekty triedy potomka
(e) žiadne z uvedených

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

3. (2b) V prípade kódu z príkladu uvedeného vyššie výpis:
`Ja som Potravina s nazvom:` dosiahneme:
(a) implicitným volaním metódy `doplnVitaminy();`
(b) volaním metódy `doplnVitaminy(Obyvatel o);`
(c) vždy po vytvorení inštancie triedy `Potraviny`
(d) konštrukciou: `System.out.println(new Potraviny);`
(e) explicitným volaním metódy toString()
(f) žiadne z uvedených

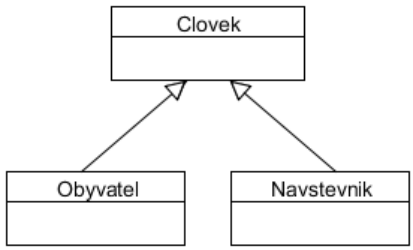
4. (1b) Implementácia metódy rovnakej signatúry v podtype:
(a) má zmysel a mala by byť identická
(b) má zmysel a nemala by byť špecifická
(c) má zmysel a mala by byť špecifická
(d) nemá zmysel ale mala by byť identická
(e) nemá zmysel
(f) žiadne z uvedených

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

6. (2b) Dané je nasledovné:
```
Obyvatel o = new Obyvatel();
```



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

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

8. (1b) Viacnásobná implementácia rozhraní v jave je:
(a) nepovolená
(b) použiteľná
(c) nepodporovaná
(d) neštandardná
(e) žiadne z uvedených

9. (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) CAfBBfBAf
(b) ABfAfBBfA
(c) BBfAfBBfA
(d) CAfBBfCAf
(e) BAfBBfBAf
(f) chyba pri kompilácii

10. (2b) Ktoré z tvrdení vo vzťahu k vykonaniu kódu uvedenému v otázke 7. je pravdivé:
(a) inštancia triedy Zviera vie „zjesť“ výhradne inštanciu triedy Seno a inštancia triedy Seno vie nakŕmiť ľubovoľné Zviera
(b) inštancia triedy Zviera vie „zjesť“ inštanciu triedy Potrava a inštancia triedy Potrava vie nakŕmiť ľubovoľné Zviera
(c) inštanciu triedy Potrava nevie „zjesť“ ľubovoľné Zviera ale inštancia triedy Seno 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

11. (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) nesiem
(d) nemusím
(e) neviem ju implementovať

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) intVariable = so.intField++;
(e) int intVariable = so.intField;

Základy objektovo-orientovaného programovania **A**
Ing. Ján Lang, PhD., UISI FIIT STU
Skúška - 20. januára 2015

Priezvisko:
Meno:

---

13. (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 zdrojového súboru, do ktorej daná trieda patrí pričom súbor a trieda nemusia mať rovnaký názov
(f) v rámci balíka a v rámci hierarchie tried, do ktorej daná trieda patrí

14. (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) AAfBfABfAf
(b) BAfBfABfAf
(c) BBfABfBABf
(d) BBfBBfABfB
(e) BBfABfABfB
(f) chyba pri kompilácii

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

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