Základy objektovo-orientovaného programovania **A**
Ing. Ján Lang, PhD., UISI FIIT STU
Skúška - 19. januára 2016

Priezvisko:
Meno:

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	

Test trvá 75 minút.
V uzavretých otázkach 1-16 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 1-16 nehodnotí. Akceptovaný bude len odovzdaný celistvý list.
Riešenie úlohy 17 píšte do prázdneho miesta na liste na ktorom sa nachádza jej znenie. Poškodený list nebude uznaný.

1. (2b) Polymorfizmus je:
(a) mechanizmus, ktorý umožňuje objektom rôznych typov odpovedať na volanie rôznych metód rovnakým spôsobom
(b) mechanizmus, ktorý umožňuje triedam rôznych objektov odpovedať na volanie rovnakej metódy rovnakým spôsobom
(c) mechanizmus, ktorý umožňuje triedam rôznych typov odpovedať na volanie rovnakej metódy rovnakým spôsobom
(d) mechanizmus, ktorý umožňuje objektom rôznych typov odpovedať na volanie rovnakej metódy rôznym spôsobom
(e) mechanizmus, ktorý umožňuje metódam rôznych objektov odpovedať na volanie rovnakej triedy rôznym spôsobom

2. (2b) Pre polymorfizmus s výnimkou statických a finálnych metód je príznačné:
(a) Dve metódy tej istej triedy môžu niesť rovnaký názov ak sa líšia v zozname parametrov
(b) Návratová hodnota sa nedá použiť na rozlíšenie medzi preťaženými metódami
(c) Promócia primitívnych typov
(d) Výber tela metódy sa uskutoční až v čase vykonávania programu
(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

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

```
public interface Skladovatelny {
       void skladuj(Sklad s);
}

public abstract class Potrava implements Sklado-
vatelny{
  void nakrm(Zviera z) {
    System.out.println("Som potrava a krmim
    zviera.");
    }
}

public class Kost extends Potrava {
  void nakrm(Zviera z) {
    System.out.println("Som Kost a krmim
    zviera.");
    }
}

public abstract class Zviera {
         void zjedz(Potrava p) {
              System.out.println("Zviera");
       }
}

public class Sklad {
       void pridajDoSkladu(Potrava p) {
       }
}
```

Nasledovná metóda

```
       public void skladuj(Sklad s) {
              s.pridajDoSkladu(this);
       }
```

s názvom skladuj:
(a) Musí byť implementovaná v triede Potrava
(b) Nemôže byť implementovaná v triede Potrava
(c) Musí byť implementovaná v triede Kost
(d) Môže byť implementovaná v triede Kost
(e) Nemôže byť implementovaná v triede Kost
(f) Žiadna z uvedených možností

4. (2b) Kľúčové slovo this v príkaze
`s.pridajDoSkladu(this);`
predstavuje:
(a) referenciu na implicitný konštruktor
(b) referenciu na explicitný konštruktor
(c) referenciu na inštanciu triedy
(d) referenciu na rozhranie (interface)
(e) referenciu na inštanciu abstraktnej triedy
(f) referenciu na inštanciu metódy

5. (1b) Ktoré z nasledovných tvrdení o dedení v Jave je nesprávne:
(a) Pozíciu v hierarchii implikuje úroveň abstrakcie
(b) Predstavuje konkretizáciu/zovšeobecnenie
(c) Definuje vzťah nadtyp a podtyp
(d) Reprezentuje princíp znovu použitia programového kódu
(e) Umožňuje zmeniť to, čo z rodičovskej triedy nevyhovuje
(f) Predstavuje rozšírenie viac ako jednej triedy

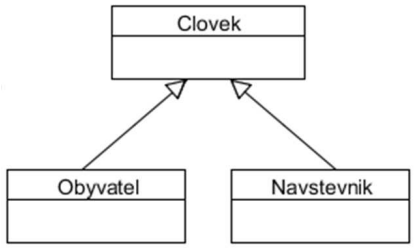
6. (2b) Daný je nasledujúci kód v Jave:

```
public class Obyvatel extends Clovek {
  Auto sukromeAuto;
  int vek;
  int silaZraku;
  int rozpocet;

void zaplatDanZNehnutelnosti(int dan) {
  rozpocet -= dan;
  }
}

public static void main(String[] args) {
  Obyvatel o = new Obyvatel();
  System.out.println(o);
  o.zaplatDanZNehnutelnosti(100);
  Clovek c;
}
```

Upcasting dosiahneme ak:
(a) Referencii c priradíme referenciu o
(b) Referencii o priradíme referenciu c
(c) Referencii o priradíme inštanciu c
(d) Vytvoríme inštanciu triedy `Clovek c=new Clovek();`
(e) Nedosiahneme nakoľko neexistuje objekt triedy Clovek

7. (2b) Keď je v rodičovskej triede implicitný konštruktor alebo je medzi konštruktormi v rodičovskej triede konštruktor bez parametrov:
(a) konštruktor v triede potomka nesmie byť implicitný
(b) konštruktor v triede potomka musí byť parametrický
(c) konštruktor v triede potomka nesmie byť parametrický
(d) konštruktor v triede potomka môže byť implicitný
(e) konštruktor v triede potomka musí byť bezparametrický

8. (3b) Daný je nasledujúci vzťah tried a kód v Jave:




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

Ktoré z nasledovných tvrdení je nesprávne:
(a) Kompilátor Javy pracuje s objektom referencovaným o ako s inštanciou triedy Object
(b) Objekt referencovaný o je typu Obyvatel; má k dispozícii všetky metódy z triedy Clovek aj Obyvatel
(c) Objekt referencovaný o sa dá pretypovať na objekt Clovek
(d) Objekt referencovaný o sa nedá pretypovať na objekt Navstevnik
(e) Referencia o je inštanciou tredy Obyvateľ

9. (1b) Dedenie predstavuje:
(a) Vytvorenie inštancie existujúcej triedy v novej triede
(b) Vytvorenie novej triedy ako typu už existujúcej triedy
(c) Vytvorenie objektu
(d) Prevzatie funkcionality existujúcej triedy a jej ďalšie možné rozšírenie vrátane modifikácie existujúcej triedy
(e) Jednoduché znovu použitie funkcionality, nie formy
(f) Žiadne z uvedeného

10. (2b) Modifikátor prístupu protected
(a) Sprístupňuje atribúty nadtypu, ktoré nemajú ostať dostupné v podtype a v celej hierarchii dedenia, ani v balíku
(b) Sprístupňuje atribúty podtypu, ktoré majú ostať dostupné v nadtype a v celej hierarchii dedenia, tiež v balíku
(c) Sprístupňuje atribúty nadtypu, ktoré majú ostať dostupné v podtype a v celej hierarchii dedenia, nie však v balíku
(d) Sprístupňuje atribúty nadtypu, ktoré majú ostať dostupné v podtype a v celej hierarchii dedenia, tiež mimo balíka
(e) Sprístupňuje atribúty nadtypu, ktoré majú ostať dostupné v podtype a v celej hierarchii dedenia, tiež v balíku

11. (1b) Deklarácia nestatickej a nefinalnej metódy rovnakej signatúry v podtype:
(a) Prekonáva pôvodnú metódu nadtypu
(b) Preťažuje pôvodnú metódu nadtypu
(c) Agreguje pôvodnú metódu nadtypu
(d) Dedí pôvodnú metódu nadtypu
(e) Nie je možná

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

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

public class B extends A {
void x() { System.out.print("Bx "); }
static void f() { System.out.print("Bf ");
      }
}
```

Čo sa vypíše po vykonaní týchto príkazov:

```
A o = new B();
o.x();
o.f();
((B) o).f();
((A) o).f();
```

(a) `Bx Af Bf Ax`
(b) `Bx Bf Bf Af`
(c) `Bx Bf Bf Ax`
(d) `Bx Bf Bx Af`
(e) `Bx Af Bx Af`
(f) `Bx Af Bf Af`
(g) `Ax Af Bx Af`
(h) `Ax Af Bf Af`

Základy objektovo-orientovaného programovania **A**
Ing. Ján Lang, PhD., UISI FIIT STU
Skúška - 19. januára 2016

Priezvisko:
Meno:
---

13. (1b) V prípade, že sa rovnomenné metódy (zdedené a pridané) v podtype líšia v parametroch hovoríme o:
(a) Prekonaní
(b) Preťažení
(c) Dedení
(d) Zapuzdrení
(e) Agregácii

14. (1b) V budovanej hierarchii tried môžeme pristupovať k skrytým atribútom a prekonaným metódam pomocou:
(a) Kľúčového slova super
(b) Kľúčového slova final
(c) Kľúčového slova import
(d) Kľúčového slova extend
(e) Kľúčového slova new

15. (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) f
(b) pcf
(c) pc
(d) fpc
(e) žiadneho z uvedených

16. (1b) Skutočnosť že trieda podtypu bez zásahu do zdedeného správania:
(a) Je nemožná
(b) Je možná
(c) Je vhodná
(d) Je neskutočná
(e) Je požadovaná
(f) Je nezaujímavá

17. (10b) V našom simulátore krajiny máme možnosť počítať a vyberať poplatky za využívanie množiny prvkov dopravnej infraštruktúry (spoplatnené úseky - úseky ciest, mosty, tunely a pod.) vybranými kategóriami dopravných prostriedkov (osobný automobil, autobus, traktor a pod). Výška poplatkov bude variabilná pre rôzne kategórie dopravných prostriedkov a času využitia spoplatnených úsekov (slabé dni, silné dni, sviatok a pod.). Poplatky bude definovať sadzobník. Systém umožní identifikáciu dopravného prostriedku, výpočet a výber poplatkov, bez nutnosti jeho zastavenia. 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:

	bodov	
1	2	d
2	2	d
3	3	c
4	2	c
5	1	f
6	2	a
7	2	d
8	3	e
9	1	b
10	2	e
11	1	a
12	3	f
13	1	b
14	1	a
15	3	c
16	1	f