Testové otázky

1. Kľúčové slovo *implements* súvisí s
    a. preťažovaním metód
    b. prekrývaním metód
    c. implementáciou interfejsov
    d. viacnásobnou dedičnosťou
2. Majme daný kód

```
System.out.print("Štart");
try {
  System.out.print("Ahoj svet");
  throw new FileNotFoundException();
} catch(EOFException e) {
  System.out.print("Nastal koniec súboru");
} catch(FileNotFoundException e) {
  System.out.print("Súbor nebol nájdený");
} finally {
  System.out.print("Koniec");
}
```

Ak vieme, že **EOFException** a **FileNotFoundException** sú podtriedami **IOException** a daný kód je súčasťou nejakej metódy nejakej triedy, ktoré tvrdenie je pravdivé a prečo?

    a. Kód sa neskompiluje
    b. Výpis kódu: Ahoj svet Nastal koniec súboru Koniec
    c. Výpis kódu: Ahoj svet Súbor nebol nájdený Koniec
    d. Výpis kódu: Ahoj svet Koniec

3. Majme hierarchiu dedičnosti výnimiek, ktoré sa týkajú chýb pri práci s indexami polí (array) a reťazcov (string)

```
Exception
    +-- RuntimeException
          +-- IndexOutOfBoundsException
                +-- ArrayIndexOutOfBoundsException
                +-- StringIndexOutOfBoundsException
```

Predstavme si metódu **žuvaj()**, ktorá hádže výnimky pri práci s indexami polí a reťazcov. Predpokladajme, že metóda **žuvaj()** nemá žiadne try-catch bloky. Ktoré z nasledovných tvrdení je správne?

    a. Hlavička metódy žuvaj() musí obsahovať "**throws ArrayIndexOutOfBoundsException, StringIndexOutOfBoundsException**".
    b. Ak metóda volajúca metódu **žuvaj()** odchytáva výnimku **IndexOutOfBoundsException**, odchytia sa výnimky aj pre chybné indexy polí i pre chybné indexy reťazcov.
    c. Ak hlavička metódy žuvaj() obsahuje "**throws IndexOutOfBoundsException"**, každá metóda, ktorá ju volá, musí mať v sebe try-catch blok.
    d. Hlavička metódy **žuvaj()** nemusí obsahovať žiadnu klauzulu throws.

4. Majme nasledovný kód

```
class B extends A {
  int getID() {
    return id;
  }
}
class C {
  public int name;
}

class A {
  C c = new C();

  public int id;
}
```

Ak *is-a* predstavuje vzťah dedičnosti a *has-a* vzťah kompozície, ktoré tvrdenia sú pravdivé?

    a. A is-a B
    b. C is-a A

c. A has-a C
d. B has-a A
e. B has-a C

5. Majme kód

```
class ČiernaSkrinka {
  public String pracuj(int x) {
    return "Hotovo";
  }
}
```

Ktorá metóda NIE JE správna v podtriedach ČiernejSkrinky?

a. public String pracuj(int x) { return "HOTOVO"; }
b. public int pracuj(int x) { return 42; }
c. public String pracuj(int x) { return "Hotovo"; }
d. public int pracuj(int x) { return "Hotovo"; }
e. public String pracuj(String s) { return "Hotovo"; }

6. Trieda **java.lang.Number** je rodičovskou triedou tried **java.lang.Integer**, **java.lang.Double** a mnohých iných. Trieda **java.lang.Number** implementuje interfejs **java.lang.Serializable**. Majme premennú **Integer číslo = new Integer(25)**. Ktoré tvrdenia sú pravdivé, resp. neobsahujú chybu?

a. číslo instanceof Object

b. Double d = číslo; Integer integer = (Integer) d;

c. Serializable serializable = číslo;
Integer integer = (Integer) serializable;
serializable.toString();

d. číslo = číslo + (25) (toto je jemný chyták)

Opravovacia časť

Dve hodiny pred odovzdaním projektu má nemenovaný študent projekt, v ktorom su doleuvedené triedy. Smutne sa pozerá do Eclipsu, ktorý mu podčiarkol značné množstvo kódu, ktoré je chybné. Popri tom je však v kóde aj niekoľko logických či návrhových chýb.

Projekt má totiž sčítavať obsahy trojuholníkov, čo vykonáva nesprávne (vracia mu to záporné čísla). Vžite sa do úlohy úbohého cvičiaceho, zistite všetky chyby a opravte ich. Ak vidíte chyby v návrhu, zdôvodnite ich slovne a navrhnite riešenie. Zároveň opravte logické chyby, aby to nemenovanému študentovi pracovalo tak, ako má.

```
public interface MaObsah {
	int dajObsah() {
		return -1;
	}
}
-----------------------------------------------
public class Usecka {
	int x, y, popis;

	public Usecka(int x, int y) {
		this.x = x;
		this.y = y;
	}

	public String toString() {
		return popis + " [" + x + ", " + y + "]";
	}
}
-----------------------------------------------
public class Trojuholnik extends Usecka implements MaObsah {
	private int z;

	 //urobi konstruktor s danymi bodmi
	public Trojuholnik(int A, int B, int C) {
		super();
```

```
            this.x = A;
            this.y = B;
            this.z = C;
      }

      // TODO: duri mi poradil Heronov vzorec
      // S = sqrt(s(s-a)(s-b)(s-c)), s = (a + b + c)
      // netusim ako ho pouzit
      public int getObsah() {
            return x * y * z;
      }
}
-----------------------------------------------
public class Tester {
  public static int spocitajObsahy(List<Usecka> trojuholniky)
{
    int q = 0;
    for (int i = 0; i < trojuholniky.size(); i++) {
      Trojuholnik t = trojuholniky.get(i);
      q = q + t.dajObsah();
    }
    return obsahy;
  }

  public static void main(String[] args) {
    Trojuholnik[] = new ArrayList<Usecka>();

    Trojuholnik t1 = new Trojuholnik(2, 4, 5);
    trojuholniky.add(t1);

    Trojuholnik t2 = new Trojuholnik(4, 5, 6);
    trojuholniky.add(t2);

    obsahy = spocitajObsahy(trojuholniky);
    // tu mi to vypisuje -2 netusim preco
    System.out.println(obsahy);
  }
}
```

Kreatívna časť

Firma ÚINF[1] nutne potrebuje nový portál, keďže na tú starú sa už nedá pozerať. Webový portál pozostáva zo sekcií, pričom každá sekcia obsahuje niekoľko článkov. Článok spadá do jedinej sekcie (napr. *Informácie pre študentov*), ale môže byť označený niekoľkými značkami (napr. *informatika*, *študenti*, *seminár*), podľa ktorých sa dajú články ďalej kategorizovať.
Každý článok obsahuje diskusiu, teda zoznam diskusných príspevkov, ktoré môžu mať stromovitú štruktúru. (Príspevok môže reagovať na iný príspevok).

V portále majú existovať autori článkov, diskutéri a jediný správca. Autor článku vie upravovať svoj článok, diskutér vie len prispievať a správca má neobmedzené možnosti. Samozrejme, roly sa môžu prekrývať – nie je vylúčené, že napr. diskutér má zároveň zverejnený článok.

Navrhnite pre tento informačný systém triedy, uveďte a slovne popíšte ich inštančné premenné a metódy. Zvážte a odôvodnite použitie dedičnosti (ak si myslíte, že dedičnosť je potrebná, slovne to zdôvodnite; ak si myslíte, že potrebná nie je, zdôvodnite to tiež).

Implementujte metódu (t. j. napíšte kód v programovacom jazyku), ktorá zistí zoznam všetkých príspevkov daného používateľa vo všetkých diskusiách. Uveďte aj triedu a relevantné inštančné premenné (v prípade, že ich metóda používa). Inak povedané, napíšte fragment triedy a úplnú implementáciu uvedenej metódy.
Naimplementujte tester (t. j. napíšte kód v programovacom jazyku) s metódou main(), v ktorej ukážete príklady použitia vami navrhnutého informačného systému a tried (predpokladajte, že všetky metódy všetkých tried už máte implementované). V testeri ukážte použitie minimálne 5 metód používaných v triedach informačného systému.

[1] Akákoľvek podobnosť so skratkami existujúcich inštitúcií, živých či mŕtvych, je čisto náhodná.