ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек за РТИ Београд 12.04.2014.

Колоквијум из
Објектно оријентисаног програмирања II

1) *(30 поена)* Одговорити концизно (по једна или две реченице) и прецизно на следећа питања:

а) Да ли (према спецификацији језика) имена променљивих у Јави могу да буду писана на ћирилици и зашто?

б) Како се одређује право приступа генерисаном (уграђеном) подразумеваном конструктору?

в) Да ли се из метода класе `I` изведене из `O` може приступити заштићеним нестатичким члановима класе `O` преко референце типа: (1) класе `O`, (2) класе `I`, (3) класе `A` изведене из класе `O`, (4) класе `II` изведене из класе `I`?

2) *(укупно 70 поена)* Написати на језику *Java* следећи пакет типова (грешке пријављивати изузецима опремљеним текстовима порука):

- *(20 поена)* ***Току*** може да се одреди реалан проточни капацитет изражен у m^3/s.
- ***Водени ток*** има назив који се задаје приликом стварања и може да се дохвати. Може да се одреди концентрација штетних материја у води (реалан број изражен у mg/m^3) и да ли је вода безбедна за пиће. Вода је безбедна за пиће ако је концентрација мања од 0,001 mg/m^3. Може да се састави текстуални опис према следећем формату: *назив* **(** *капацитет* **|** *безбедан* **)**. Не може да се копира.
- *(30 поена)* ***Поток*** је водени ток који има има проточни капацитет и концентрацију штетних материја у води, који се задају приликом стварања. Грешка је ако су капацитет или концентрација негативне вредности.
- ***Река*** је водени ток који се састоји од произвољног броја водених токова, који се додају након стварања реке. Проточни капацитет једнак је укупном проточном капацитету токова у саставу реке. Концентрација штетних материја се рачуна према следећој формули, где је *konc* тражена концентрација, а $konc_i$ и kap_i концентрација и капацитет *i*-тог тока, респективно:

$$konc = \frac{\sum_i konc_i \cdot kap_i}{\sum_i kap_i}.$$

- *(20 поена)* Хидрографски ***билтен*** има назив и садржи задати број водених токова. Капацитет се задаје приликом стварања, а појединачни водени токови се накнадно додају. Грешка је ако се билтен препуни. Може да се састави текстуални опис тако што се најпре наведе назив, а затим описи садржаних водених токова, по један у сваком реду. Може да се копира, при чему се не праве копије водених токова.

Написати класу са главним програмом у којем се ствара неколико водених токова различитог типа и хидрографски билтен у који се направљени водени токови додају, а затим испише билтен.

НАПОМЕНЕ: **а)** Колоквијум траје **120** минута.

б) Рад се предаје искључиво у вежбанци за испите (-5 поена за неадекватну вежбанку). Није дозвољено имати поред себе друге листове папира, нити уз себе имати мобилни телефон, без обзира да ли је укључен или искључен.

в) Водити рачуна о уредности. Нечитки делови текста ће бити третирани као непостојећи. Решења задатака навести по горњем редоследу (-1 поен за лош редослед). Препоручује се рад обичном графитном оловком.

г) Резултати колоквијума биће објављени на *Web*-у на адреси: `home.etf.rs/~kraus/` (одреднице: *настава* | <име предмета> | *оцене* | *колоквијуми*).

```
// Tok.java
package hidrografija;
public interface Tok { double kapac(); }
```

```
// Vodeni.java
package hidrografija;
public abstract class Vodeni implements Tok {
  private String naziv;
  public Vodeni(String _n) { naziv = _n; }
  public abstract double konc();
  public boolean bezbedno() { return konc() < 0.001; }
  public String toString()
    { return naziv+"("+kapac()+"|"+bezbedno()+")"; }
  public Vodeni clone()
     throws CloneNotSupportedException {
  throw new CloneNotSupportedException();
  }
}
```

```
// GNegVred.java
package hidrografija;
public class GNegVred extends Exception {
  public GNegVred()
    { super("*** Negativna vrednost!"); }
}
```

```
//Potok.java
package hidrografija;
public class Potok extends Vodeni {
  private double kapac;
  private double konc;
  public Potok(String ime, double kap,double kon)
      throws GNegVred {
    super(ime);
     if (kap<0 || kon<0) throw new GNegVred();
    kapac = kap;
    konc = kon;
  }
  public double kapac() { return kapac; }
  public double konc() { return konc; }
}
```

```
// Reka.java
package hidrografija;
public class Reka extends Vodeni {
  private static class Elem {
    Elem sled;
    Vodeni tok;
    Elem(Vodeni _t) { tok = _t; }
  }
  private Elem prvi, posl;
  public Reka(String ime) { super(ime); }
  public Reka dodaj(Vodeni t) {
    Elem novi = new Elem(t);
    if (prvi == null) { prvi = novi; }
      else { posl.sled = novi; }
    posl = novi;
    return this;
  }
  public double kapac() {
    double kapac = 0;
    for (Elem e=prvi; e!=null; e=e.sled)
      kapac += e.tok.kapac();
    return kapac;
  }
  public double konc() {
    double kapac = 0;
    double konc = 0;
    for (Elem e=prvi; e!=null; e=e.sled){
      kapac += e.tok.kapac();
      konc += e.tok.kapac()*e.tok.konc();
    }
    return konc/kapac;
  }
}
```

```
// GBiltenPun.java
package hidrografija;
public class GBiltenPun extends Exception {
  public GNegVred() { super("*** Bilten je pun!"); }
}
```

```
// Bilten.java
package hidrografija;
public class Bilten implements Cloneable{
  private String naziv;
  private Vodeni[] tokovi;
  private int pop;
  public Bilten(String naz, int kap) {
    naziv = naz; tokovi=new Vodeni[kap];
  }
  public Bilten dodaj(Vodeni t) throws GBiltenPun {
    if (pop == tokovi.length) throw new GBiltenPun();
    tokovi[pop++] = t;
    return this;
  }
  public String toString() {
    StringBuffer s = new StringBuffer();
    s.append(naziv).append("\n");
    for (Vodeni v: tokovi) s.append(v).append("\n");
    return s.toString();
  }
  public Bilten clone() {
    try {
      Bilten b = (Bilten)super.clone();
      b.tokovi = tokovi.clone();
      return b;
    }
    catch(CloneNotSupportedException e)
      { return null; }
  }
}
```

```
// Glavni.java
import hidrografija.*;
public class Glavni {
  public static void main(String[] argv){
    try {
      Bilten b = new Bilten("Izvestaj", 5);
      Potok p1 = new Potok("Gorski", 0.3, 0);
      Potok p2 = new Potok("Gradski", 0.4, 0.003);
      Reka r1  = new Reka("Topciderska")
                     .dodaj(p1).dodaj(p2);
      Potok p3 = new Potok("Bistri", 0.7, 0);
      Reka r2  = new Reka("Sava")
                     .dodaj(r1).dodaj(p3);
      b.dodaj(p1).dodaj(p2).dodaj(p3)
                 .dodaj(r1).dodaj(r2);
      System.out.println(b);
    } catch(Exception g) {
      System.out.println(g);
    }
  }
}
```

```
Izvestaj
Gorski(0.3|true)
Gradski(0.4|false)
Bistri(0.7|true)
Topciderska(0.7|false)
Sava(1.4|true)
```