ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек за СИ

Београд 23.04.2014.

Други колоквијум из
Објектно оријентисаног програмирања II

1) *(30 поена)* Одговорити концизно (по једна или две реченице, или "да"/"не" где се не тражи образложење) и прецизно на следећа питања:

а) Ако је `I` интерфејс, да ли је дозвољено (1) дефинисати референцу типа интерфејса `I i;` (2) проследити референцу на интерфејс `I` као аргумент некој методи (3) створити објекат типа интерфејса помоћу `new I()`?

б) Да ли се и зашто нека класа на Јави може дефинисати у неком изразу?

в) Навести начине на које се нит блокира тако што сама себе суспендује и одговарајуће начине како се деблокира.

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

- *(30 поена)* ***Регистровано*** је нешто чему се може дохватити регистарска ознака (ниска).
- Регистровано ***возило*** има аутоматски генерисану регистарску ознаку која може да се дохвати. Регистарска ознака садржи једно од "BG", "NS" и "NI" иза чега стоји случајан троцифрен број.
- ***Камера*** има једнозначан аутоматски генерисан целобројан идентификатор. Она на захтев сними задато возило, када памти возило и време снимања (мерено од почетка извршавања програма) и може да испоручи снимак возила (ниска) у облику **[***идКам***:***време***]***регБр*.
- Саобраћајна ***трака*** садржи произвољан број возила у реду и камеру. Може да се дода возило на крај реда и да се узме возило са почетка реда (`null` ако је ред празан). Приликом узимања возила из траке возило се сними камером траке и снимак се памти до следећег снимања. Ако је ред при покушају узимања возила празан, снимак је празан. Може да се дохвати последњи снимак.
- *(30 поена)* Активна ***саобраћајница*** ствара задат број саобраћајних трака (може и 0). Негативан број трака је грешка. На сваких *T* sec, где је *T* случајно време између задатих *minT* и *maxT*, извршава се нека активност. Рад саобраћајнице може да се привремено обустави, поново покрене или трајно заустави.
- ***Улица*** је саобраћајница са неколико трака чија је активност додавање возила у случајно изабрану траку. Може да се дохвати снимак возила које се том приликом узима из случајно одабране траке улице, при чему се узето возило одбацује.
- ***Раскрсница*** је саобраћајница која има задату улицу, нема ни једну траку и чија се активност састоји од дохватања снимка возила (које је управо напустило задату улицу) уз исписивање снимка, ако снимак постоји.

(10 поена) Написати на језику *Java* програм (класу с главном функцијом) који направи једну улицу и једну раскрсницу, покрене их, после неког врмена их заустави, а затим трајно обустави рад улице и раскрснице. Улица и раскрсница имају: *minT*=0.1s и *maxT*=0.3s.

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*).

```
// Registrovano.java
package saobracaj;
interface Registrovano { String regOzn(); }
```

```
// Vozilo.java
package saobracaj;
public class Vozilo implements Registrovano  {
  private static final String[]
    mesta = {"BG", "NS", "NI"};
  private String reg;
  public Vozilo() {
    reg = mesta[(int)(Math.random()*mesta.length)]+
          (int)(Math.random()*10) +
          (int)(Math.random()*10) +
          (int)(Math.random()*10);
  }
  public String regOzn() { return reg; }
}
```

```
// Kamera.java
package saobracaj;
public class Kamera {
  private static int posId = 0;
  private int id = ++ posId;
  private Vozilo vozilo;
  private static long t0 =
    System.currentTimeMillis();
  private long t;
  public Kamera snimi(Vozilo v) {
    vozilo = v;
    t = System.currentTimeMillis() - t0;
    return this;
  }
  public String snimak()
    { return "[" + id + "," + t + "]" +
             vozilo.regOzn(); }
}
```

```
// Traka.java
package saobracaj;
public class Traka {
  private Kamera kam;
  private String snimak;
  private Elem prvi = null, posl = null;
  private class Elem {
    Vozilo voz; Elem sled = null;
    Elem(Vozilo v) {
      voz = v;
      if (prvi == null) prvi = this;
        else posl.sled = this;
      posl = this;
    }
  }
  public Traka() { kam = new Kamera(); }
  public synchronized void stavi(Vozilo v)
    { new Elem(v); }
  public synchronized Vozilo uzmi() {
    snimak = null;
    if (prvi == null) return null;
    Vozilo v = prvi.voz;
    if ((prvi = prvi.sled) == null) posl = null;
    kam.snimi(v); snimak = kam.snimak();
    return v;
  }
  public String snimak() { return snimak; }
}
```

```
// NegBr.java
package saobracaj;
public class NegBr extends Exception {
  public String toString()
    { return "Negativan broj!"; }
}
```

```
// Saobracajnica.java
package saobracaj;
public abstract class Saobracajnica extends Thread{
  protected Traka[] trake;
  private long tMin, tMax;
  private boolean radi = false;
  protected abstract void radnja();
  public Saobracajnica(int brTr, long t1, long t2)
                                    throws NegBr {
    if (brTr<0) throw new NegBr();
    tMin=t1; tMax=t2;
    trake = new Traka [brTr];
    for (int i=0; i<brTr; trake[i++]=new Traka());
  }
  public void run() {
    try {
      while (!interrupted()) {
        synchronized (this) { if (!radi) wait(); }
        sleep((long)(tMin +
                     Math.random()*(tMax-tMin)));
        if (radi) radnja();
      }
    } catch (InterruptedException g) {}
  }
  public synchronized void kreni()
    { radi = true; notify(); }
  public void stani() { radi = false; }
  public void zavrsi() { interrupt(); }
}
```

```
// Ulica
package saobracaj;
public class Ulica extends Saobracajnica {
  public Ulica(int brTr, long t1, long t2)
                                      throws NegBr
    { super(brTr, t1, t2); start(); }
  protected void radnja() {
    trake[(int)(Math.random()*trake.length)].
       stavi(new Vozilo());
  }
  public String snimak(){
   Traka traka =
     trake[(int)(Math.random()*trake.length)];
   traka.uzmi();
   return traka.snimak();
  }
}
```

```
// Raskrsnica
package saobracaj;
public class Raskrsnica extends Saobracajnica{
  private Ulica ulica;
  public Raskrsnica(Ulica u, long t1, long t2)
                                      throws NegBr
    { super(0, t1, t2); ulica = u; start(); }
  protected void radnja() {
   String snimak = ulica.izaslo();
   if (snimak != null)
   System.out.println(snimak);
  }
}
```

```
// Program.java
import saobracaj.*;
public class Program{
  public static void main(String[] varg) {
   Ulica ul; Raskrsnica ras;
    try {
      ul = new Ulica(3, 100, 300);
      ras = new Raskrsnica(ul, 100,300);
    } catch (NegBr i) {return;}
    ul.kreni(); ras.kreni();
    try { Thread.sleep(3000); }
    catch (InterruptedException g) {}
    ul.stani(); ras.stani();
    ul.zavrsi(); ras.zavrsi();
    try { ul = new Ulica(-1, 100, 300); }
      catch (NegBr i) { System.out.println(i); }
  }
}
```

```
[3,390]NS271
[1,1265]NS218
[3,1453]NI018
[2,1797]BG509
[1,2500]NI033
[1,2703]NI623
[1,2968]BG425
Negativan broj!
```