ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек за СИ Београд 28.04.2017.

Други колоквијум из Објектно оријентисаног програмирања II

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

- *(15 поена)* Активно ***возило*** може бити у једном од 3 стања – *MIROVANJE, KRETANJE, SERVISIRANJE*. Ствара се у стању мировања са задатом минималном и максималном брзином, периодом ажурирања километраже (у секундама) и километражом између сервисирања. Могу да се дохвате минимална и максимална брзина, тренутно стање возила, текућа (од последњег сервисирања) и укупна километража и да се постави тренутна брзина возила. Возило периодично ажурира километражу (и текућу и укупну) увећањем за производ текуће брзине и периода ажурирања. Уколико текућа километража пређе километражу између сервисирања, возило се привремено зауставља, прелази у стање сервисирања и текућа километража му се анулира. Возило може да се привремено заустави (пређе у стање мировања), поново покрене (пређе у стање кретања) и трајно заустави (пређе или се задржи у стању мировања).
- *(40 поена)* Активна ***особа*** има јединствен аутоматски генерисан целобројни идентификатор, задато име, као и минимално и максимално време за које може да изврши неки понављајући посао. Може да се дохвати име особе, да се испита да ли тренутно ради, да јој се привремено заустави рад, настави рад, трајно прекине рад, као и да јој се састави текстуални опис у облику *име_особе* **(***ид_особе***)**.
- **Механичар** је особа чији је посао сервисирање возила. Има бројач сервиса који се инкрементира при сваком сервису. Може да се састави текстуални опис у облику **`M –`** *особа***:** *бројач_сервиса*.
- **Возач** је особа која има своје возило и тим механичара. Ствара се са задатим капацитетом тима. Возачу може да се промени возило (само ако је привремено заустављено), може да ангажује механичара (грешка је ако је тим већ попуњен). Возач може да почне вожњу само ако има возило и попуњен тим механичара, може да стане са вожњом (при чему привремено заустави и возило) и да прекине са радом, при чему се возило трајно зауставља, а механичари прекидају са радом. Возачев посао се састоји од мењања брзине возила и очитавања текућег стања возила. Уколико очита стање сервисирања, он привремено зауставља возило и јавља својим механичарима да крену (наставе) са послом. Када и последњи механичар заврши са послом, возач поново покреће возило и наставља са вожњом. Може да се састави текстуални опис у облику **`V`** – *особа* **`:`** *ук_прешао*, а затим у следећем реду тим његових механичара.
- *(15 поена)* **Трка** има задат максимални број возача који могу да се пријаве за трку (ако се места попуне, накнадне пријаве се игноришу). Трка може да се започне и заврши. Може да се састави текстуални опис у облику **`--Trka--`**, а затим сваки учесник трке у посебном реду.

Приложена је класа са главном функцијом која симулира једну трку са 2 учесника.

НАПОМЕНЕ: **а)** Колоквијум траје **120** минута. Време за израду задатка је **100** минута.
б) Рад се предаје искључиво на предвиђеном мрежном диску.
в) Називе типова ускладити са називима апстракција из текста задатка, али користити латинично писмо и велико почетно слово.
г) На располагању је приступ *Web* адреси: `https://docs.oracle.com/javase/8/docs/api/`. Није дозвољено имати поред себе друге материјале, нити уз себе имати електронске уређаје, без обзира да ли су укључени или искључени.
д) Резултати колоквијума биће објављени на *Web*-у на адреси: `http://rti.etf.bg.ac.rs/rti/ir2oo2/index.html`

```java
// Vozilo.java
package demo;
public class Vozilo extends Thread {
   public enum Stanje {MIROVANJE, SERVIS, KRETANJE};
   private Stanje stanje = Stanje.MIROVANJE;
   private volatile int vmin, vmax;
   private int brzina, presao, ukPut, servis, tAzur;
   public Vozilo(int min, int max, int s, int t) {
      vmin = min; vmax = max; servis = s; tAzur = t; }
   public int dohvMinBrzina() { return vmin; }
   public int dohvMaxBrzina() { return vmax; }
   public synchronized Stanje dohvStanje() {return stanje;}
   public synchronized int dohvUkPut() { return ukPut; }
   public synchronized void postaviBrzina(int d) {brzina=d;}
   public synchronized void kreni() {
      stanje = Stanje.KRETANJE; notify();   }
   public synchronized void stani() {
      brzina = 0; stanje = Stanje.MIROVANJE; }
   public synchronized void prekini() { interrupt(); }
   public void run() {
      try {
         while(!interrupted()) {
            synchronized (this) {
               while(stanje == Stanje.MIROVANJE) wait(); }
            sleep(tAzur*1000);
            synchronized (this) {
               presao += brzina * tAzur;
               ukPut += brzina * tAzur;
               if(presao >= servis) {
                  stanje = Stanje.SERVIS;
                  wait(); presao = 0;
               }}}
      } catch (InterruptedException e) {} }
}
```

```java
// Osoba.java
package demo;
public abstract class Osoba extends Thread {
   private String ime; private static int ID;
   private int id = ++ID; private int tmin, tmax;
   protected boolean radi;
   public Osoba(String name, int min, int max) {
      ime = name; tmin = min; tmax = max; }
   public String dohvIme() { return ime; }
   public synchronized boolean dohvRadi() { return radi; }
   public String toString() {
      return ime + "(" + id + ")"; }
   public abstract void zapocniPosao();
   public abstract void zavrsiPosao()
                              throws InterruptedException;
   public void run() {
      try {
         while(!interrupted()) {
            synchronized (this) { while(!radi) wait(); }
            zapocniPosao();
            int vreme = (int)(tmin + (Math.random() *
                                          (tmax - tmin)));
            sleep(vreme); zavrsiPosao(); }
      } catch (InterruptedException e) {} }
   public synchronized void kreni() {
      radi = true; notify(); }
   public synchronized void stani() { radi = false; }
   public void prekini() { interrupt(); }
}
```

```java
// Mehanicar.java
public class Mehanicar extends Osoba {
   private int srv;
   public Mehanicar(String ime, int min, int max) {
      super(ime, min, max); }
   public void zapocniPosao() {}
   public void zavrsiPosao() throws InterruptedException {
      synchronized (this) {
         srv++; radi = false; notify(); }}
   public String toString() {
      return "M-" + super.toString() + ": " + srv; }
}
```

```java
// Vozac.java
package demo;
public class Vozac extends Osoba {
   private Vozilo voz; private Mehanicar []tim;
   private int br;
   public Vozac(String ime, int min, int max, int kap) {
      super(ime, min, max); tim = new Mehanicar [kap]; }
   public synchronized void promeniVozilo(Vozilo v) {
      if(!radi) {  voz = v;  v.start(); }}
   public void angazujRadnika(Mehanicar r) throws GTimPun {
      if(br>=tim.length) throw new GTimPun(dohvIme(), 1);
      tim[br++] = r;  r.start(); }
   public synchronized void pokreni()
                              throws GTimPun, GNemaVozilo {
      if(voz==null) throw new GNemaVozilo(dohvIme());
      if(br!=tim.length) throw new GTimPun(dohvIme(), 0);
      voz.kreni(); kreni(); }
   public synchronized void stani() {
      radi = false; voz.stani(); }
   public void prekini() {
      voz.interrupt();
      for(int i=0; i<br; i++) tim[i].prekini();
      super.prekini(); }
   public void zapocniPosao() {
      voz.postaviBrzina(voz.dohvMinBrzina() +
               ((int)(Math.random() * (voz.dohvMaxBrzina() -
               voz.dohvMinBrzina()))));  }
   public void zavrsiPosao() throws InterruptedException {
      synchronized (voz) {
         if(voz.dohvStanje() == Vozilo.Stanje.SERVIS) {
            voz.stani();
            for(int i=0; i<br; i++) tim[i].kreni();
            for(int i=0; i<br; i++)
               while(tim[i].dohvRadi())
                  synchronized (tim[i]) { tim[i].wait(); }
            voz.kreni(); }}}
   public synchronized String toString() {
      StringBuilder sb = new StringBuilder("V-" +
               super.toString() + " : " + voz.dohvUkPut() +
                                                "\nTim: ");
      for(int i=0; i<br; i++) {
         sb.append(tim[i]);
         if(i<br-1) sb.append(", "); }
      return sb.toString(); }
}
```

```java
// Trka.java
package demo;
public class Trka {
   private Vozac []vozaci;
   private int br;
   public Trka(int kap) { vozaci = new Vozac [kap]; }
   public void prijaviVozaca(Vozac voz) {
      if(br<vozaci.length) vozaci[br++] = voz;     }
   public void zapocniTrku() throws GTimPun, GNemaVozilo {
      for(int i=0; i<br; i++) {
         vozaci[i].start(); vozaci[i].pokreni(); }}
   public void zavrsiTrku() {
      for(int i=0; i<br; i++) {
         vozaci[i].prekini();
         try {  vozaci[i].join();
         } catch (InterruptedException e) {}}}
   public String toString() {
      StringBuilder sb = new StringBuilder("--Trka--\n");
      for(int i=0; i<br; i++)
         sb.append(vozaci[i] + "\n");
      return sb.toString(); }
}
```

```java
// GTimPun.java
package greske;
public class GTimPun extends Exception {
   private String poruka; private int rezim;
   public GTimPun(String str, int r) {
      poruka = str; rezim = r; }
   public String toString() {
      return "Tim vozaca " + poruka + (rezim==1?"je":"nije")
                                          + " popunjen!"; }
}
```

```java
// GNemaVozilo.java
package greske;
public class GNemaVozilo extends Exception {
   private String vozac;
   public GNemaVozilo(String str) { vozac = str; }
   public String toString() {
      return "Vozac " + vozac + " nema vozilo!"; }
}
```

```java
// Program.java
import demo.*;
public class Program {
   public static void main(String []args) {
      Trka trka = new Trka(2);
      try {
         Vozilo v1 = new Vozilo(50, 200, 500, 1),
         v2 = new Vozilo(50, 200, 500, 1);
         Vozac d1 = new Vozac("Misko", 1, 3, 1),
         d2 = new Vozac("Zile", 1, 3, 1);
         Mehanicar w1 = new Mehanicar("Meca", 2, 3), w2 =
         new Mehanicar("Neca", 2, 3);
         d1.promeniVozilo(v1); d2.promeniVozilo(v2);
         d1.angazujRadnika(w1); d2.angazujRadnika(w2);
         trka.prijaviVozaca(d1); trka.prijaviVozaca(d2);
         trka.zapocniTrku(); Thread.sleep(10000);
      } catch (Exception e) { System.err.println(e); }
      finally {trka.zavrsiTrku(); System.out.println(trka);}
   }
}
```

```
//primer izvrsavanja

--Trka--
V-Misko(1) : 1010
Tim: M-Meca(3): 1
V-Zile(2) : 1163
Tim: M-Neca(4): 2
```