ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек за ИР Београд 23.04.2017.

Колоквијум из
Објектно оријентисаног програмирања II

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

а) Навести пример за сваки од начина на који се изван пакета `p` може приступити типу `T` који је дефинисан у пакету `p`.

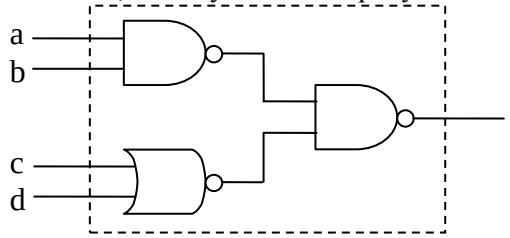
б) Навести називе класа корена одговарајућих хијерархија класа: (1) свих изузетака, (2) корисничких изузетака, (3) системских непроверених изузетака и (4) системских грешака.

в) Да ли се подацима у непосредно окружујућем досегу може приступити директним именовањем и која су ограничења за следеће случајеве: (1) из статичке угнежђене класе, (2) из унутрашње класе, (3) из локалне класе, (4) из анонимне класе?

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

- *(30 поена)* Логички **елемент** има јединствен аутоматски генерисан целобројни идентификатор, задат број улазних логичких сигнала и један излазни логички сигнал (видети ниже). Ствара се без улазних сигнала, који се касније могу додавати појединачно задавањем индекса (грешка је ако индекс није исправно задат). Може да се дохвати излазни сигнал и назив елемента. Елемент може да постави вредност излазног сигнала на основу улазних сигнала (уколико неки улазни сигнал не постоји, сматрати да се на улазу налази логичка вредност *false*). Може да се састави текстуални опис у облику *назив_id*`[`*улазни_сигнал*`, ...,`*улазни_сигнал*`][`*излазни_сигнал*`]`(на месту без сигнала текст је **`prazno`**).
- **НИ** (NAND) је логички елемент који за вредност излазног сигнала поставља вредност логичке операције НИ, чији су операнди вредности свих улазних сигнала. Назив је **`NAND`**.
- **НИЛИ** (NOR) је је логички елемент који за вредност излазног сигнала поставља вредност логичке операције НИЛИ, чији су операнди вредности свих улазних сигнала. Назив је **`NOR`**.
- Логички **сигнал** има јединствен аутоматски генерисан целобројни идентификатор, логичку вредност (која је подразумевано *false*, а може да се постави и дохвати) и произвољан број логичких елемената на чије улазе је повезан. Ствара се без логичких елемената, а касније се може повезивати на улазе логичких елемената са задатим индексом, појединачно. Приликом сваке промене вредности сигнала, постављају се излази свих логичких елемената на чије је улазе повезан. Такође, када се сигнал повеже на улаз неког логичког елемента, поставља се излаз тог елемента. Може да се састави текстуални опис сигнала у облику **`signal_`***id***`(`***vrednost_signala***`)`**.
- *(30 поена)* Логичка **мрежа** има задат број логичких елемената и произвољан број улазних и излазних сигнала. Ствара се без елемената, који се касније могу додавати појединачно (грешка је ако мрежа већ садржи највећи могући број елемената), без улазних и излазних сигнала. Могуће је повезати излазни сигнал првог задатог елемента мреже на улаз са задатим индексом другог задатог елемента мреже. Елементи који се повезују се задају својим индексима у мрежи. Грешка је уколико ниједан од елемената не постоји. Уколико не постоји први задати елемент, на улаз са задатим индексом другог задатог елемента се повезује нови подразумевани улазни сигнал мреже. Уколико не постоји други задати елемент излазни сигнал првог задатог елемента постаје нови излаз мреже. Може да се дохвати улазни сигнал мреже са задатим индексом и излазни сигнал мреже са задатим индексом (индекси почињу од 0). Текстуални опис садржи описе свих логичких елемената, сваки у посебном реду.

(10 поена) Написати на језику *Java* **програм** који направи мрежу са слике, постави вредности улазних сигнала на a = *true*, b = *false*, c = *true*, d = *false* и испише мрежу на стандардни излаз, а потом постави вредности улазних сигнала на a =*false*, b = *false*, c = *true*, d =*true* и испише мрежу на стандардни излаз. Користити константне параметре (не треба ништа учитавати)**.**



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: `http://rti.etf.bg.ac.rs/rti/ir2oo2/index.html`

```
// Element.java
package elementi;
public abstract class Element {
   private static int ukId = 0; private int id = ++ukId;
   protected Signal[] ulazi; protected Signal izlaz;
   public Element(int n)
    {ulazi = new Signal[n]; izlaz = new Signal();}
   public void postaviUlaze(int i, Signal s) throws GOpseg {
      if (i< 0 || i>= ulazi.length) throw new GOpseg();
      if(ulazi[i]!=null){
         ulazi[i].ukloni(this);
      }
      ulazi[i] = s;
   }
   public Signal dohvIzlaz() {return izlaz;}
   public abstract String ime();
   public abstract void propagacija();
   public String toString() {
      StringBuilder s = new StringBuilder();
      s.append(ime() + "_" + id + "[");
      for (int i = 0; i<ulazi.length; i++) {
         if (i > 0) {s.append(", ");}
         if (ulazi[i] != null) {s.append(ulazi[i]);}
         else {s.append("prazno");}
       }
      s.append("]"+ "[" + izlaz + "]");
      return s.toString();
   }
}
```

```
//GOpseg.java
package elementi;
public class GOpseg extends Exception {
   public String toString()
   {return "Greska zbog neispravno zadatog indeksa";}}
```

```
// NI.java
package elementi;
public class NI extends Element {
   public NI(int n) {super(n);}
   public void propagacija() {
      for (Signal ulaz : ulazi) {
         if (ulaz == null || !ulaz.vrednost())
           {izlaz.postaviVrednost(true);return;}}
      izlaz.postaviVrednost(false);
   }
   public String ime() {return "NI";}
}
```

```
// NILI.java
package elementi;
public class NILI extends Element {
   public NILI(int n) {super(n);}
   public void propagacija() {
      for (Signal ulaz : ulazi) {
         if (ulaz != null && ulaz.vrednost())
           {izlaz.postaviVrednost(false); return;}}
      izlaz.postaviVrednost(true);
   }
   public String ime() { return "NILI";}
}
```

```
// Signal.java
package elementi;
public class Signal {
   private static int ukId = 0; private int id = ++ukId;
   private boolean vrednost; private Elem prvi, posl;
   private static class Elem {
      Element elem; Elem sled;
      Elem(Element e) {elem = e;}
   }
   public void postaviVrednost(boolean v) {
      vrednost = v;
      for (Elem tek = prvi; tek != null; tek = tek.sled)
      {tek.elem.propagacija();}
   }
   public boolean vrednost() {return vrednost;}
   public void povezi(Element e, int i) throws GOpseg {
      e.postaviUlaze(i, this);
      Elem povezan = new Elem(e);
      if (prvi == null) {prvi = povezan;}
        else {posl.sled = povezan;}
      posl = povezan; e.propagacija();
   }
   public void ukloni(Element e){
      Elem tek = prvi, preth = null;
      for(; tek != null; tek = tek.sled){
         if(tek.elem == e) break;
         preth = tek;
       }
      if(tek == null) return;
      if(preth != null) preth.sled = tek.sled;
         else prvi = tek.sled;
      if(posl == tek) posl = preth;
   }
   public String toString()
      {return"signal_" + id + "(" + vrednost + ")";}
```

```
// Kolo.java
package elementi;
public class Kolo {
   private Elem prviU, poslU, prviI, poslI;
   private Element[] elementi; private int pop;
   private static class Elem {
      Signal signal; Elem sled; Elem(Signal s) {signal = s;}
   }
   public Kolo(int n) {elementi = new Element[n];}
   public void dodajElem(Element e) throws GPun {
      if (pop == elementi.length) throw new GPun();
      elementi[pop++] = e;
   }
   private Element dohvElem(int i) {
      if (i< 0 || i>= elementi.length) {return null;}
      return elementi[i];
   }
   public void povezi(int ind1, int ind2, int ind)
throws GOpseg, GElement {
      Element e1 = dohvElem(ind1), e2 = dohvElem(ind2);
      if (e1 == null && e2 == null) throw new GElement();
      if (e1 == null) {postaviUlaz(e2, ind);}
         else if (e2 == null) {postaviIzlaz(e1);}
            else {e1.dohvIzlaz().povezi(e2, ind);}
   }
   private void postaviUlaz(Element e, int i) throws GOpseg {
      Signal ulaz = new Signal(); ulaz.povezi(e, i);
      Elem povezan = new Elem(ulaz);
      if (prviU == null) {prviU = povezan;}
        else {poslU.sled = povezan;}
      poslU = povezan;
   }
   private void postaviIzlaz(Element e) {
      Elem povezan = new Elem(e.dohvIzlaz());
      if (prviI == null) {prviI = povezan;}
        else {poslI.sled = povezan;}
      poslI = povezan;
   }
   public Signal dohvUlaz(int ind)
      {return dohvSignal(ind, prviU);}
   public Signal dohvIzlaz(int ind)
       {return dohvSignal(ind, prviI);}
   private Signal dohvSignal(int ind, Elem prvi) {
      Elem tek = prvi;   int i = ind;
      for (; tek != null && i> 0; i--, tek = tek.sled);
      if (i == 0) {return tek.signal; } return null;
   }
   public String toString() {
      StringBuilder sb = new StringBuilder();
      for (int i = 0; i<pop; i++)
         {sb.append(elementi[i] + "\n");}
      return sb.toString();
}}
```

```
// GPun.java
package elementi;
public class GPun extends Exception {
   public String toString()
     {return "Greska zbog prepunjavanja niza";}}
```

```
// GElement.java
package elementi;
public class GElement extends Exception {
   public String toString()
     {return "Greska zbog nepostojecih elemenata";}}
```

```
// Test.java
package elementi;
public class Test {
   public static void main(String[] args) {
      Kolo c = new Kolo(5);
      try {
         c.dodajElem(new NI(2)); c.dodajElem(new NILI(2));
         c.dodajElem(new NI(2));
         c.povezi(-1, 0, 0); c.povezi(-1, 0, 1);
         c.povezi(-1, 1, 0); c.povezi(-1, 1, 1);
         c.povezi(0, 2, 0);  c.povezi(1, 2, 1);
         c.povezi(2, -1, 0);
          c.dohvUlaz(0).postaviVrednost(true);
         c.dohvUlaz(1).postaviVrednost(false);
         c.dohvUlaz(2).postaviVrednost(true);
         c.dohvUlaz(3).postaviVrednost(false);
         System.out.println(c);
         c.dohvUlaz(0).postaviVrednost(false);
         c.dohvUlaz(1).postaviVrednost(true);
         c.dohvUlaz(2).postaviVrednost(false);
         c.dohvUlaz(3).postaviVrednost(false);
         System.out.println(c);
       } catch (GPun|GOpseg|GElement e) {System.out.println(e);}
   }}
```

```
// primer izvrsavanja
NI_1[signal_4(true), signal_5(false)][signal_1(true)]
NILI_2[signal_6(true), signal_7(false)][signal_2(false)]
NI_3[signal_1(true), signal_2(false)][signal_3(true)]

NI_1[signal_4(false), signal_5(true)][signal_1(true)]
NILI_2[signal_6(false), signal_7(false)][signal_2(true)]
NI_3[signal_1(true), signal_2(true)][signal_3(false)]
```