Segurtasun- eta bizitasun-propietateak

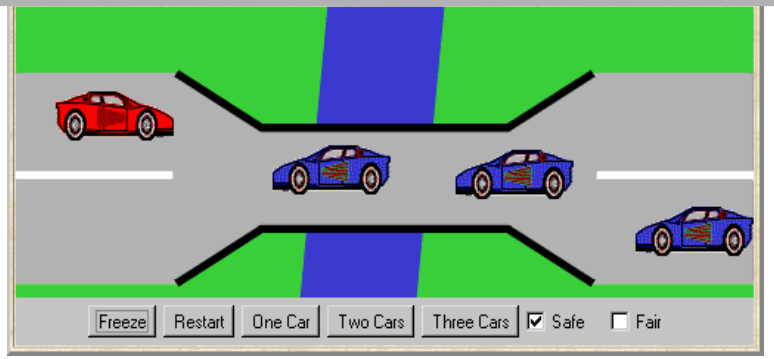
7.1 Segurtasun- eta bizitasun-propietateak

- Propietatea: Exekuzio posible guztientzat egiazko atributua
- Segurtasuna: Ez da ezer **txarrik** gertatzen
- Bizitasuna: Zerbait **ona** *noizbait* gertatzen da

Helburua:
segurtasun- eta bizitasun-propietateak betetzea.

7.2 Segurtasuna

Segurtasun-propietatea: Ez da ezer txarrik gertatzen



Adibidea: Bide bakarreko zubiaren problema

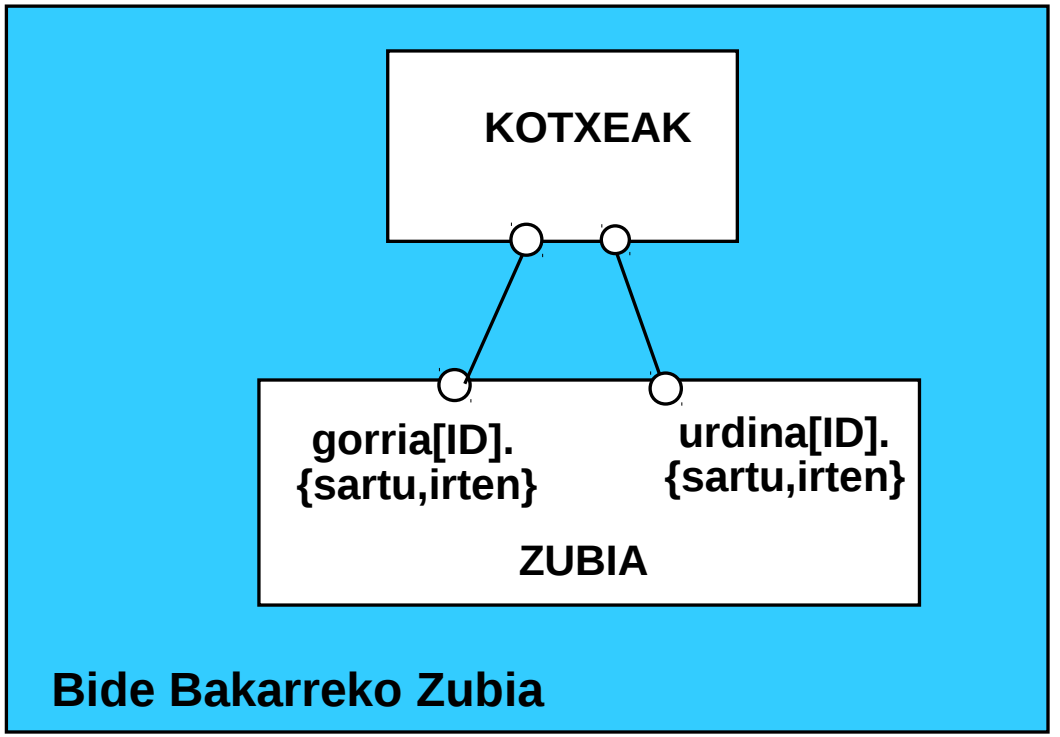
Ibai baten gaineko zubi bat oso hestua da,
eta soilik kotxe bat sartzen da zabaleran.

Horregatik kotxeak zubian konkurrenteki mugi daitezke,
soilik **norabide berean** mugitzen badira.

Segurtasun bortxaketa bat gertatuko da
aldi berean bi kotxe zubian sartzen badira norabide ezberdinetan.

Bide bakarreko zubia - eredua

- Gertaera edo ekintza interesgarriak identifikatu:
  - **sartu eta irten**
- Prozesuak identifikatu:
  - **kotxeak eta zubia**
- Propietateak identifikatu:
  - **norabidebakarra**
- Prozesu bakoitza eta elkarekintzak definitu:
  - **(egitura)**



Bide bakarreko zubia - KOTXEAK eredua

```
const N  = 3      // kotxe mota bakoitzeko kopurua
range T  = 0..N   // kotxe kontagailuaren mota
range ID = 1..N   // kotxeen zenbakiak

KOTXEA = (sartu->irten->KOTXEA).
```

Kotxe batek beste bat aurreratu ezin duela modelatzeko, norabide berean doazen kotxeen **KONBOI** bat modelatuko dugu.

Norabide bakoitzerako konboi gorria eta urdina izango dugu, konboi bakoitzean gehienez N kotxe egon ahal izango direla

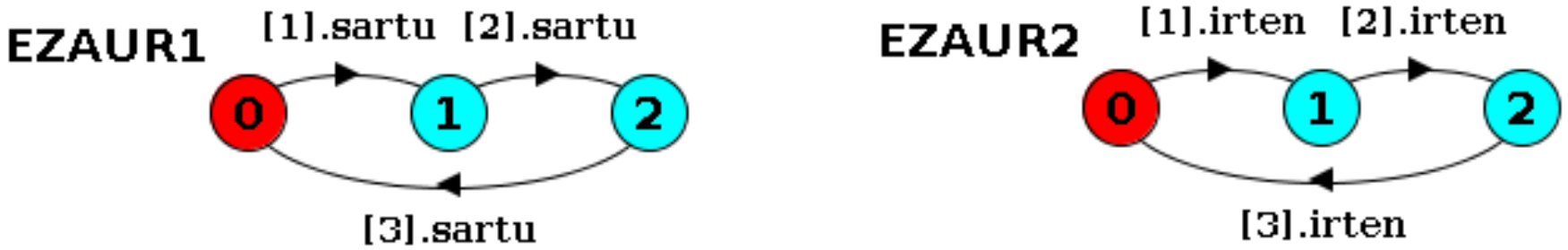
```
|| KOTXEAK =(gorria:KONBOI||urdina:KONBOI ).
```

Bide bakarreko zubia - KONBOI eredua

```
EZAUR1 = C[1],                                  // sartzeko ordena mantentzen du
  C[i:ID]= ([i].sartu -> C[i%N+1]).

EZAUR2 = C[1],                                  // irtetzeko ordena mantentzen du
  C[i:ID]= ([i].irten -> C[i%N+1]).

||KONBOI = ([ID]:KOTXEA||EZAUR1||EZAUR2).
```



Bai: 1.sartu→2.sartu→1.irten→2.irten ☑

Ez: 1.sartu→2.sartu→2.irten→1.irten ☒

Ez du uzten aurreratzen...

Bide bakarreko zubia - ZUBIA eredua

Kotxeak mugi daitezke zubian konkurrenteki soilik norabide berdinean.

Zubiak zenbat kotxe urdin eta gorri dauden bertan kontrolatzen du.

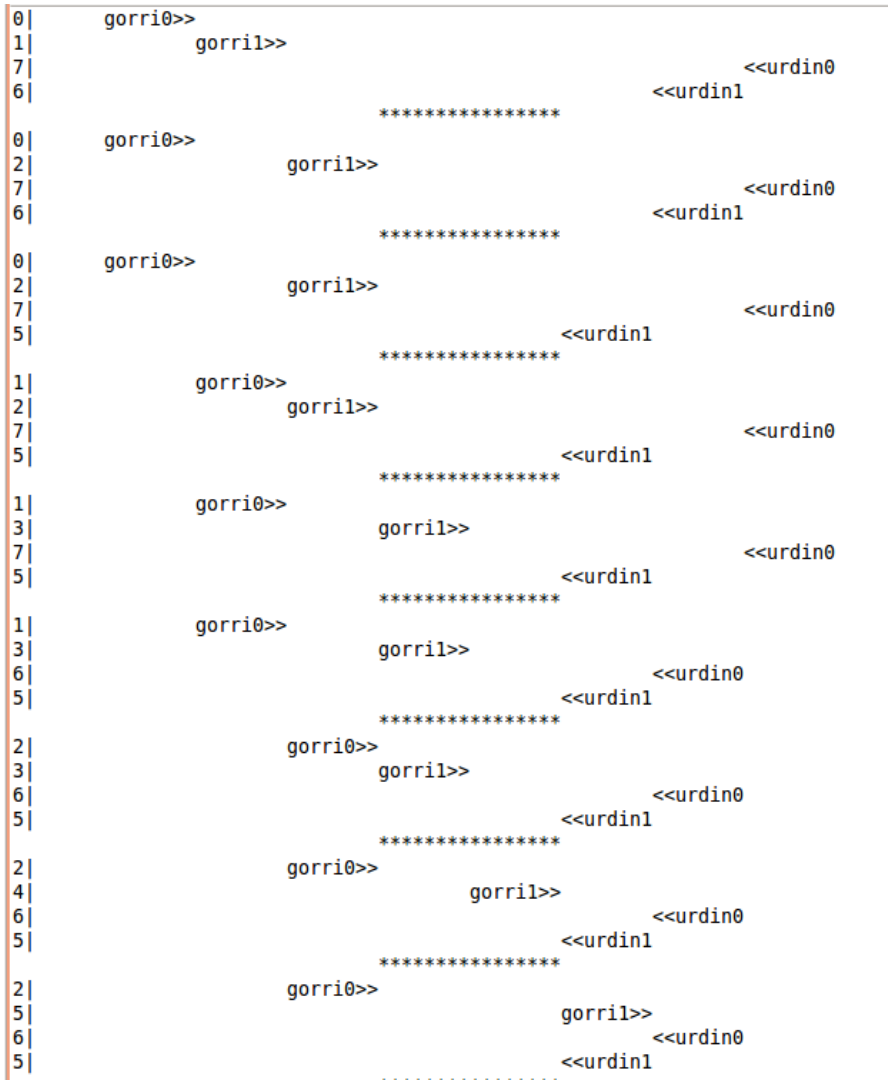
Kotxe gorriak sartu daitezke soilik urdinen kontagailua zero denean,
eta alderantziz.

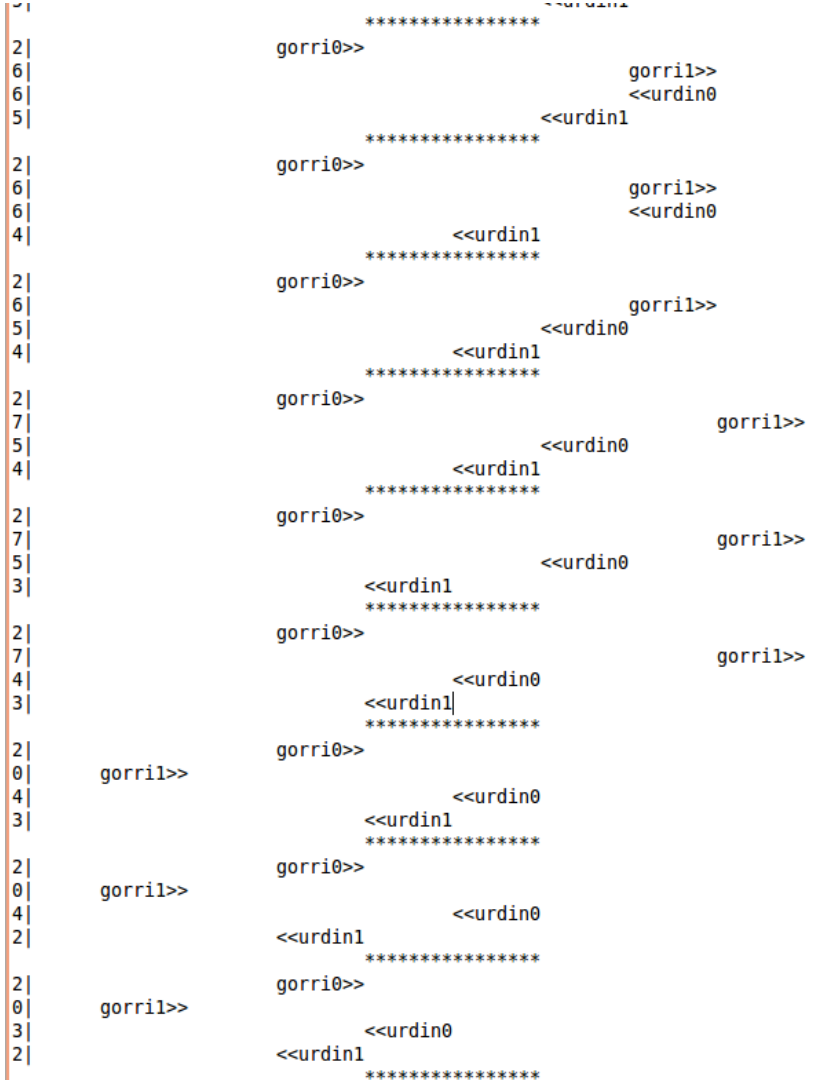
```
ZUBIA = ZUBIA[0][0],                    //hasieran hutsik
ZUBIA[kg:T][ku:T] =                     //kg: gorrien kontagailua   //ku: urdinena
 (
   when (ku==0)  gorria[ID].sartu -> ZUBIA[kg+1][ku]
  |              gorria[ID].irten -> ZUBIA[kg-1][ku]
  |when (kg==0)  urdina[ID].sartu -> ZUBIA[kg]  [ku+1]
  |              urdina[ID].irten -> ZUBIA[kg]  [ku-1]
 ).

||KOTXEAK =(gorria:KONBOI||urdina:KONBOI||ZUBIA).
```

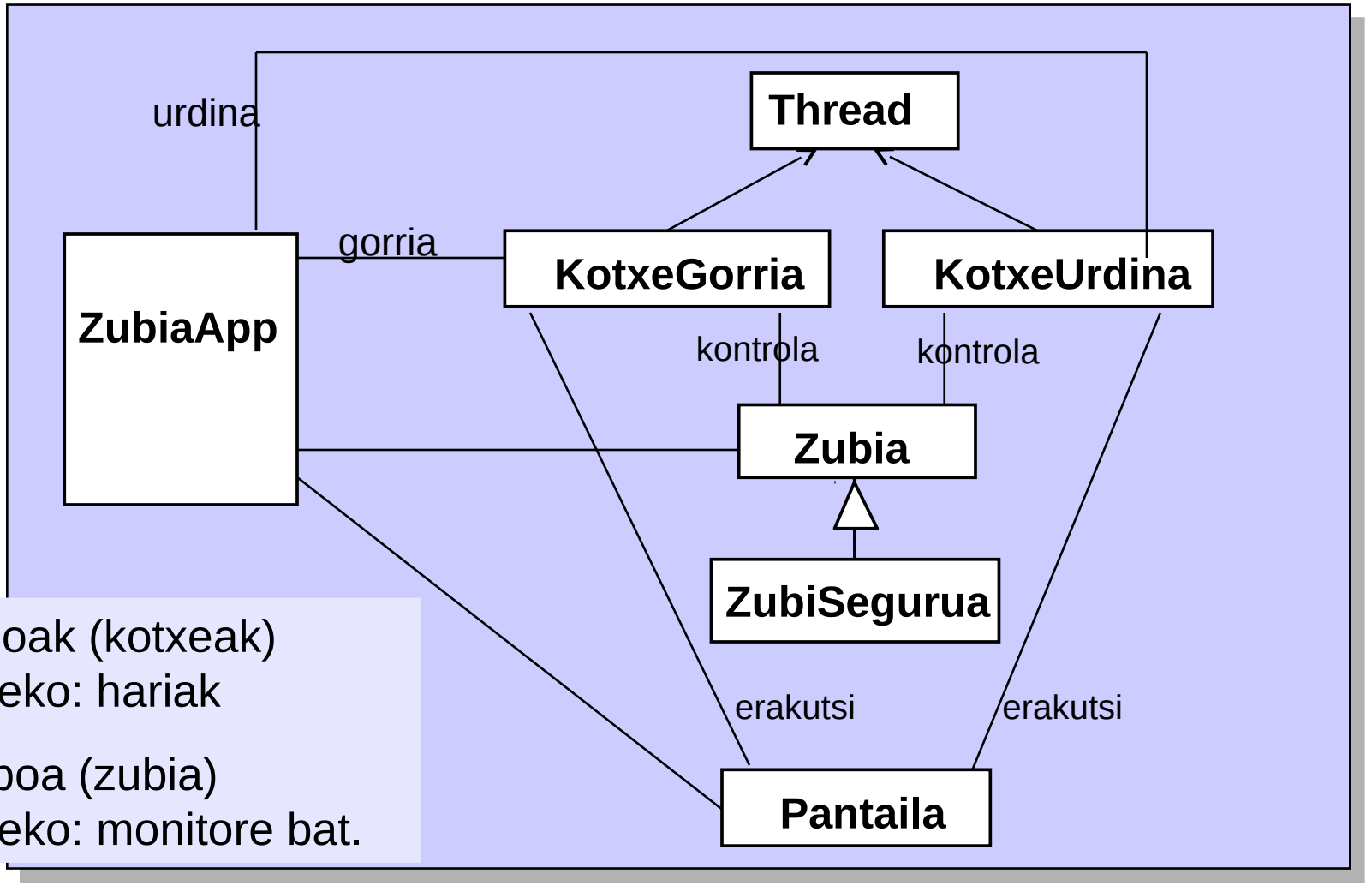
*0 denean, **irten** ekintzak kotxeen kontagailua dekrementatzen uzten du.*
***LTSA**k mapeatzen ditu definitu gabeko egoera hauek **ERROR**era.*

Bide bakarreko zubia





Bide bakarreko zubia – Java-n inplementazioa



Entitate aktiboak (kotxeak) inplementatzeko: hariak

Entitate pasiboa (zubia) inplementatzeko: monitore bat.

Bide bakarreko zubia - ZubiaApp

```
public class ZubiaApp {
  public static void main (String args[]) {
       int max = 2;                  int zabalera = 5;
       int zubezk=zabalera/2;        int zubesk=(zabalera/2)+1;

       KotxeGorria[] gorria= new KotxeGorria[max];
       KotxeUrdina[] urdina= new KotxeUrdina[max];
       Pantaila p= new Pantaila(max, zabalera, zubezk, zubesk);
       Zubia z;
       //z = new Zubia();
       z = new ZubiSegurua();

       for (int i = 0; i<max; i++) {
            gorria[i] = new KotxeGorria(z,p,i);
            urdina[i] = new KotxeUrdina(z,p,i); }
       for (int i = 0; i<max; i++) {
            gorria[i].start();
            urdina[i].start(); }
    }
}
```

Bide bakarreko zubia - Pantaila

ZubiaApp-k sortzen du **Pantaila** klasearen instantzia bat,
eta bere erreferentzia pasatzen zaie
sortzen diren **KotxeGorria** eta **KotxeUrdina** objektu guztiei.

```
class Pantaila {
 // Eraikitzailea: kotxe kopurua, bidearen zabalera eta zubiaren ezker eta eskuina ezartzen ditu
 Pantaila(int m, int zabalera, int ezk, int esk) {...}

 // Mugitu urrats bat i zenbakia duen kotxe gorria
 // Itzultzen du true kotxea zubian edo zubiaren aurretik dagoenean
 boolean mugituGorria(int i)
                                  throws InterruptedException{...}

 // Mugitu urrats bat i zenbakia duen kotxe urdina
 // Itzultzen du true kotxea zubian edo zubiaren aurretik dagoenean
 boolean mugituUrdina(int i)
                                  throws InterruptedException{...}

 // Kotxeak pantailan erakusten ditu
 public void pantailaratu (){...}
```

Bide bakarreko zubia - Pantaila

```
class Pantaila {
        int zab, max, zubezk, zubesk;
        int[] gorriaX,urdinaX;
        String[] tabul;

        Pantaila(int m, int zabalera, int ezk, int esk) {
                max = m;          zubezk=ezk;
                zab = zabalera;   zubesk=esk;
                gorriaX = new int[max];
                urdinaX = new int[max];

                //Kotxe bakoitzaren hasierako posizioa
                for (int i = 0; i<max ; i++) {
                        gorriaX[i] = i;
                        urdinaX[i] = zab-i;
                }
                //Tabulazioak
                tabul = new String[zab+1];
                for (int i=0; i<zab+1; ++i){
                    tabul[i]="";
                    for (int j=0; j<i; ++j)  tabul[i]=tabul[i]+"\t";
                }
                pantailaratu();
        }
//...
```

Bide bakarreko zubia - Pantaila

```
//..
boolean mugituGorria(int i) throws InterruptedException {
  int X = gorriaX[i];
  synchronized (this) {
         if      (X==zab && gorriaX[(i+1)%max]!=0)   X=0;   //Bukaerara iristean
         else if (X!=zab && gorriaX[(i+1)%max]!=X+1) X=X+1; //Beste posizioetan
         if (gorriaX[i]!=X) {        //Mugitu ahal bada, mugitu eta pantailaratu
                  gorriaX[i]=X; pantailaratu();}
  }
  try{Thread.sleep(2000);} catch(InterruptedException e) {}
  return (X>=zubezk-1 && X<=zubesk);          //Zubira sartzeko edo zubian badago
}

boolean mugituUrdina(int i) throws InterruptedException{ //...
}

void pantailaratu() {
  // Gorriak
  for (int i = 0; i<max ; i++)
      System.out.println(gorriaX[i]+"|\t"+tabul[gorriaX[i]]+"gorri"+i+">>");
  // Urdinak
  for (int i = 0; i<max ; i++)
      System.out.println(urdinaX[i]+"|\t"+tabul[urdinaX[i]]+"<<urdin"+i);
  // Zubia
  System.out.print("\t"+tabul[zubezk]);
  for (int i = zubezk; i<zubesk+1 ; i++) System.out.print("********");
  System.out.println();
}
```

Bide bakarreko zubia - KotxeGorria

```
class KotxeGorria extends Thread{
 Zubia zubia; Pantaila pantaila; int zenb;

 KotxeGorria(Zubia b, Pantaila p, int zenb) {
   this.zenb = zenb; zubia = b; pantaila = p; }

 public void run() {
  try {
    while(true) {
     while (!pantaila.mugituGorria(zenb)); // mugitu zubitik kanpo
     zubia.sartuGorria();                  // eskatzen du zubiaren atzipena
     while (pantaila.mugituGorria(zenb));  // mugitu zubian
     zubia.irtenGorria();                  // askatzen du zubiaren atzipena
    }
  } catch (InterruptedException e){}
 }
}
```

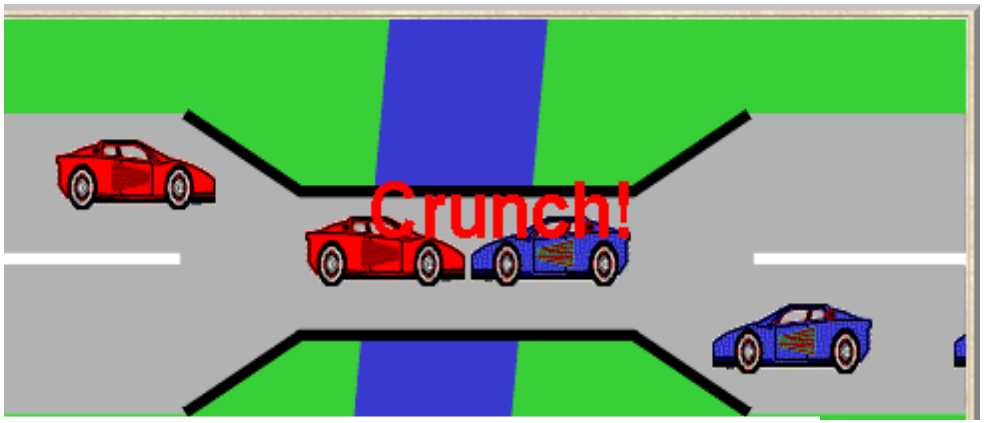
Antzekoa **KotxeUrdina**-rentzat

Bide bakarreko zubia – Zubia

```
class Zubia {
    synchronized void sartuGorria()throws InterruptedException {}
    synchronized void irtenGorria(){}
    synchronized void sartuUrdina()throws InterruptedException {}
    synchronized void irtenUrdina(){}
}
```

Zubia klaseak atzipen metodoen inplementazio hutsa ematen du, hau da, ez du murrizpenik jartzen zubiaren atzipenean.

Zer gertatzen da…… ?



Segurtasuna ziurtatzeko, ZubiSegurua instantziatu behar da

```
class ZubiaApp{...
            Zubia z;
            //z = new Zubia();
            z = new ZubiSegurua();
```

Bide bakarreko zubia – ZubiSegurua...

```
class ZubiSegurua extends Zubia {
 private int kGorria = 0;//kotxe gorrien kopurua zubian
 private int kUrdina = 0;//kotxe urdinen kopurua zubian;
 // Monitoreraren Inbariantea: kGorria>=0 and
 //                            kUrdina>=0 and
 //                            not (kGorria>0 and kUrdina>0)

 synchronized void sartuGorria()
           throws InterruptedException {
     while (kUrdina>0) wait();
     ++kGorria;
 }
 synchronized void irtenGorria(){
     --kGorria;
     if (kGorria==0) notifyAll();
    }
```

ZUBIA ereduaren itzulpen zuzena da.

Bide bakarreko zubia – …ZubiSegurua

```
    synchronized void sartuUrdina()
                throws InterruptedException {

        while (kGorria>0) wait();
              ++kUrdina;
        }
        synchronized void irtenUrdina(){
            --kUrdina;
            if (kUrdina==0) notifyAll();
        }
    }
```

Behar ez diren **wait**-en esnatzeak saihesteko *baldintzapeko notifikazioa* erabiltzen dugu zai dauden hariak esnatzeko soilik zubian dauden kotxeen kopurua zero denean, hau da, azkeneko kotxea zubitik irten denean.

*Ba al dute kotxe guztiek zubia **inoiz** zeharkatzeko aukera?*
*Hau **bizitasun**-propietate bat da.*

7.3 Bizitasuna

Segurtasun-propietate batek adierazten du
ez dela ezer **txarrik** gertatuko

Bizitasun propietate batek adierazten du
zerbait **ona *noizbait*** gertatuko dela

Bide bakarreko zubia:

*Ba al dute kotxe guztiek zubia **inoiz** zeharkatzeko aukera?*
Hau da, **AURRERAPENIK** egiteko aukera?

Aurrerapen (progress) propietate batek adierazten du
beti emango dela kasua non ekintza bat noizbait izango den egikaritua.

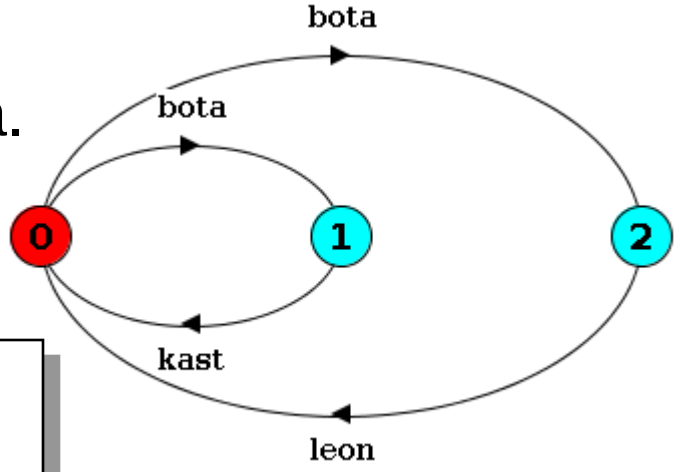
Aurrerapenaren aurkakoa:
gosez hil (starvation), ekintza bat inoiz egikaritzen ez den egoera

Aurrerapen-propietateak – bidezko aukera

Bidezko aukera: Trantsizio multzo baten gaineko aukera bat infinitu aldiz egikaritzen bada, multzoko trantsizio bakoitza infinitu aldiz egikarituko da.

Txanpon bat infinitu aldiz botatzen bada, espero dezakegu leon aukeratuko dela infinitu aldiz eta kastillo ere infinitu aldiz aukeratuko dela.

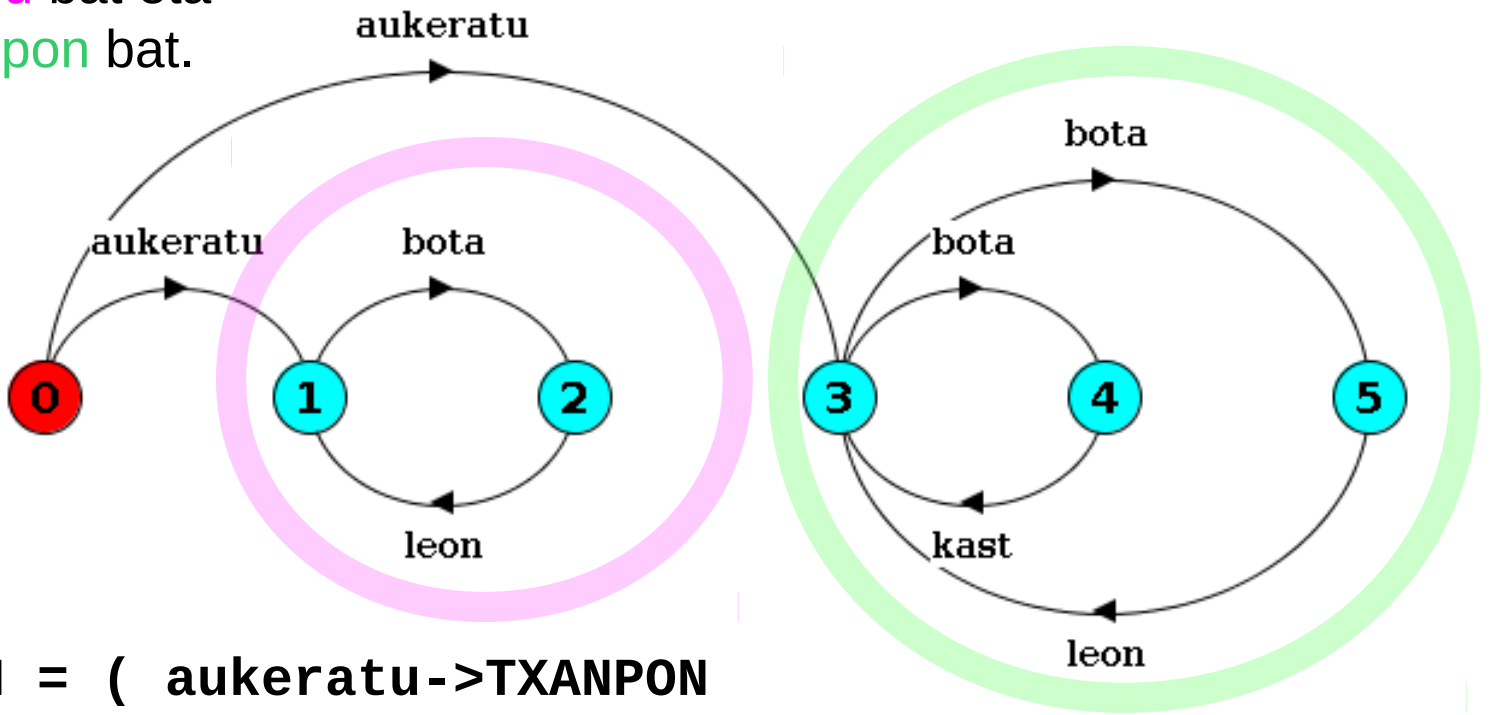
Honek eskatzen du **bidezko aukera**!




```
TXANPON = (bota->leon->TXANPON
          |bota->kast->TXANPON).
```

Aurrerapen-propietateak

Demagun aukeratuak izan daitezkeen bi txanpon posible daudela:
txanpon faltsu bat eta
egiazko txanpon bat.

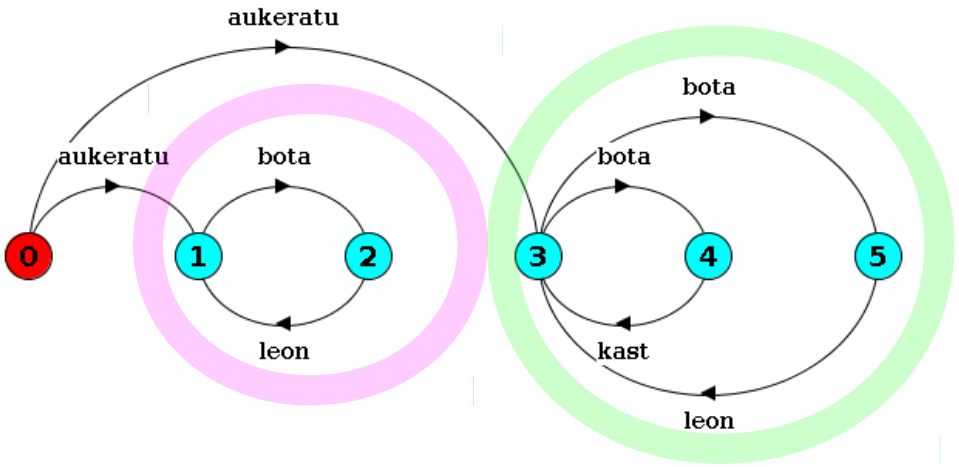



```
BITXANPON = ( aukeratu->TXANPON
            |aukeratu->FALTSUA ),
FALTSUA   = ( bota->leon->FALTSUA ),
TXANPON   = ( bota->leon->TXANPON
            |bota->kast->TXANPON ).
```

Aurrerapenaren analisia

Berezko analisia BITXANPON sistemarako:
ekintza bakoitzarentzat aurrerapen propietate bat.

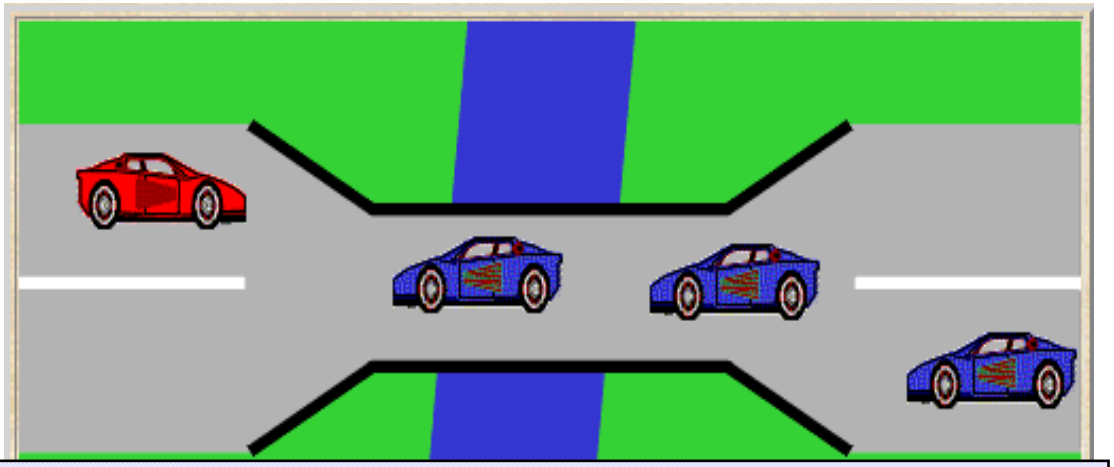
```
Progress violation for actions:
{aukeratu, kast}
Path to terminal set of states:
     aukeratu
Actions in terminal set:
{bota, leon}
```



Aurrerapen-propietatea - Bide bakarreko zubia

Bide bakarreko zubiaren inplementazioan aurrerapen bortxaketa eman daiteke. Hala ere, **berezko aurrerapen analisia** ereduari aplikatzen badiogu ez du bortxaketarik detektatzen!!

Zergatik ez?



No progress violations detected.

Bidezko aukerak adieratzen du
edozein exekuzio posible noizbait gertatuko dela,
kotxeak gosez hiltzen ez diren egoerak barne.
Aurrerapen-arazoak detektatzeko,
ekintzetarako planifikazio-politika (scheduling policy) bat gainean jarri beharko da, zubia beteta dagoen egoera modelatzeko.

```
0|      gorri0>>
1|              gorri1>>
2|                      gorri2>>
3|                              gorri3>>
4|                                      gorri4>>
7|                                                              <<urdin0
6|                                                      <<urdin1
                                ***************
0|      gorri0>>
1|              gorri1>>
2|                      gorri2>>
4|                                      gorri3>>
5|                                              gorri4>>
6|                                                      <<urdin0
5|                                              <<urdin1
                                ***************
0|      gorri0>>
1|              gorri1>>
2|                      gorri2>>
4|                                      gorri3>>
6|                                                      gorri4>>
6|                                                      <<urdin0
5|                                              <<urdin1
                                ***************
0|      gorri0>>
2|                      gorri1>>
3|                              gorri2>>
5|                                              gorri3>>
7|                                                              gorri4>>
6|                                                      <<urdin0
5|                                              <<urdin1
                                ***************
1|              gorri0>>
3|                              gorri1>>
5|                                              gorri2>>
6|                                                      gorri3>>
0|      gorri4>>
6|                                                      <<urdin0
5|                                              <<urdin1
                                ***************
2|                      gorri0>>
4|                                      gorri1>>
5|                                              gorri2>>
7|                                                              gorri3>>
1|              gorri4>>
6|                                                      <<urdin0
5|                                              <<urdin1
                                ***************
3|                              gorri0>>
5|                                              gorri1>>
6|                                                      gorri2>>
7|                                                              gorri3>>
2|                      gorri4>>
6|                                                      <<urdin0
5|                                              <<urdin1
                                ***************
```

```
                                ***************
3|                              gorri0>>
5|                                              gorri1>>
6|                                                      gorri2>>
0|      gorri3>>
2|                      gorri4>>
6|                                                      <<urdin0
5|                                              <<urdin1
                                ***************
4|                                      gorri0>>
6|                                                      gorri1>>
7|                                                              gorri2>>
0|      gorri3>>
3|                              gorri4>>
6|                                                      <<urdin0
5|                                              <<urdin1
                                ***************
5|                                              gorri0>>
6|                                                      gorri1>>
7|                                                              gorri2>>
1|              gorri3>>
3|                              gorri4>>
6|                                                      <<urdin0
5|                                              <<urdin1
                                ***************
5|                                              gorri0>>
7|                                                              gorri1>>
0|      gorri2>>
2|                      gorri3>>
4|                                      gorri4>>
6|                                                      <<urdin0
5|                                              <<urdin1
                                ***************
6|                                                      gorri0>>
7|                                                              gorri1>>
0|      gorri2>>
2|                      gorri3>>
4|                                      gorri4>>
6|                                                      <<urdin0
5|                                              <<urdin1
                                ***************
6|                                                      gorri0>>
7|                                                              gorri1>>
1|              gorri2>>
2|                      gorri3>>
4|                                      gorri4>>
6|                                                      <<urdin0
5|                                              <<urdin1
                                ***************
6|                                                      gorri0>>
0|      gorri1>>
1|              gorri2>>
2|                      gorri3>>
4|                                      gorri4>>
6|                                                      <<urdin0
```

Aurrerapena – ekintzen lehentasuna

Ekintzen lehentasun-adierazpenek antolaketa-propietateak deskribatzen dituzte:

Lehentasun handia(“**<<**”)

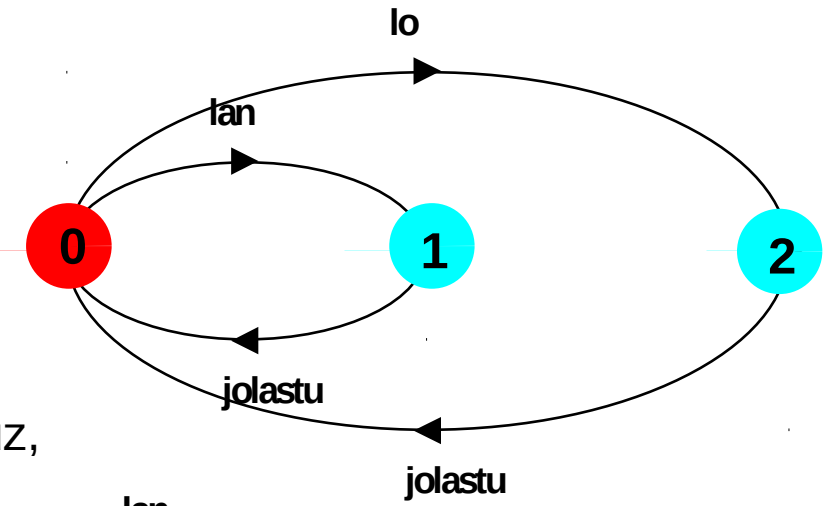
||C = (P||Q)<<{a1,…,an} konposaketak adierazten du
a1,..,an ekintzek **P||Q**-ko alfabetoko beste edozein ekintza,
tau ekintza isila barne, baino lehentasun handiagoa dutela.
Sistema honetako edozein aukeran,
*trantsizio bat edo gehiago etiketatzen bada **a1,..,an** ekintzekin,*
lehentasun baxuago duten trantsizioak baztertzen dira.

Lehentasun txikia(“**>>**”)

||C = (P||Q)>>{a1,…,an} konposaketak adierazten du
a1,..,an ekintzek **P||Q**-ko alfabetoko beste edozein ekintza,
tau ekintza isila barne, baino lehentasun txikiagoa dutela.
Sistema honetako edozein aukeran,
*trantsizio bat edo gehiago ez bada etiketatzen **a1,..,an** ekintzekin,*
***a1,..,an** ekintzekin etiketatuko trantsizioak baztertzen dira.*

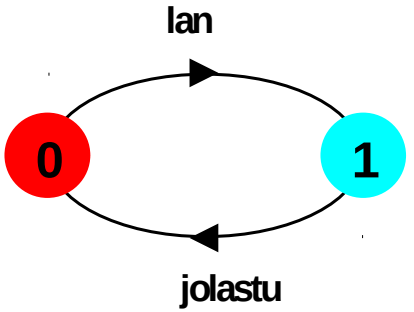
Aurrerapena – ekintzen lehentasuna

```
NORMAL =(lan->jolastu->NORMAL
        |lo->jolastu->NORMAL).
```

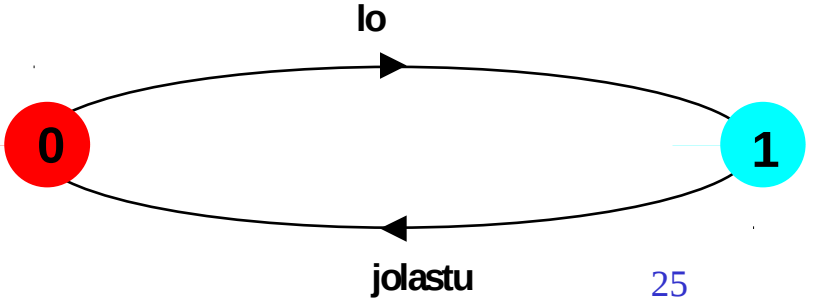


Ekintzen lehentasunak
LTSren emaitzak sinplifikatzen ditu,
aukeretako lehentasun baxuko ekintzak baztertuz,
hau da, ezabatuz.

```
||HANDI =(NORMAL)<<{lan}.
```




```
||TXIKI =(NORMAL)>>{lan}.
```



7.4 Bide bakarreko zubia beteta

Nola adierazi zubia beteta ekintzen lehentasuna erabiliz?

Eman diezaiekegu kotxe gorriei
lehentasun gehiago urdinei baino (edo alderantziz)?
Ez. Pribilegiorik ez!!

Soilik eragin dezakegu zubia betetzea
kotxe guztien zubitik **irten** ekintzei **lehentasuna jeisten** badiegu.

```
||ZubiaBeteta = (BidebakarrekoZubia)
                  >>{gorria[ID].irten,urdina[ID].irten}.
```

Aurrerapenaren analisia ? LTS?

Bide bakarreko zubia beteta

```
Aurrerapena violation: URDINAPASA
Trace to terminal set of states:
     gorria.1.sartu
     gorria.2.sartu
Actions in terminal set:
{gorria.1.sartu, gorria.1.irten,
gorria.2.sartu, gorria.2.irten,
gorria.3.sartu, gorria.3.irten}

Aurrerapena violation: GORRIAPASA
Trace to terminal set of states:
     urdina.1.sartu
     urdina.2.sartu
Actions in terminal set:
{urdina.1.sartu, urdina.1.irten,
urdina.2.sartu, urdina.2.irten,
urdina.3.sartu, urdina.3.irten}
```

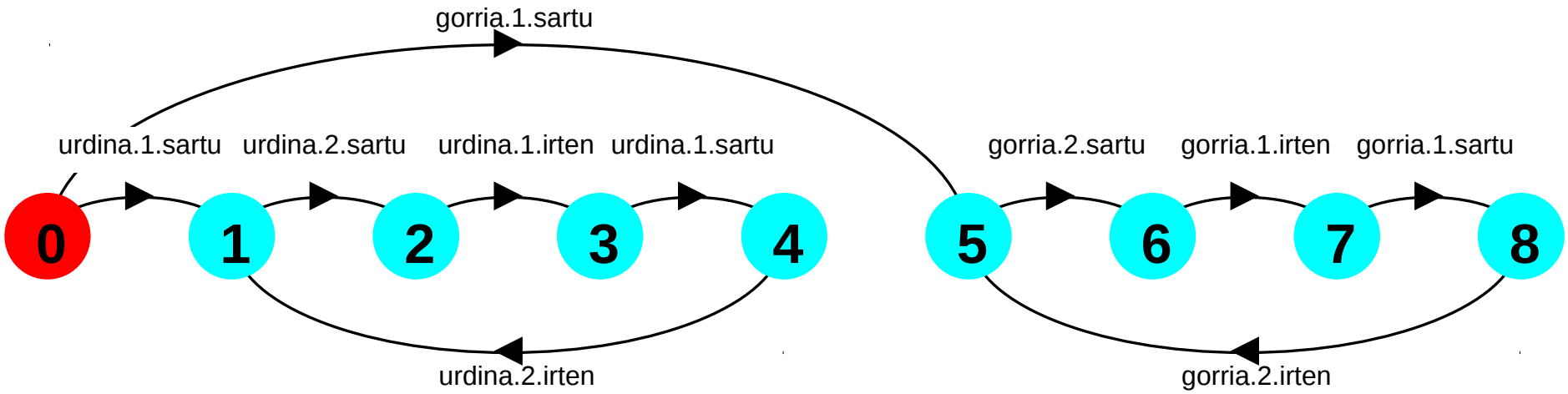
Kolore bakoitzeko kotxe bat baino gehiago daudenean, posible da

- lehenengoz kotxe gorri bat zubian sartzea,
- eta, ondoren, kotxe gorriak sartzen jarraitzea,
- urdinei inoiz pasatzen ez utziz

(edo alderantziz).

Bide bakarreko zubia beteta

```
||ZUBIABETETA = (BIDEBAKARREKOZUBIA)
                >>{gorria[ID].irten,urdina[ID].irten}.
```



Zer gertatuko da, zubia beteta modelatzeko,
kotxeen sartu ekintzari lehentasun handia ematen badiegu?

Gerta daiteke zubia beteta,
norabide bakoitzean kotxe bakarra mugitzen baldin badago?

Aurrerapena - Bide bakarreko zubiaren eredua zuzentzen

Zubiak jakin behar du kotxeak pasatzeko **zain** dauden ala ez.

KOTXEA aldatu :

```
KOTXEA = (eskatu->sartu->irten->KOTXEA).
```

ZUBIA aldatu:

Kotxe gorriak zubian sartzeko baimena dute
ez badago zubian kotxe urdinik
eta ez badago kotxe urdinik zain zubian sartzeko.

Kotxe urdinak zubian sartzeko baimena dute
ez badago zubian kotxe gorririk
eta ez badago kotxe gorririk zain zubian sartzeko.

Aurrerapena - Bide bakarreko zubiaren eredua zuzentzen

```
//kg zubian dauden gorrien kontagailua
//ku zubian dauden urdinen kontagailuak
//zg zubian sartzeko zain dauden gorrien kontagailua
//zu zubian sartzeko zain dauden urdinen kontagailuak

ZUBIA = ZUBIA[0][0][0][0],
ZUBIA[kg:T][ku:T][zg:T][zu:T] =
  (                        gorria[ID].eskatu -> ZUBIA[kg]  [ku]  [zg+1][zu]
  |when (ku==0 && zu==0)   gorria[ID].sartu  -> ZUBIA[kg+1][ku]  [zg-1][zu]
  |                        gorria[ID].irten  -> ZUBIA[kg-1][ku]  [zg]  [zu]
  |                        urdina[ID].eskatu -> ZUBIA[kg]  [ku]  [zg]  [zu+1]
  |when (kg==0 && zg==0)   urdina[ID].sartu  -> ZUBIA[kg]  [ku+1][zg]  [zu-1]
  |                        urdina[ID].irten  -> ZUBIA[kg]  [ku-1][zg]  [zu]
  ).
```

Orain ondo?

Bide bakarreko zubia zuzendua-ren analisia

```
Trace to DEADLOCK:
   gorria.1.eskatu
   gorria.2.eskatu
   gorria.3.eskatu
   urdina.1.eskatu
   urdina.2.eskatu
   urdina.3.eskatu
```

Kotxeak zai daude
zubiaren bi aldeetan,
eta beraz,
zubiak ez du uzten sartzen
ez gorriak
ez urdinak.

Konponbidea?

Sartu **asimetria**-ren bat probleman (filosofoen afarian bezala).

Izan daiteke aldagai boolear bat (ut) elkar-blokeaketa puskatzen duena jakinarazten noiz den zubian sartzeko kotxe urdinen txanda eta noiz gorriena.

Arbitrarioki hasieratu ut true, hasieran kotxe urdinei aurrekotasuna emanik.

Aurrerapena - Bide bakarreko zubiaren eredua ber-zuzentzen

```
const True = 1
const False = 0
range B = False..True
//  ut : true adierazten du urdinen txanda,   false adierazten du gorrien txanda
ZUBIA = ZUBIA[0][0][0][0][True],
ZUBIA[kg:T][ku:T][zg:T][zu:T][ut:B] =
(                              gorria[ID].eskatu-> ZUBIA[kg]  [ku]  [zg+1][zu]  [ut]
 |when (ku==0&&(zu==0||!ut)) gorria[ID].sartu -> ZUBIA[kg+1][ku]  [zg-1][zu]  [ut]
 |                           gorria[ID].irten -> ZUBIA[kg-1][ku]  [zg]  [zu]  [True]
 |                           urdina[ID].eskatu-> ZUBIA[kg]  [ku]  [zg]  [zu+1][ut]
 |when (kg==0&&(zg==0||ut)) urdina[ID].sartu -> ZUBIA[kg]  [ku+1][zg]  [zu-1][ut]
 |                           urdina[ID].irten -> ZUBIA[kg]  [ku-1][zg]  [zu]  [False]
).
```

➡ ***Analisia ?***

Bide bakarreko zubia zuzenduaren implementazioa - BidezkoZubia

```
class BidezkoZubia extends Zubia {
  private int kGorria = 0;   //kotze gorrien kopurua zubian;
  private int kUrdina = 0;   //kotze urdinen kopurua zubian;
  private int zaiGorria = 0;//sartzeko zai dauden gorrien kopurua
  private int zaiUrdina = 0;//sartzeko zai dauden urdinen kopurua
  private boolean urdinTx = true;
  synchronized void sartuGorria()throws InterruptedException {
     ++zaiGorria;
     while (kUrdina>0||(zaiUrdina>0 && urdinTx)) wait();
     --zaiGorria;
     ++kGorria;
  }
  synchronized void irtenGorria(){
     --kGorria;
     urdinTx = true;
     if (kGorria==0) notifyAll();
  }
  //...
```

FSP ereduaren itzulpena.

Bide bakarreko zubia zuzenduaren implementazioa - BidezkoZubia

```
  //...
  synchronized void sartuUrdina()
                    throws InterruptedException {
     ++zaiUrdina;
     while (kGorria>0||(zaiGorria>0 && !urdinTx)) wait();
     --zaiUrdina;
     ++kUrdina;
  }
  synchronized void irtenUrdina(){
     --kUrdina;
     urdinTx = false;
     if (kUrdina==0) notifyAll();
  }
}
```

```
class ZubiaApp{
 ...
 b = new BidezkoZubia();
```

Ez da beharrezkoa monitorean **eskatu** metodo berri bat gehitzea. Aldatzen dugu **sartu** metodoa, dei egileak zubia atzitu dezakeen ala ez konprobatu aurretik, **`zain`** kontagailu bat inkrementatzeko.

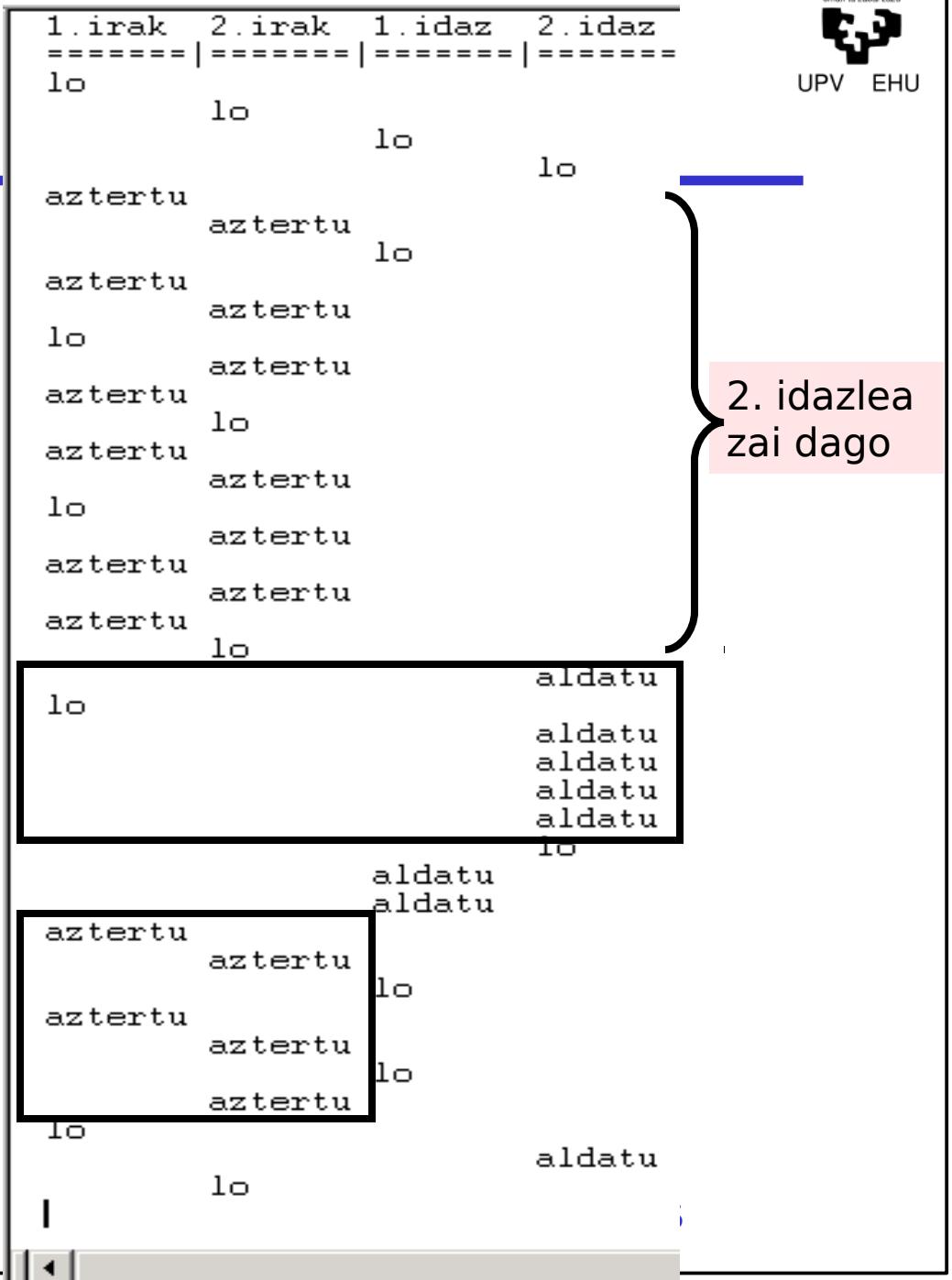
7.5 Irakurleak eta idazleak

Datu-base konpartitu bat
bi prozesu motak atzitzen dute:

- **Irakurleek** datubasea aztertzen duten eragiketak burutzen dituzte eta
- **Idazleek** aztertu eta eguneratu egiten dute.

Idazle batek datu-basearen atzipen exklusiboa behar du izan;

Hainbat **Irakurlek** datubasea konkurrenteki atzitu dezakete.



Irakurleak/Idazleak eredua

- Gertaera edo ekintza interesgarriak?

  eskuratuIrakurri, askatuIrakurri, eskuratuIdatzi, askatuIdatzi

- Prozesuak identifikatu:

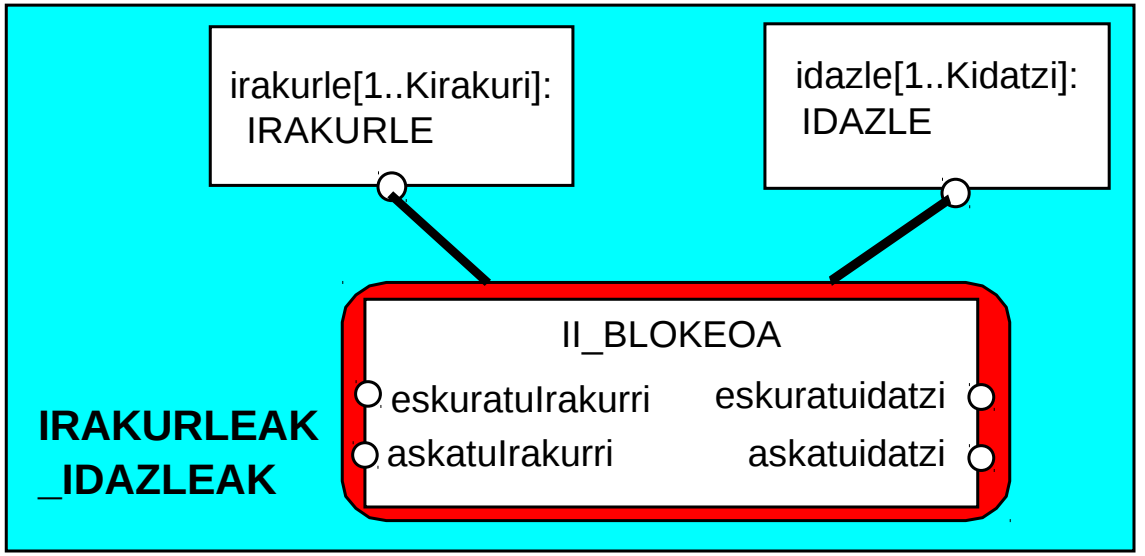
  Irakurleak, Idazleak eta II_Blokeoa

- Propietateak identifikatu:

  II_Segurua,
  II_Aurrerapena

- Prozesuak eta elkarekintzak definitu:

  egitura →



Irakurleak/Idazleak eredua

```
IRAKURLE = (eskIrak-> aztertu-> askIrak->IRAKURLE).
IDAZLE   = (eskIdaz-> aldatu -> askIdaz->IDAZLE).

const False = 0
const True  = 1
range Bool  = False..True
const Kirak = 2            // Irakurle kopuru maximoa
const Kidaz= 2             // Idazle kopuru maximoa

II_BLOKEOA = II[0][False],
II[irak:0..Kirak][idazten:Bool] =
  (when (!idazten)              irak[1..Kirak].eskIrak -> II[irak+1][idazten]
  |                             irak[1..Kirak].askIrak -> II[irak-1][idazten]
  |when (irak==0 && !idazten)   idaz[1..Kidaz].eskIdaz -> II[irak]  [True]
  |                             idaz[1..Kidaz].askIdaz -> II[irak]  [False]
  ).

||IRAKURLEAK_IDAZLEAK = (   irak[1..Kirak]:IRAKURLE
                         || idaz[1..Kidaz]:IDAZLE  || II_BLOKEOA).
```

Blokeoak mantentzen ditu

- irakurleen kopurua duen kontagailu bat

  eta

- idazleentzat aldagai boolear bat.

Segurtasunaren eta aurrerapenaren analisiak?

Irakurleak/Idazleak - aurrerapena

Aztertzeko nola ibiltzen den sistema une larrietan (gainkargatua, stressatua,…) kontrako baldintzak ezarriko ditugu ekintzen lehentasuna erabiliz:

Jeisten dugu `askatu` ekintzen lehentasuna irakurleentzat eta idazleentzat.

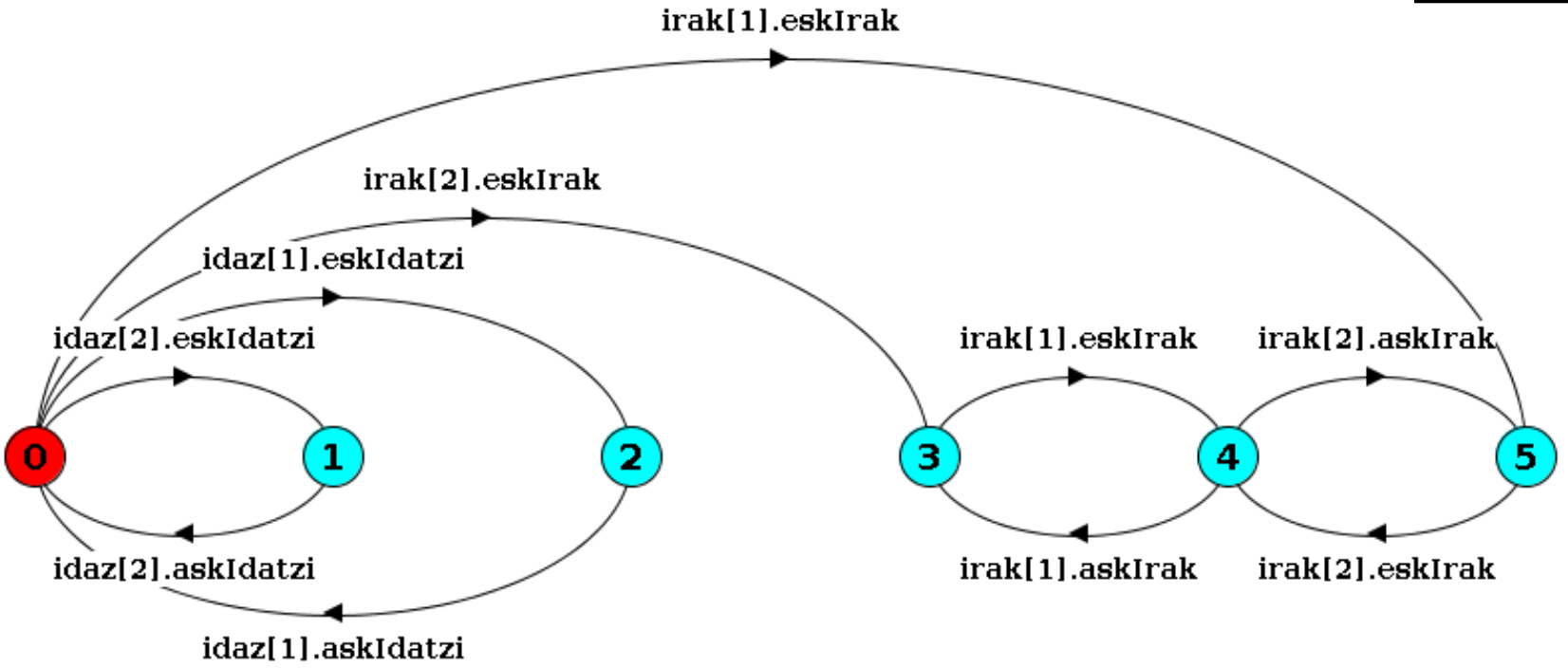
```
||II_AURRERAPENA = IRAKURLEAK_IDAZLEAK
                        >>{irak[1..Kirak].askIrak,
                           idaz[1..Kirak].askIdaz}.
```

Aurrerapenaren analisia? LTS?

Irakurleak/Idazleak eredua - aurrerapena

```
Aurrerapena violation: IDATZI
Path to terminal set of states:
      irakurle.1.eskuratuIrakurri
Actions in terminal set:
{irakurle.1.eskuratuIrakurri, irakurle.1.askatuIrakurri,
 irakurle.2.eskuratuIrakurri, irakurle.2.askatuIrakurri}
```

Idazlea gosez hilik: irakurleen kopurua ez da inoiz zerora erortzen.



Irakurleak/Idazleak - Idazlearen lehentasuna

Estrategia: *Irakurleak blokeatzea, idazle bat zai badago.*

```
IDAZLE   = (eskatu->eskIdaz-> aldatu-> askIdaz->IDAZLE).
```

```
II_BLOKEOA = II[0][False][0],
II[irak:0..Kirak][idazten:Bool][zai:0..Kidaz] =
  (when (!idazten && zai==0)
        irak[1..Kirak].eskIrak  -> II[irak+1][idazten][zai]
  |     irak[1..Kirak].askIrak  -> II[irak-1][idazten][zai]
  |when (irak==0 && !idazten)
        idaz[1..Kidaz].eskIdaz  -> II[irak]  [True]   [zai-1]
  |     idaz[1..Kidaz].askIdaz  -> II[irak]  [False]  [zai]
  |     idaz[1..Kidaz].eskatu   -> II[irak]  [idazten][zai+1]
  ).
```

Segurtasunaren eta aurrerapenaren analisia ?

Irakurleak/Idazleak eredua - Idazlearen lehentasuna

No deadlocks/errors

```
Progress violation for actions:
        irak[1..2].{askIrak, aztertu, eskIrak}
Trace to terminal set of states:
        idaz.2.eskatu
Cycle in terminal set:
        idaz.1.eskatu
        idaz.1.eskIdaz
        idaz.1.aldatu
        idaz.1.askIdaz
Actions in terminal set:
        idaz[1..2].{aldatu, askIdaz, eskIdaz, eskatu}
```

Irakurlea gosez hilda:
idazle bat beti zai badago.

Praktikan hau onargarria litzateke,
normalean irakurleen atzipen gehiago daudelako idazleenak baino.
Gainera irakurleek nahi izaten dute eguneratuen dagoen informazioa.

*Nahi izanezkero, biak **IRAKURRI** eta **IDATZI** aurrerapen propietateak bete daitezke,*
***txanda** aldagai bat erabiltzen badugu (bide bakarreko zubian bezala).*

Irakurleak/Idazleak inplementatzen: monitorearen interfazea

Konzentratuko gara monitorearen inplementazioan:

```
interface IrakurriIdatzi {
     public void eskuratuIrakurri()
         throws InterruptedException;
     public void askatuIrakurri();
     public void eskuratuIdatzi()
         throws InterruptedException;
     public void askatuIdatzi();
}
```

Definitzen dugu interface bat,
inplementatuko ditugun monitorearen metodoak erazagutzeko,
eta interfaze honen inplementazio alternatibo ezberdinak garatuko ditugu

Lehenengoa, **IRAKIDATZBLOKEOA *segurua*.**

Irakurleak/Idazleak implementatzen: IrakurriIdatziSegurua

```
class IrakurriIdatziSegurua implements IrakurriIdatzi {
  private int irakurleak = 0;
  private boolean idazten = false;

  public synchronized void eskuratuIrakurri()
              throws InterruptedException {
    while (idazten) wait();
    ++irakurleak;
  }

  public synchronized void askatuIrakurri() {
    --irakurleak;
    if(irakurleak==0) notify();
  }
```

Desblokeatu idazle bakar bat, *irakurle gehiago ez dagoenean.*

Irakurleak/Idazleak implementatzen: IrakurriIdatziSegurua

```
  public synchronized void eskuratuIdatzi()
                  throws InterruptedException {
     while (irakurleak>0 || idazten) wait();
     idazten = true;
   }

   public synchronized void askatuIdatzi() {
     idazten = false;
     notifyAll();
   }
}
```

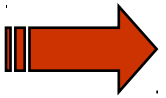
Desblokeatu irakurle **guztiak**

```
1.irak  | 2.irak  | 1.idaz  | 2.idaz
========|=========|=========|=======
lo
        lo
                  lo
                            lo
aztertu
        aztertu
                  lo
                            lo
aztertu
        aztertu
                            lo
aztertu
        aztertu
aztertu
        lo
aztertu
        aztertu
aztertu
        aztertu
lo
        aztertu
aztertu
        lo
aztertu
        aztertu
aztertu
        aztertu
aztertu
        aztertu
aztertu
        lo
aztertu
        aztertu
lo
        aztertu
aztertu
        aztertu
aztertu
        lo
aztertu
        aztertu
```

IrakurriIdatziSegurua

Hala ere, monitorearen inplementazio honek **IDATZI** aurrerapenaren arazoa sufritzen du:

idazlea gosez hil daiteke ez bada inoiz zerora erortzen irakurleen kopurua



konponbidea?

Irakurleak/Idazleak implementazioa: IrakurriIdatziLehentasuna

```
class IrakurriIdatziLehentasuna implements IrakurriIdatzi{
  private int irakurleak =0;  private boolean idazten = false;
  private int zaiW = 0; // zai dauden Idazleen kopurua.
  public synchronized void eskuratuIrakurri() throws InterruptedException{
    while (idazten || zaiW>0) wait();
     ++irakurleak;
}
 synchronized public void askatuIrakurri() {
    --irakurleak;
    if (irakurleak==0) notify();
  }
  synchronized public void eskuratuIdatzi() {
    ++zaiW;
    while (irakurleak>0||idazten) try{wait();}
                                  catch(InterruptedException e){}
    --zaiW;
    idazten = true;
  }
  synchronized public void askatuIdatzi() {
    idazten = false;
    notifyAll();
  }
}
```

Ariketak

Ondoko ariketetarako FSP eredua eta Java inplementazioa egin, segurtasun- eta bizitasun-propietateak justifikatuz:

1. LIFO pilaren arazoa, prozesu guztiak noizbait pilatik aterako direla ziurtatuz.

2. Konkurrenteko ikasleek azterketa garaian ikasten egoten dira (suposatzen da).
   Zerbait ez badute ulertzen irakaslearen bulegora joaten dira, bulegoan sartu, galdera egin eta erantzuna jaso ondoren bulegotik ateratzen dira, berriz ikastera joateko.
   Konkurrenteko irakasleak, ikasle batek galdera bat egiten dionean, erantzun baino lehen pentsatu egiten du (suposatzen da).
   Gainera irakaslea nahiko berezia denez, tutoretzetarako ondoko arauak ditu:
   - Bulegoan bi ikasle batera egon daitezke, baina ez gehiago.
   - Ikasle batek egin duen galdera erantzun arte, besteak ezin du galdetu.
   - Ikasle bakoitzak galdera bakar bat egin dezake tutoretza bakoitzean.
   - Bulegoan sartzeko eta galdera egiteko ez da errespetatzen aurretik zein zegoen.

3. Zebrabide batetara iristean, kotxeak zain geldituko dira
   oinezkoren bat pasatzen edo pasatzeko zain baldin badago.
   Oinezkoek, ordea, zain geldituko dira
   une horretan kotxe bat pasatzen ari bada, harrapatuak ez izateko.
   Gainera ataskorik sor ez dadin,
   hiru kotxe baino gehiago zain badaude, oinezkoek ere itxarongo dute.

Ariketak

(Ariketa Extra)

4. Zubiaren soluzioak egokitu, kotxe grafikoekin inplementatzeko.

   Exekuzioan zubiaren mota aldatzeko aukera emango da:

   - zubi ez segurua,
   - zubi segurua
   - eta bidezko zubia.