

6. gaia

Elkar-blokeaketa

Elkar-blokeaketa

Sistemaren **elkar-blokeaketa**:
Aurrerapenik ezin da egin.

Asmoa:
Elkar-blokeaketa saihestu,
elkar-blokeaketa gertatuko ez den
sistemak diseinatzeko

Elkar-blokeaketa: Lau baldintza beharrezko eta nahiko

- ◆ Seriean berrerabilitako baliabideak:

Prozesuek erabiltzen dituzte baliabide konpartituak, elkar-bazterketa ziurtatuz.

- ◆ Eskuratze inkrementala:

Prozesuek heltzen dituzte eskuratutako baliabideak, baliabide gehiago eskuratzeko zai dauden bitartean.

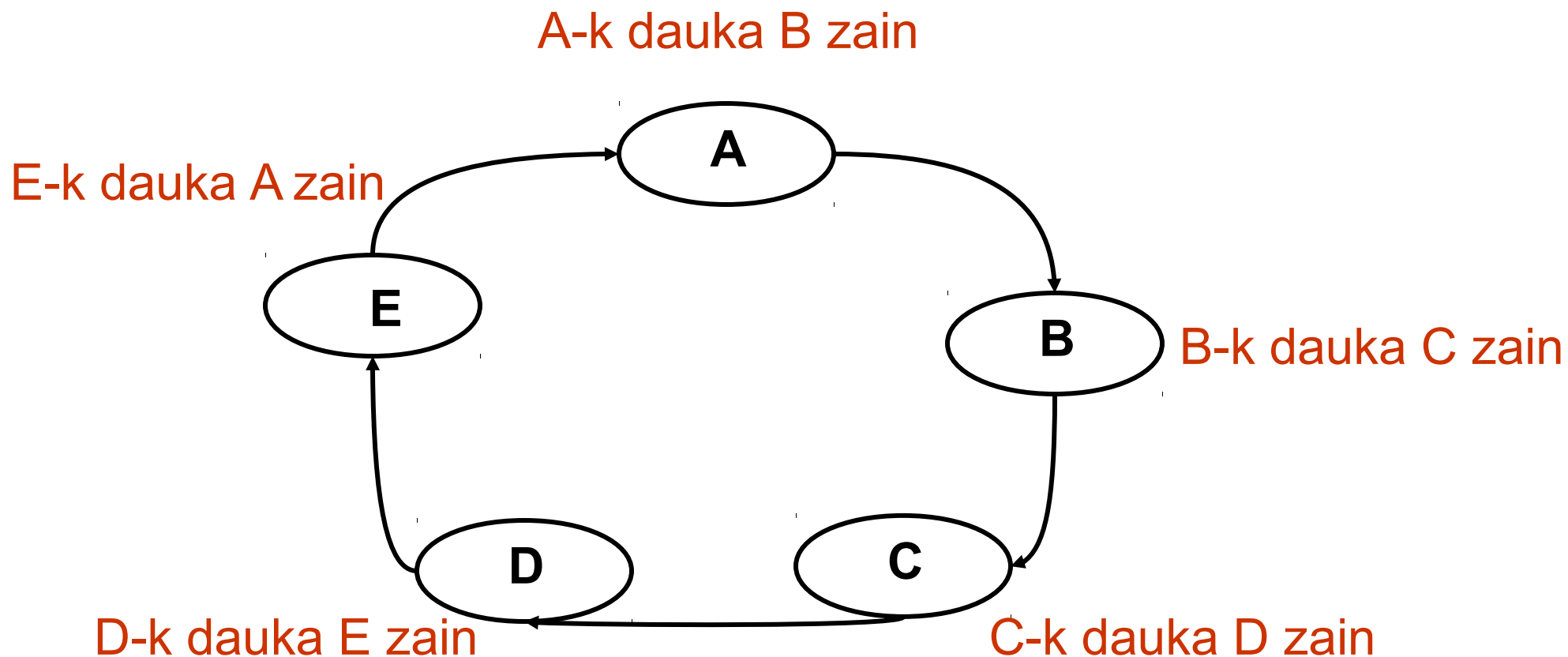
- ◆ Ezin indarrez kendu:

Prozesu batek baliabide bat eskuratu duenean soilik borondatez askatuko du, ezin izango zaio indarrez kendu.

- ◆ Zain-zikloa:

Prozesuen zikloa emango da prozesu bakoitzak baliabide bat heltzen duenean, zikloko aurreko prozesua baliabide hori eskuratzeko zain dagoela.

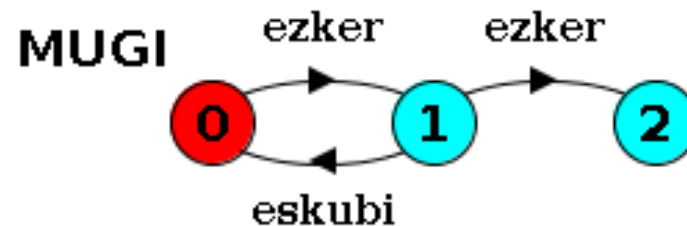
Zain-zikloa



6.1 Elkar-blokeaketaren analisia

- ◆ Egoera blokeatua:
irteerazko transitziorik ez duen egoera da.
- ◆ FSP-n: **STOP** prozesua

MUGI = (ezker->(eskubi->MUGI|ezker->**STOP**)) .



- ◆ Analisia **LTSA** erabiliz:

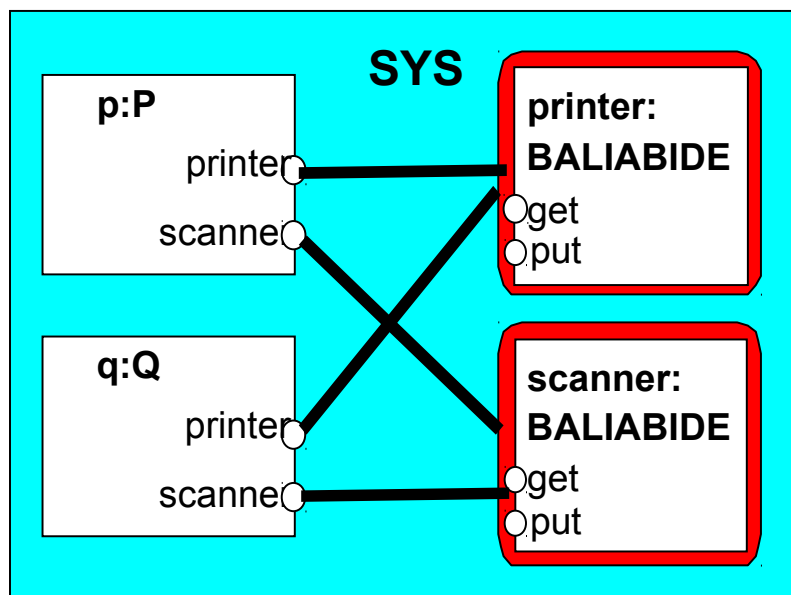
(trazarik motzena **STOP**-era)

Trace to DEADLOCK:
ezker
ezker

Normalean guri interesatuko zaizkigun blokeaketak ez dira izango STOP erabiliz explizitoki erazagutzen direnak.

Elkar-blokeaketaren analisia – Konposaketa paraleloa

- Elkarreragiten duten prozesuen **konposaketa paraleloa**atik eman daiteke elkar-blokeaketa sistemetan.



Blokeorako traza?

Nola saihestu?

```
P = ( printer.get->scanner.get
      ->kopiatu
      ->printer.put->scanner.put
      ->P ) .
```

```
Q = ( scanner.get->printer.get
      ->kopiatu
      ->scanner.put->printer.put
      ->Q ) .
```

```
BALIABIDE = BALIABIDE[0] ,
BALIABIDE[h:0..1] =
  (when (h==0) get->BALIABIDE[1]
   |when (h==1) put->BALIABIDE[0]) .
```

```
||SISTEMA = (p:P||q:Q
              ||{p,q}::printer:BALIABIDE
              ||{p,q}::scanner:BALIABIDE
              ) .
```

Elkar-blokeaketaren analisia – Nola saihestu

- ◆ Baliabideak ordena berean eskuratuz.
- ◆ Denbora-muga:

```
P          = (printer.get->GETSCANNER) ,
GETSCANNER = (scanner.get->kopiatu
               ->printer.put->scanner.put->P
               | denboramuga->printer.put->P
               ) .

Q          = (scanner.get->GETPRINTER) ,
GETPRINTER = (printer.get->kopiatu
               ->printer.put->scanner.put->Q
               | denboramuga->scanner.put->Q
               ) .
```

**LTSA-rekin
analizatu:**
- **Safety?**
- **Progress?**

Denbora-mugarena ez da soluzio ona, gerta daitekeelako bi prozesuak lehenengo baliabidea eskuratzea, denbora-muga iristea, askatu eta berriz lehenengoa eskuratzea... ziklo hau errepikatuz aurrerapenik egin gabe. Arazo hau datorren gaian aztertuko dugu.

6.2 Filosofoen afaria

Bost filosofo esertzen dira mahai borobil batean.

Filosofoek bere bizitza ematen dute honela:

une bakoitzean **pentsatzen** edo **jaten**.

Hau bizitza gogorra!

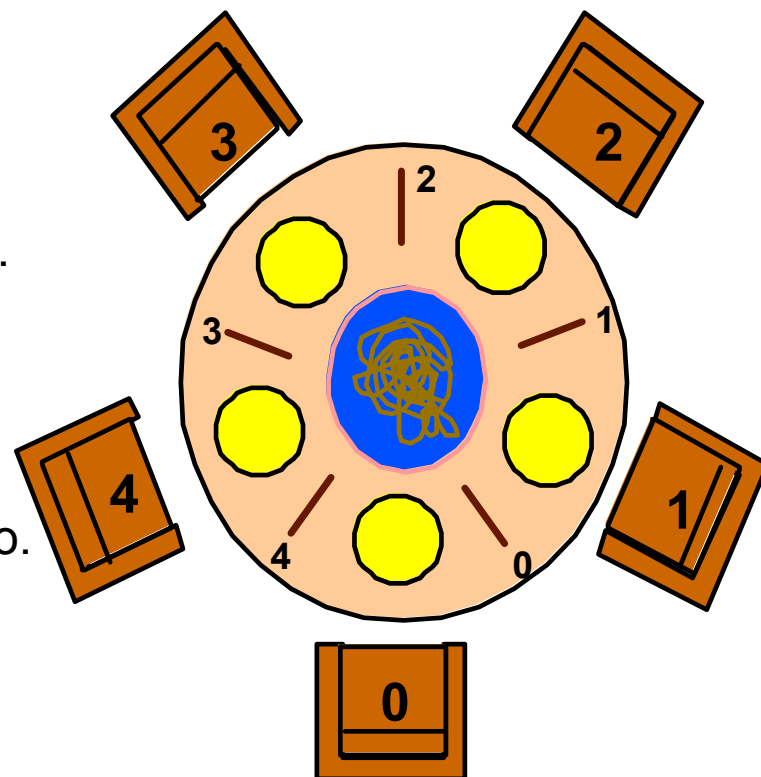
Mahaiaren erdian spaghetti plater handi bat dago.

Filosofo bakoitzak bi sardeska behar ditu spaghettiak jateko.

Filosofoen soldata informatikariena bezain txarra denez soilik bost sardeska dituzte.

Sardeska bakoitza bi filosofoen artean jarrita dago.

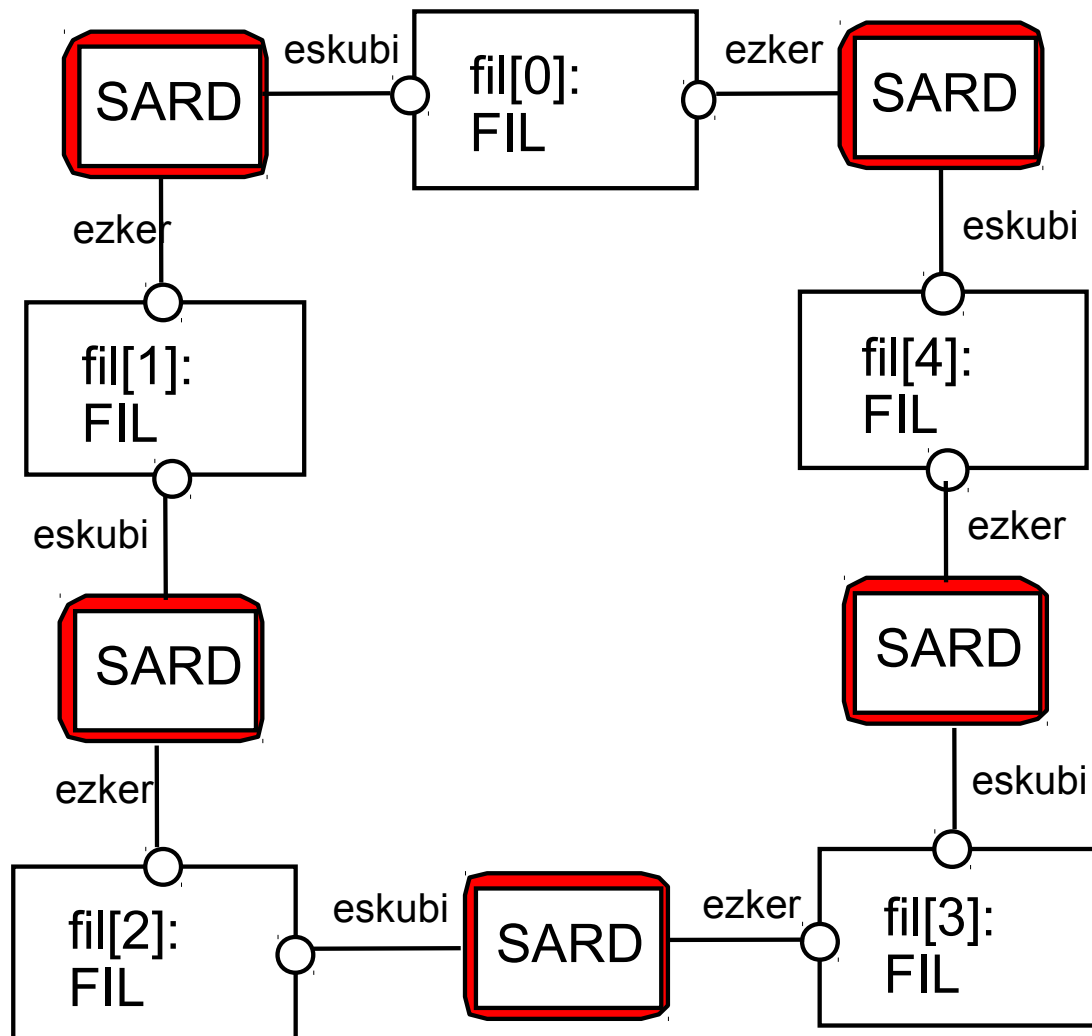
Filosofo bakoitzak soilik bere ezkerreko eta bere eskubiko sardeskak erabiliko ditu.



Filosofoen afaria – Egitura diagrama modelatu

SARDeska bakoitza hartu eta **utzi** ekintzak dituen **baliabide konpartitu** bat da.

Gose denean FIlosofo bakoitzak eskubi eta ezkerreko SARDeskak hartu behar ditu jaten hasi baino lehen.



Filosofoen afaria - Eredua

```
FIL = (eseri->eskubi.get->ezker.get  
      ->jan->eskubi.put->ezker.put  
      ->altxatu->FIL) .
```

```
SARD = SARD[0] ,  
SARD[h:Bool] = (when (h==0) get->SARD[1]  
                  |        put->SARD[0]) .
```

Filosofoen mahaia:

```
|| AFARIA(N=5) = forall [i:0..N-1]  
    (fil[i]:FIL ||  
     {fil[i].ezker, fil[(i-1)+N]%N}.eskubi} :: SARD  
    ) .
```

Eman daiteke elkar-blokeaketa?

Filosofoen afaria – Ereduaren analysisia

Trace to DEADLOCK:

```
fil.0.eseri  
fil.0.eskubi.get  
fil.1.eseri  
fil.1.eskubi.get  
fil.2.eseri  
fil.2.eskubi.get  
fil.3.eseri  
fil.3.eskubi.get  
fil.4.eseri  
fil.4.eskubi.get
```

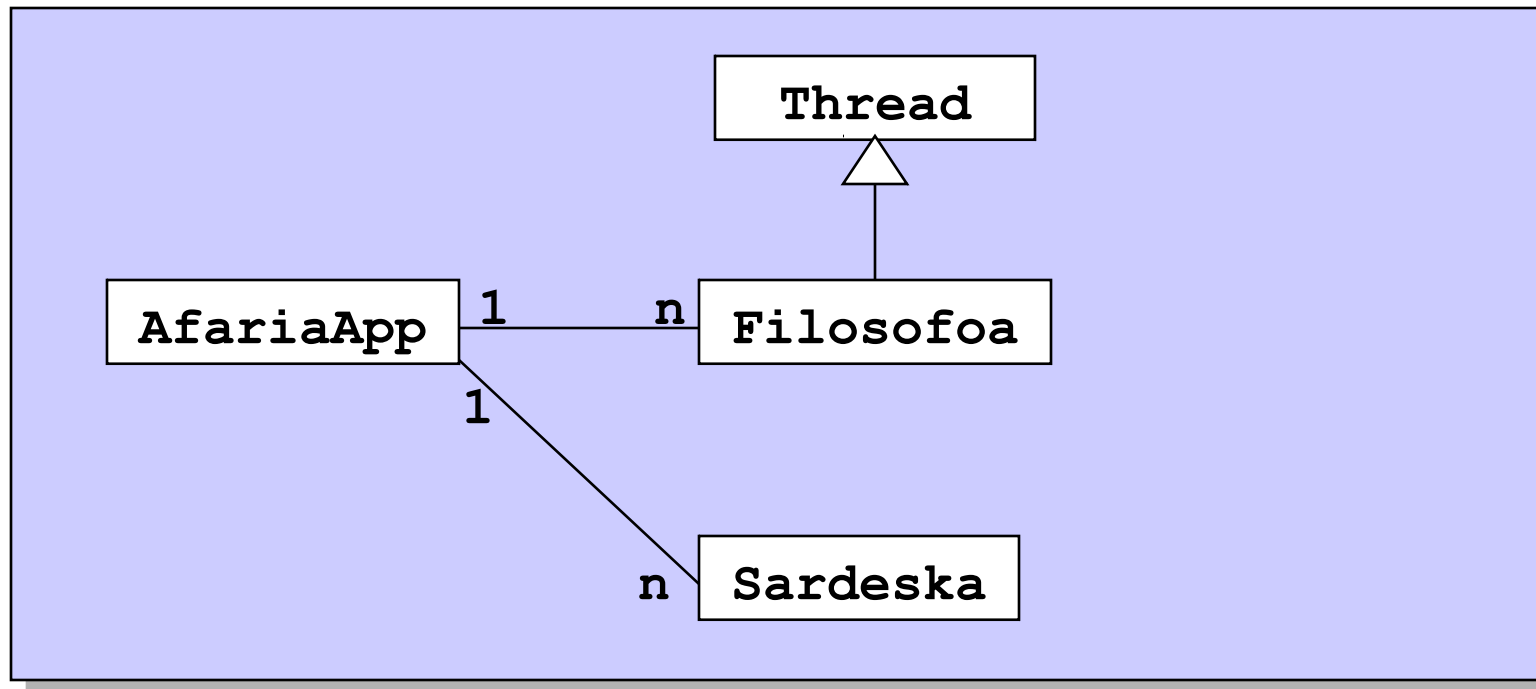
Filosofo guztiak gose dira aldi berean,
esertzen dira
eta filosofo bakoitzak
bere **eskubiko** sardeska hartzen du.

Sistemak ezin du aurreratu,
filosofo bakoitzak
bere ondokoak heltzen duen
sardeskaren zain dagoelako...
zain ziklo bat dago!!!

Elkar-blokeaketa erraz aurkitu dugu **ereduan**.

*Erreza al da elkar-blokeaketa posiblea aurkitzea **implementazioan**?*

Filosofoen afaria – Java-n implementazioa



- ◆ **filosofoak**: entitate aktiboak -> hariak
- ◆ **sardeskak**: entitate pasibo konpartituak -> monitoreak

Filosofoen afaria - Sardeska **monitoreo**

```
class Sardeska {  
    private boolean hartua=false;  
    private int zenbakia;  
    private Pantaila pant;  
  
    Sardeska(int zenb, Pantaila p){  
        zenbakia = zenb; pant=p; }  
  
    public synchronized void put() {  
        hartua=false;  
        pant.idatzi(zenbakia+" utzia ");  
        notify();  
    }  
  
    //when (h==0) get->SARD[1]  
    public synchronized void get() throws java.lang.InterruptedException {  
        while (hartua) wait();  
        hartua=true;  
        pant.idatzi(zenbakia+" hartua");  
    }  
}
```

hartua-k sardeskaren
egoera kodetzen du

Filosofoen afaria – Filosofoa **haria**

```
class Filosofoa extends Thread {  
    ...  
    while (true) {  
        pant.idatzi(zenb, "pentsatzen");    lo(1000);  
        pant.idatzi(zenb, "gose");          lo(1000);  
        eskubikoa.get() ;  
        pant.idatzi(zenb, "eskub.hartu du"); lo(1000);  
        ezkerrekoa.get() ;  
        pant.idatzi(zenb, "ezker.hartu du"); lo(1000);  
        pant.idatzi(zenb, "jaten");          lo(500);  
        eskubikoa.put() ;  
        pant.idatzi(zenb, "eskub.utzi du");  lo(1000);  
        ezkerrekoa.put() ;  
        pant.idatzi(zenb, "ezker.utzi du");  lo(1000);  
    }  
}
```

Eredua jarraituz
(eseri eta altxatu kendu dira)

Probatu lo denborak aldatzen

Filosofoen afaria – Filosofoak eta sardesken sorkuntza

Filosofo hariak eta sardeska monitoreak sortzeko kodea:

```
for (int i=0; i<N; ++i)
    sar[i] = new Sardeska(i,pant);

for (int i=0; i<N; ++i){
    fil[i] = new Filosofoa(i, sar[(i-1+N)%N],
                           sar[i], pant);
    fil[i].start();
}
```

Filosofoen afaria

Sardeak	Fil[0]	Fil[1]	Fil[2]	Fil[3]	Fil[4]
	pentsatzen		pentsatzen	pentsatzen	
		pentsatzen			pentsatzen
		gose	gose		gose
	gose			gose	
4 hartua					eskub.hartu du
1 hartua					
3 hartua		eskub.hartu du			
					ezker.hartu du jaten
2 hartua			eskub.hartu du		
4 utzia					eskub.utzi du
3 utzia					ezker.utzi du
3 hartua				eskub.hartu du	
0 hartua	eskub.hartu du				
4 hartua					pentsatzen
	ezker.hartu du				gose
	jaten				
0 utzia	eskub.utzi du				
0 hartua		ezker.hartu du			
4 utzia	ezker.utzi du				

Filosofoen afaria

Proba ezazue filosofoek jaten eta pentsatzen ematen duten denbora aldatzen. Gutxiagotzen badugu ziurtatuko dugu noizbait elkar-blokekatea gertatzea, hori gertatzeko probabilitatea handitzen delako.

Sardeak	Fil[0]	Fil[1]	Fil[2]	Fil[3]	Fil[4]
	pentsatzen				
3 hartua		pentsatzen gose		pentsatzen gose	
1 hartua	gose			eskub.hartu du	
2 hartua		eskub.hartu du	pentsatzen gose		
0 hartua			eskub.hartu du		pentsatzen gose
4 hartua	eskub.hartu du				eskub.hartu du

Filosofoen afaria elkar-blokeaketarik gabe

Elkar-blokeaketa saihestu daiteke zain ziklo bat ez dela ematen ziurtatzen badugu.

Nola?

Sartu *asimetria* bat filosofoen gure definizioan.

Erabili filosofoen zenbakia:

- zenbaki **bakoitia** dutenek aurretik **ezkerreko** sardeska hartzen dute, eta
- zenbaki **bikoitia** dutenek aurretik **eskubikoa**.

```
FIL(I=0) = (
  when (I%2==0) eseri ->ezker.get->eskubi.get->jan
                        ->ezker.put->eskubi.put->altxatu->FIL
| when (I%2==1) eseri ->eskubi.get->ezker.get->jan
                        ->ezker.put->eskubi.put->altxatu->FIL ) .
|| AFARIA(N=5)=forall[i:0..N-1]
    (fil[i]:FIL(i) ||
    {fil[i].ezker,fil[(i-1)+N)%N].eskubi}::SARD ) .
```

Beste estrategiarik?

Ariketak

1. Filosofoen afaria asimetriarekin Java-z implementatu
2. Filosofoen afarirako beste soluzio bat:
bost filosofoetatik soilik lau filosofo eser daitezke batera.
MAIORDOMO prozesu bat espezifikatu, proposatutako ereduarekin konposatzean, gehienez lau filosofo **eseri** ekintza egitea baimentzen duena, **altxatu** ekintza bat gertatu aurretik.
Frogatu ez dela elkar-blokeaketarik ematen.
FSPz modelatu eta Javaz implementatu.
3. Javaren **wait** primitiba erabiliz **Sardeska** monitorearen implementazioa aldatu, segundu bat zain egon ondoren **get** deia itzultzeko false balioarekin.
Filosofoak beste sardeskarik bazuen, askatu beharko du eta berriz saiatu.
Aztertu sistemaren jokaera.

```
public final void wait(long denboramuga)
    throws InterruptedException
```
- 4**. Asmatu problema bat, apunteetan emandakoetatik oso desberdina, non elkar-blokeaketa ematen den.
Blokeaketa ematen ez den bertsioa ere egin. FSPz eta Javaz.