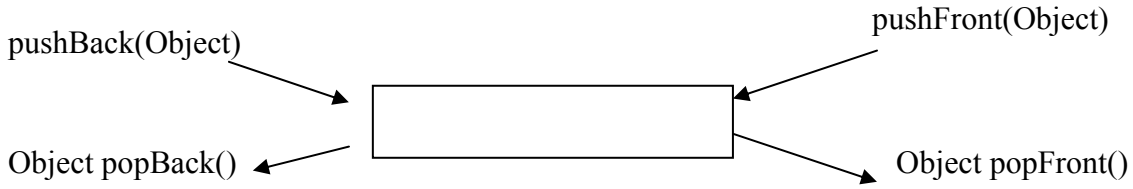
DATU-EGITURAK ETA ALGORITMOAK I - SIIT 2. MAILA
2003ko urtarrilaren 21a

Ariketa 1 (3 puntu)

DoubleList klasea inplementatu. Egitura honetan elementuak txertatu/lortu daitezke edozein muturretatik kopuru mugatu gabe.



Hurrengo klasearen egitura eta metodoen inplementazioa garatzea eskatzen da

DoubleList klasea

```
class DoubleList {

void init();
-- post: Bil bilara hasieratu egiten du

boolean isEmpty();
-- post: True itzultzen du b.s.b. Bil bilara hutsik badago.

void pushBack(Object elem);
-- post: elem elementua ezkerreko muturretik sartzen du

void pushFront(Object elem);
-- post: elem elementua eskuineko muturretik sartzen du

Object getBack();
-- post: ezkerreko muturreko elementua itzultzen du ezabatu gabe

Object getFront();
-- post: eskuineko muturreko elementua itzultzen du ezabatu gabe.

Object popBack();
-- post: ezkerreko muturreko elementua itzultzen du eta ezabatzen du.

Object popFront();
-- post: eskuineko muturreko elementua itzultzen du eta ezabatzen du.
}
```

Ebazpena

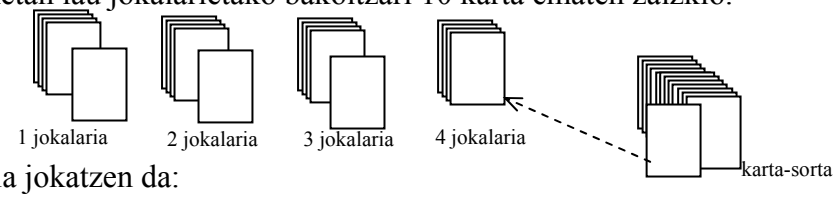
```
public class DoubleList<T> {
  Node<T> back;
  Node<T> front;
  public DoubleList() {
   back=null;
   front=null;
  }
  public void pushBack(T o) {
   Node<T> n=new Node<T>(o);
   if (back==null) { //empty list
    back=n;
    front=n;
   } else { //not empty list
      back.left=n;
      n.right=back;
      back=n;
    }
  }
  public void pushFront(T o) {
   Node<T> n=new Node<T>(o);
   if (front==null) { //empty list
    front=n;
    back=n;
   } else { //not empty list
      front.right=n;
      n.left=front;
      front=n;
    }
  }
  public T getBack() {
   return back.content;
  }
  public T getFront() {
   return front.content;
  }
  public T popBack() {
   T aux;
   if (back==null) return null; //empty list
   //not empty list
   aux=back.content;
   if (back==front) {
    back=null;
    front=null;
   } else {
     back=back.right;
     back.left=null;
    }
   return aux; }
```

```
public T popFront() {
   T aux;
   if (front==null) return null; //empty list
   //not empty list
   aux=front.content;
   if (back==front) {
    back=null;
    front=null;
    } else {
      front=front.left;
      front.right=null;
     }
    return aux;
  }
}
```

Ariketa 3 (4 puntu)

Lau lagun SEIKOEN karta-jokoan hastekotan dira. Jokoaren ezaugarriak honakoak dira:

- 40 karta daude lau palotan bereizita (urrea, kopa, ezpata eta bastoia). Palo bakoitzak 10 karta ditu 1etik 10era zenbakituta.
- Banaketan lau jokalarietako bakoitzari 10 karta ematen zaizkio.



- Horrela jokatzen da:
  - Lehenengo jokalariak bere kartetatik gainean dagoenarekin jokatzen du:
    - Ahal izanez gero mahaian jartzen du.
    - Ezin badu mahaian jarri, karta hori bere karta guztien azpian jartzen du.
  - Txanda hurrengo jokalariari pasatzen zaio.
- Jokalari batek bere karta mahaian jartzeko bi aukera dauzka:
  - Karta 6koa da.
  - Karta ez da 6koa, baina mahaian dagoen kartaren baten segidakoa da, adibidez, 2ko bastoia eskuan edukita mahaian 3ko bastoia egongo balitz, edo 8ko urrea edukita mahaian 7ko urrea balego.
- Jokoa amaitzen da jokalari batek bere azken karta jartzen duenean, hau da, kartarik gabe geratzen denean.

Ariketa honetan erabili daitezken klaseak KartaSorta eta DoubleList dira. Azkeneko klase honetan elementuak txertatu/atzitu daitezke bi muturretatik. **(ez da inongo eragiketarik inplementatu behar)**. Klase honen espezifikazioa lehenengo ariketan dago.

KartaSorta klasea

```
public class KartaSorta
{
  public void hasieratu()
  post: Ks karta-sortak 40 karta dauzka ausazko ordenan

 public Karta getCarta()
  -- aurre: karta-sortak badu gutxienez karta bat.
  -- post: karta-sortan zegoen lehenengo Karta itzultzen da,
           eta karta hori karta-sortatik kendu egin da.

  public boolean isEmpty() {}
  - post: True itzultzen du b.s.b. kartarik ez badago.

}
public class Karta
 {
    public int zenbakia;
    public int paloa;
 }
```

Eskatzen da:

a) Definitu Javan jokalarien kartak eta joko-mahaiaren egoera errepresentatzeko behar diren klaseak.

b) Partida bat simulatzen duen metodoa diseinatu eta idatzi. Metodo horrek irabazlea zein izan den idatziko du pantailan.

Ebazpena

a) Datu Egiturak

```
public class Jokalari {
 DoubleList<Karta> bereKartak;
}
public class Partida
{
  Jokalari[] jokalariak=new Jokalari[4];
  DoubleList<Karta>[] tablero=new DoubleList<Karta>[4];
  Baraja baraja=new Baraja();
  int txanda =0;
}
```

b) Diseinua

partida Simulatu

```
1.kartakBanatu
2 irabazle=partidaJokatu
Idatzi irabazlea
```

1. kartak banatu

```
Baraja hasieratu
Jokalariak hasieratu
while kartak bajaran
  karta bat lortu barajatik
  uneko jokalariari karta eman
  hurrengo jokalarira joan
```

2 **patida jokatu**

```
txanda=lehendabiziko jokalaria
irabazle=false;
while (ez irabarlerik) {
       jolariaren goiko karta lortu => k
       while (2.1 kartaIpini(k) && jokalariak kartak gehiago {
        jolariaren goiko karta lortu => k
       }
       if (jokaria ez dauka kartarik)
        irabazle=true;
        else {//ezin izan du karta ipini
         karta ipini atzeko partean
         txanda pasa hurrengo jokalariari
        }
      }
txanda itzuli irabazle bezala
```

2.1 boolean kartaIpini(Karta)

```
if karta 6 bada then bere paloan kokatu eta true itzuli
 else
  if ez badago palo horretako kartarik then  false itzuli
   else
    if  karta 6 baino txikiago eta palo horretako listan beheko karta
zenbaki bat haundiago bada then palo horretako beheko partean kokatu
karta eta true itzuli
     else
       if  karta 6 baino handiagoa eta palo horretako listan goiko
karta zenbaki bat txikiagoa bada then palo horretako goiko partean
kokatu karta eta true itzuli
  else false itzuli
```

c) Inplementazioa

```
public class Jokalari {
 DoubleList<Karta> bereKartak;
  public Jokalari() {
   bereKartak=new DoubleList<Karta>();
  }
  public void kartaBanatu(Karta pKarta) {
   bereKartak.pushFront(pKarta);
  }
  public Karta getKarta() {
   return bereKartak.popFront();
  }
  public void putKarta(Karta pKarta) {
   bereKartak.pushBack(pKarta);
  }
  public boolean hasKartak() {
   return (!bereKartak.isEmpty());
  }
}
```

2. public class Partida {

```
 Jokalari[] jokalariak=new Jokalari[4];
 Baraja baraja=new Baraja();
 DoubleList<Karta>[] tablero=new DoubleList<Karta>[4];
 int txanda=0;

 public Partida() {
    //Jokalariak sortu
    for (int j=0;j<3;j++)
      jokalariak[j]=new Jokalari();
    for (int t=0; t<=3;t++)
      tablero[t]=new DoubleList<Karta>();
 }
  public void txandaPasa() {
       txanda=(txanda +1)%4;
  }
 public void kartakBanatu() {
  int j=0;
  while (!baraja.isEmpty()){
   jokalariak[j].kartaBanatu(baraja.getKarta());
   j=(j+1)%4;
  }
 }
 public boolean kartaIpini(Karta k) {
  int pos=k.paloa ;
  if (k.zenbakia==6){
   tablero[pos].pushBack(k);
   return true;
  }
   else
   {
    if (tablero[pos].isEmpty())
      return false;
     else {
       Karta front=tablero[pos].front();
       if (front.zenbakia-1==k.zenbakia){
          tablero[pos].pushFront(k);
          return true;
       }
      else {
        Karta back=tablero[pos].back();
        if (back.zenbakia+1==k.zenbakia){
         tablero[pos].pushBack(k);
         return true;
        }
      }
     }
  }
  return false;
 }
```

```
  public int partidaJokatu() {
     boolean irabazle=false;
     while (!irabazle) {
      Karta k=jokalariak[txanda].getKarta();
      while (kartaIpini(k) && jokalariak[txanda].hasCartak()) {
       k=jokalariak[txanda].getKarta();
      }
      if (!jokalariak[txanda].hasCartak())
       irabazle=true;
       else {
        jokalariak[txanda].putKarta(k);
        txandaPasa();
       }
     }
  return txanda;
  }
}
```

3. Nagusia

```
   public static void main(String[] args){
              //Partida berria sortu
           Partida p=new Partida();

           p.kartakBanatu();

           int irabazle=p.partidaJokatu();
           System.out.println(irabazle);
   }
```