SARRERA

Gaur arte bakarrik lan egin dugu errealitate indibidualak errepresentatzen duten klaseekin. Adibidez, Pertsona klasea, zeinek pertsona baten abstrakzioa errepresentatzen duen. Baina, zer gertatzen da errealitate indibidualen multzo bat adierazi nahi dugunean?
Orain arte ezagutzen zenuten lengoaian (ADA) eta datu motekin, elementu multzo bat adierazi nahi genuenean 2 eremuz osatutako erregistro bat erabiltzen genuen.

```
================================================================================
package org.pOinarrizkoProg.hainbatPertsona is

//datu motak
type Pertsona is record
   izena:String;
   adina: integer;
   nan:String;

type Tpertsonak is  array(Integer range<>) of Pertsona;
type ListaPertsonak is record
   luzera: integer;
   lista:Tpertsonak;
end record;

//eragiketak

procedure gehituPertsona(plP: in out ListaPertsonak; pPertsona: in Pertsona) is

begin
   //gehitzearen kodea
end  gehituPertsona;

procedure eliminatuPertsona(plP: in out ListaPertsonak; pPertsona: in Pertsona)
is

begin
   //eliminatzearen kodea
end  eliminatuPertsona;

..... eta ListaPertsonak motarekin egin daitezken gainontzeko eragiketak....

end package;
```

Orain, OB ikuspegitik, arrayen antzeko adierazpen bat erabiltzeko aukera ere badago, beti luzera kudeatzen noski. Ez dugu pakete bat izango baizik eta klase bat. **ListaPertsonak klasea.** Klase honek `lista eta luzera` izango ditu atributu gisa eta metodo gisa, ADAko paketean agertzen diren eragiketa berak (hau da, gehituPertsona(), eliminatuPertsona() etab.).

Jarraian **ListaPertsonak** klasea Javaz erakusten da.

PERTSONA KLASEA

```
package org.pmoo.packlistaArrayak;

public class Pertsona {

	//atributuak
		private String izena;
		private int adina;
		private String nan;

	//eraikitzailea
		public Pertsona(String pIzena, int pAdina, String pNan){
			this.izena=pIzena;
```

```
                this.adina=pAdina;
                this.nan=pNan;
            }

      //getters eta setters

           public String getNan(){
                return(this.nan);
            }

            public String getIzena(){
                return(this.izena);
            }

            public int getAdina(){
                return(this.adina);
            }
}
```

PERTSONEN KLASE EDUKITZAILEA (KASU HONETAN ARRAY BATEN BITARTEZ KUDEATUKO DENA)

```
package org.pmoo.packlistaArrayak;


public class ListaPertsonak {
            //atributuak
            private int luzera;
            private Pertsona[] lista;

            //Eraikitzailea
            /**
             * luzera 0ra jarriko du, hasieran lista hutsik egongo baita*/
            public ListaPertsonak(){
                this.luzera=0;
                this.lista=new Pertsona[10];
            }

            //getters eta setters

            /**hitzarmenez, lista objektu bat denez getLista() metodorik gabe
             *atzitzen da. Hemen gehitu dugu zuek array bat nola bueltatzen den
             *ikus dezazuten*/

            public Pertsona[] getLista(){
                return(this.lista);
            }

            public int getLuzera(){
                return(this.luzera);
            }

            private void setLuzera(int pBalioa){
                this.luzera=pBalioa;
            }

            /**
            *setLuzera pribatua izango da, ez dugulako nahi ListaPertsonatik
            *kanpoko beste inork luzera aldatzea ez bada Pertsona berri bat gehitu
            *delako ala Pertsona bat eliminatu delako (eta bi hauek Klase barneko
            *metodoak dira... beraz setLuzera pribatu jartzen badugu erabili
            *dezakete klase barruan daudelako
             */
            private void handituLuzera(){
                this.setLuzera(this.getLuzera()+1);
            }

            /**
             * hau ere pribatua setLuzera bezala eta arrazoi berdinengatik.*/
```

```
  private void txikituLuzera(){
        this.setLuzera(this.getLuzera()-1);
  }



  //gainontzeko metodoak
public void gehituPertsona(Pertsona pPertsona){
      this.getLista()[this.getLuzera()]=pPertsona;
      this.handituLuzera();
}

/**Hau inplementatu dugu ezabatu ordenatuta izango balitz bezala, baina
  *eraginkorrena printzipioz azkeneko elementuarekin ordezkatu genezake
  *eta luzera txikitu*/
public void eliminatuPertsona(Pertsona pPertsona){
   int pos=this.bilatuPertsona(pPertsona);
   if(pos!=-1){
           this.desplazatuEzk(pos);
           this.txikituLuzera();
       }
}


/**bilatuPertsona era pribatua izan zitekeen printzipioz bakarrik klaseak
  jakin nahiko duelako ea elementuren bat bere baitan dagoen.
  * @param pPertsona pertsona bat izango da
  * Aurre: Berez, ez dugu zertan pertsona topatu behar
  * Post: -1 bueltatuko du pPertsona lista barruan ez balego
 */
private int bilatuPertsona(Pertsona pPertsona){
int posTopatua=-1;
int ind=0;
boolean topatua=false;
      while(ind<this.getLuzera() && !topatua){
          if(this.getLista()[ind].equals(pPertsona)){//helbide bera izan
                                        //behar du, equals funtzionatu dezan
             topatua=true;
             posTopatua=ind;
          }
          else{
            ind++;
          }
      }

  return posTopatua;
  }


/**pribatua hau ere, klase barruan baino ez delako erabiliko
 *
 * @param pNundik <= elementu kop (luzera)
 * Aurre:hutsik egon zitekeen, pNundik <= elemen kop (luzera)
 * Post:pNundik aurrera elementu guztiak ezkerretara desplazatuko ditu*/
 private void desplazatuEzk(int pNundik){
  int ind=pNundik;
  if(this.getLuzera()>=1){
        while(ind<this.getLuzera()-1){
        this.getLista()[ind]=this.getLista()[ind+1];
        ind++;
        }
  }
}


}//Klasearen bukaera
```

Orain, Javako edukitzaileak erabilita (adibidez ArrayList<>), soluzioa jarraian ikusten den bezala izango da. Eta ikusiko denez, errezagoa izango da Javako edukitzaileek jadanik inplementatuta dauden hainbat metodo eskeintzen dituztelako:

- size(): edukitzaileak dituen elementu kopurua bueltatzen du
- contains() elementu bat edukitzailean dagoen ala ez bueltatzen du
- add() edukitzailean elementu bat gehitzen du
- remove() edukitzailetik elementu bat eliminatzen du

PERTSONA KLASEKO HAINBAT ELEMENTU GORDETZEKO KLASE EDUKITZAILEA (KASU HONETAN, ArrayList<> JAVAKO KLASE GENERIKOAREKIN INPLEMENTATUTA)

```
package org.pmoo.packarrayak;
import java.util.*;/*pakete hau behar da, hemen baitaude ArrayList<> eta Iteradore klaseen definizioa*/


public class ListaPertsonak2 {
	//atributuak
	private ArrayList<Pertsona> lista;

	//eraikitzailea
	public PertsonakArrayLista(){
		this.lista=new ArrayList<Pertsona>();
	}

	//getters eta setters
	private ArrayList<Persona> getLista(){
		return(this.lista);
	}

	//gainontzeko metodoak
	public int getLuzera(){
		return(this.getLista().size());
	}

	public void gehituPertsona(Pertsona pPertsona){
	  this.getLista().add(pPertsona);
	}

	public boolean badagoPersona(Pertsona pPertsona){
		this.getLista().contains(pPertsona);
	 }

	 public void eliminatuPertsona(Pertsona pPertsona){
	   this.getLista().remove(pPertsona);
	 }
	 //ikus dezazuten ArrayList bat nola zeharkatzen den iteradorearekin
	 public void inprimatuLista(){
	 Pertsona pertsonaBat;
	 Iterator<Pertsona> itr=this.getLista().iterator();//iteradorea lortu
		while(itr.hasNext()){//Begiratzen dugu ea iteradoreak elementuak dituen
		    pertsonaBat=itr.next();//Egungo elementua lortu eta aurrera jo
		    System.out.println(pertsonaBat.getIzena()+" "+personaBat.getNan());
		}
	 }
}
```

IKASKETA HELBURUA

JAVAKO EDUKITZAILE BAT ERABILTZEAREN ABANTAILAK (kasu honetan ArrayList<>)

Hemen ikusten da metodo gutxiago inplementatu behar ditugula, Edukitzaileak berak inplementatuta dakarkilako. Ez da behar **.bilatuPertsona(Pertsona pPertsona)** ezta **.desplazatuEzk(int pNundik)** elementu bat eliminatzeko ArrayList<> clase generikoak **.remove** metodo bat daukalako, eta metodo honek bere baitan bilaketa eta desplazamendua egiten duelako, baita luzeraren kudeaketa **.size()**

.gehituPertsona() ere sinplifikatzen da, ArrayList<> klase generikoak **.add()** metodoa duelako eta elementu bat gehitzeaz (hau da desplazamendua egiteaz) eta luzera kudeatzeaz arduratzen da.

Beraz, ez dugu beharko ez **HandituLuzera()**, ezta **txikituLuzera()**.

Horretaz gain, ez naiz arduratu behar ez alokatutako lekuaren muga gainditu dudan. ArrayList<> batentzat ez da tamaina finkorik alokatu behar (gogoratu A atalean 10 zen).

DESABANTAILAK

Iteradore bat ez duzuela inoiz erabili, baina behin ohitura hartzen duzuela, erreza bihurtzen da. Ariketa pare bat egin ostean menperatuko duzue.

GAURKO KLASEKO GAUZA INTERESGARRIAK:

```
public void inprimatuLista(){
       Pertsona pertsonaBat;
       Iterator<Pertsona> itr=this.getLista().iterator();
            while(itr.hasNext()){
                personaBat=itr.next();
                System.out.println(pertsonaBat.getIzena()+"
"+pertsonaBat.getNan());
            }
       }
```

VERSUS

```
public void inprimatuLista(){
       Pertsona personaBat;
       Iterator<Pertsona> itr=this.getLista().iterator();
            while(itr.hasNext()){


                System.out.println(itr.next().getIzena()+" "+itr.next().getNan());
            }
       }
```

Suposatu dezagun lehenengo pertsona **Maria 30666980C** dela eta bigarrena **Jon 10000000H.**

Lehenengo bertsioan (berdean) .next() bueltatzen digun pertsona jasotzen da pertsonaBat aldagaian, gero bere izena atzitzen da eta bere nan inprimatzeko. Eta inprimatuko litzateke:

Maria 30666980C
eta bigarren bueltan
Jon 10000000H

Bigarren bertsioan (gorrian) ez da .next()-ek bueltatzen diguna aldagai batean jasotzen. Eta emaitza lerro bakar bat litzateke

Maria 10000000H ERROREA!! Ikus daitekeenez nan-a hurrengoarena da, Jonena!!

.next() egiten den bakoitzean 2 gauza gertatzen dira, elementua bueltatzen digula eta HURRENGO POSIZIORA pasatzen dela.