2.2 Gaia

Programazio Modularra eta OB Programazioaren Sarrera

Aurkibidea

- Gaiaren Helburuak
- Kontzeptuak barneratzeko analogia
- Objektuak eraikitzen
- **Objektu zerrendak (klase ArrayList)**
- Klaseen arteko erlazioak
- Elementu estatikoak
- EMA vs. DMA
- Klase-diagramaren diseinua

Orain arte (ADAz)

```
package org.pb.listaPertsonak is

type Pertsona is record
  izena:String;
  adina: integer;
  nan:String;
end record;

type Tpertsonak is array
   (Integer range<>) of Pertsona;

type ListaPertsonak is record
  luzera: integer;
  lista: Tpertsonak;
end record;
```

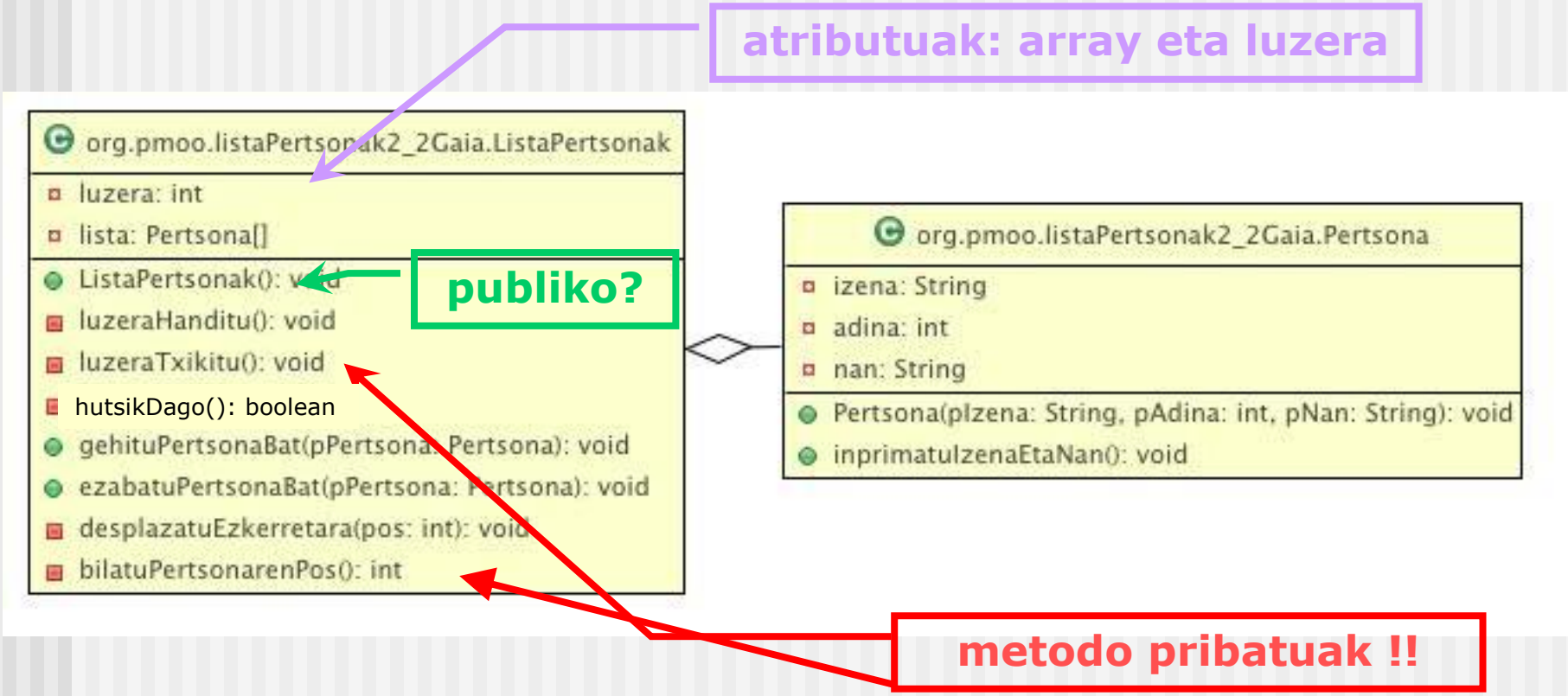
Datu motak

```
procedure gehituPertsona (plP: in out ListaPertsonak;
                          pPertsona: in Pertsona) is
begin
      //gehitzeko kodea
End gehituPertsona;

procedure ezabatuPertsona (plP: in out ListaPertsonak;
                           pPertsona: in Pertsona) is
begin
      //ezabatzeko kodea
end ezabatuPertsona;

// ...eta zerrendak kudeatzeko gainontzeko programak
end package;
```

operazioak

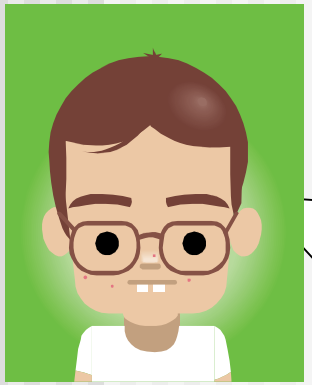
Orain (Javaz)

- OB metodologian gauza bera egin daiteke



Adi, galdera

- Zein izango litzateke hurrengo metodoen inplementazioa...?
  - *gehituPertsona()*
  - *ezabatuPertsona()*



Hurrengo metodoei deitu beharko zaie:

- **luzeraHanditu()**
- **bilatuPertsonarenPos() (-1 ez balego)**
- **desplazatuEzkerrera(pos)**
- **luzeraTxikitu()**

Ondo egina!!, baina...

- Orain arte bezala programatzearen desabantaila
  - Luzera eguneratua mantentzeaz pentsatu behar da
  - Elementu bat ezabatzean desplazatu behar da
  - Zaila da, Pertsonak ez diren Objektuekin sortutako metodoak erabiltzea

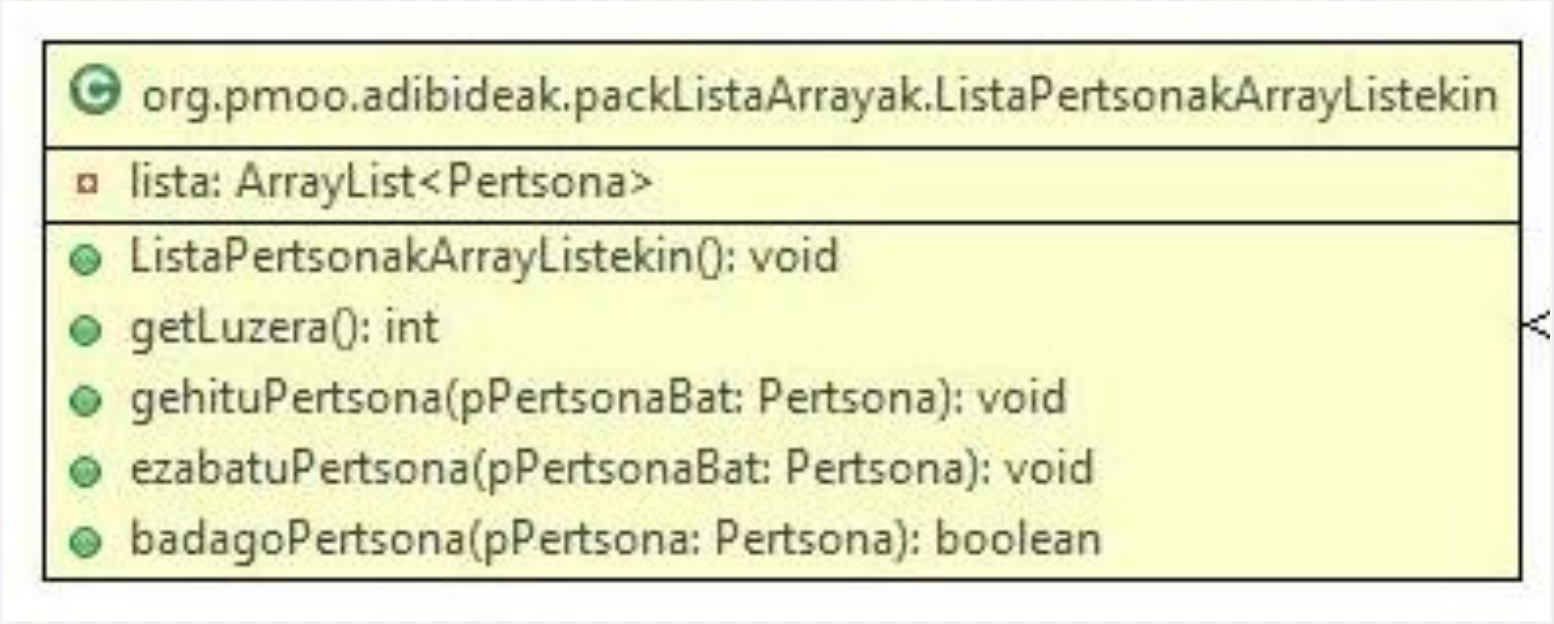
Zelan erreztu lana?

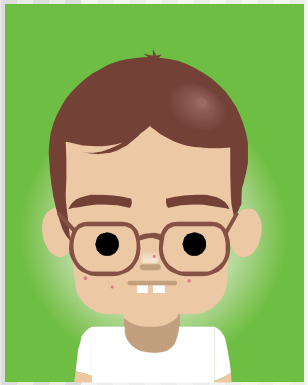
- Javako kontenedore edo kolekzio bat erabiliz
  - **ArrayList**
  - Vector
  - HashSet
  - LinkedList
  - ...

kontenedoreak

- Javako berezko klaseak; objektu zerrendak kudeatzeko oinarrizko metodoak eskaintzen dituzte
  - .add(): elementu bat gehitu
  - .remove(): elementu bat ezabatu
  - .size(): elementuen kopurua bueltatu
  - .contains(): elementu bat barne dagoen adierazi
  - .iterator(): elementuak zeharkatzen ahalbidetu

Orain (ArrayList-ekin)





Luzera atributurik ez dago!!!, ez luzeraHanditu(), ez luzeraTxikitu(), ez luzeraEzarri(), ez bilatuPertsona ezta desplazatuEzker()

inplementazioa

```
public class ListaPertsonakArrayListekin
{
  // atributu
  private ArrayList<Pertsona> lista;

  // eraikuitzailea
 public ListaPertsonakArrayListekin()
   {
    this.lista = new ArrayList<Pertsona>();
   }
```

```
// gainontzeko metodoak

public int getLuzera()
   { return (this.lista.size() ); }

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

public boolean badagoPertsona(Pertsona pPertsona)
   { return (this.lista.contains(pPertsona) ); }

public void ezabatuPertsona(Pertsona pPertsona)
   { this.lista.remove(pPertsona); }

}
```

Ondo Eginda! adi ezkutaketari...

- *.remove()* eta *.add()*
  - Bilatu eta desplazatu elementuak
  - Tamaina eguneratu automatikoki(*.size()*)
- Lista generikoak → kode bererabilgarria edozein elementu motako zerrendekin

.iterator()

- Iterator klaseko elementu bat bueltatzen du
  - Kontenedore baten elementuak zeharkatzeko erabilgarria da
- Bere metodoen artean hurrengoak daude
  - *next()*: egungo elementua bueltatzen du eta hurrengora apuntatzen jartzen da
  - *hasNext()*: hurrengo elementurik dagoen adierazten du

iteradore erabileraren adibidea

```
// Pretsona lista baten iteradore bueltatzen duen metodo pribatua
private Iterator<Pertsona> getIteradorea()
{    return this.lista.iterator();  }

// iteradorea erabilita Pertsona zerrendaren elementuak inprimatzeko metodoa
public void listaInprimatu()
{
 Pertsona personaBat;
 Iterator<Pertsona> itr=this.getIteradorea(); //iterador lortu
 while(itr.hasNext())  // Iteradoreak oraindik elementuren bati apuntatzen dion jakiteko
  {
     pertsonaBat=itr.next();  // egungo elementua bueltatzen du (hurrengoa apuntatuz)
     pertsonaBat.inprimatuIzenaEtaNan();
  }
```

Adi, galdera

- Hurrengo eragiketa taldeen artean ze desberdintasuna dago?

```
while(itr.hasNext()){
        pertsonaBat=itr.next();
        pertsonaBat. inprimatuIzena();
        pertsonaBat. inprimatuAbizena();
}
```

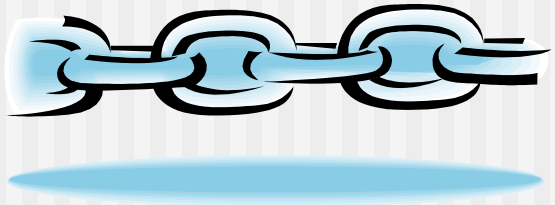
```
while(itr.hasNext()){
        itr.next().inprimatuIzena();
        itr.next().inprimatuAbizena();
}
```

Aurkibidea

- Gaiaren Helburuak
- Kontzeptuak barneratzeko analogia
- Objektuak eraikitzen
- Objektu zerrendak (klase ArrayList)
- **Klaseen arteko erlazioak**
- Elementu estatikoak
- EMA vs. DMA
- Klase-diagramaren diseinua

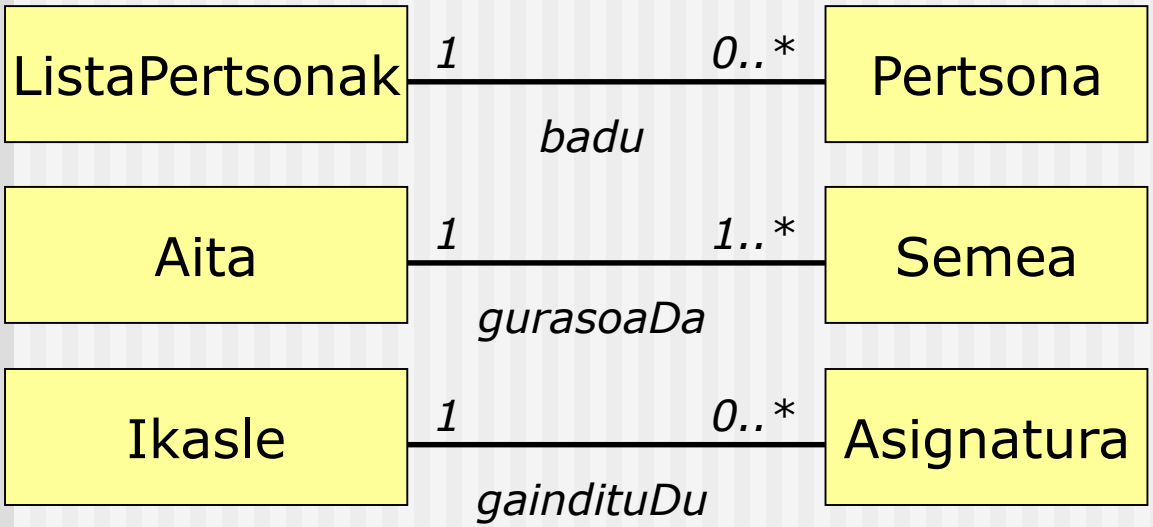
Klaseen arteko erlazioak

- Aplikazio bat = hainbat klase elkar-erlazionatutak
  - Diseinu egokia → ez dago klase isolaturik
- 3 erlazio mota daude
  - Elkartasun (“badu”/“badauka”/“badu erlazioa”)
  - Menpekotasun (“zerbitsua eskatzen dio”/“deitzen dio”)
  - Orokortasuna/Berezitasuna (“bada”/“is-a”)



Elkartasun

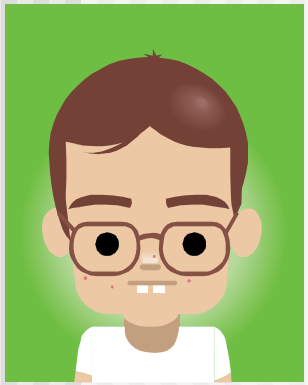
- Erlazio oso estua da
  - "badu"/"badauka"/"erlazioa du"
- Lerro jarrai batez adierazten da
  - Kardinalitate bat esleitzen zaio



Kardinalitatea	Esangura
1	Bakarrik bat
0..1	Zero ala Bat
N..M	Ntik Mra
*	Zero ala Batzuk
0..*	Zero ala Batzuk
1..*	Bat ala Batzuk

Menpekotasuna

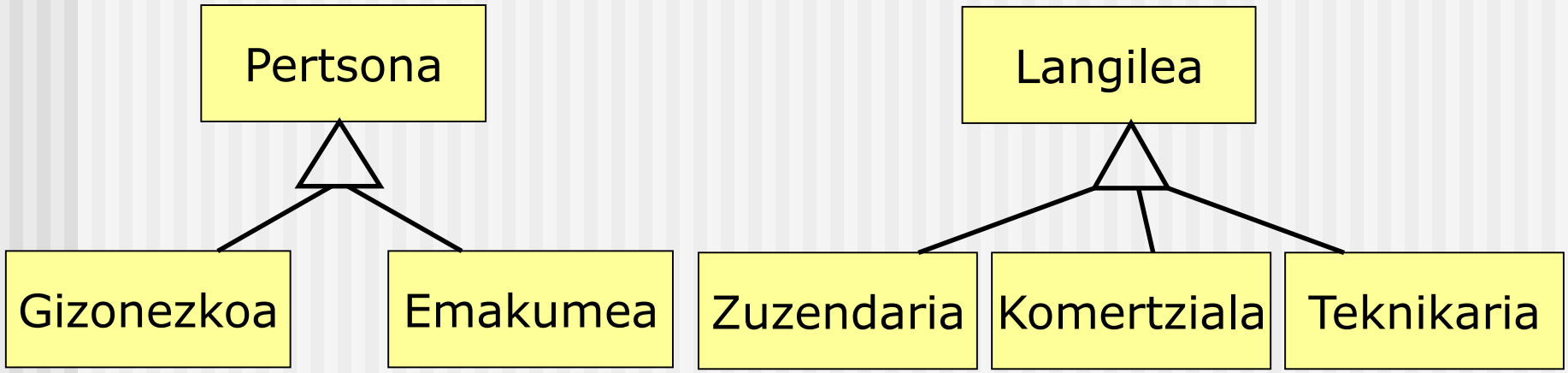
- Erlazio ahula
  - “zerbitzua eskatu”/“deitu”/ “galdetu”
- Gezi ez-jarraiarekin adierazten da
  - Ez da kardinalitaterik adierazten



Kasu honetan bibliotekarioak *ez du liburua* baina liburuak eskaintzen duen zerbitzu (metodo) bati deitzen dio, adibidez, idatzi duen idazlea jakiteko edo mailegatuta dagoen jakiteko

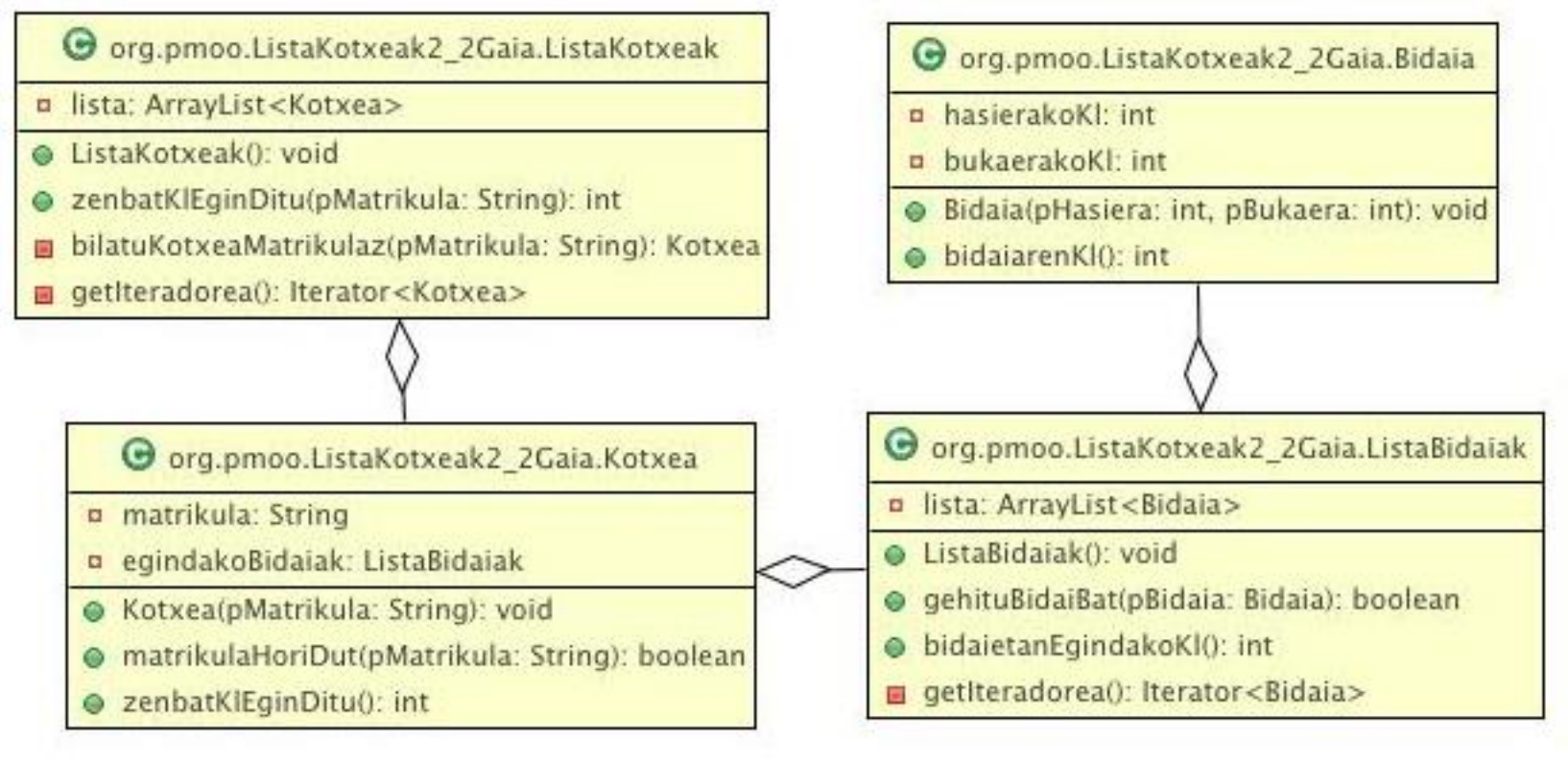
Herentzia erlazioak

- Orokortasun/Berezitasun erlazioa
  - “bada”/“is-a”
- Hiruki batekin adierazten da
- Hurrengo gaian ikusiko da

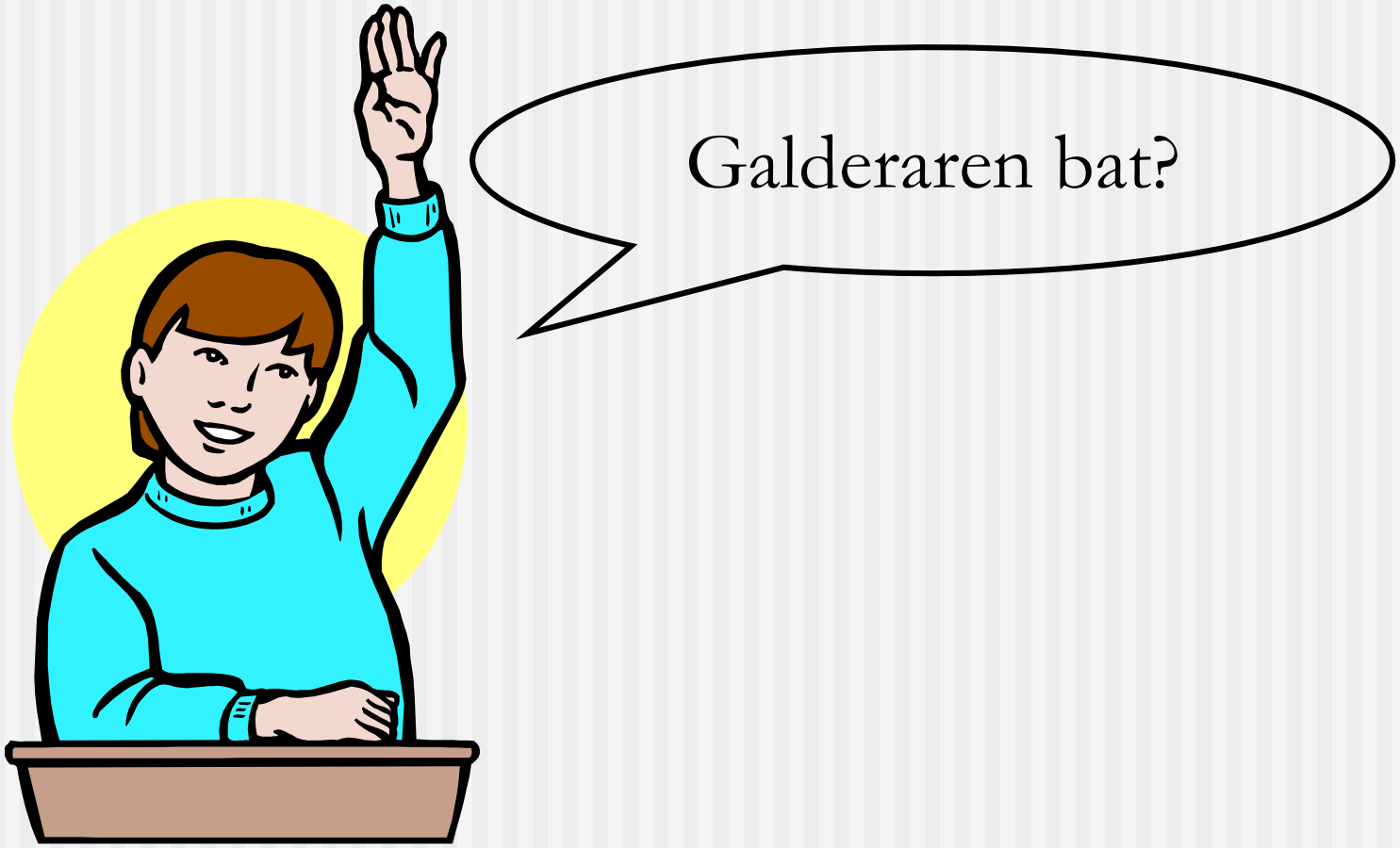


Ariketa bat

Hurrengo klase-diagrameko metodoak inplementatu



```
package org.pmoo.ListaKotxeak2_2Gaia;
import java.util.*;
public class ListaKotxeak {
//atributuak
private ArrayList<Kotxea> lista;
//eraikitzaileak
public ListaKotxeak(){
      this.lista=new ArrayList<Kotxea>();
}
//gainontzeko metodoak
/**
 * @param pMatrikula
* @return -1 ez badu kotxea topatzen bestela egin dituen kilometroak
 */
public int zenbatKlEginDitu(String pMatrikula){
//aldagai lokalak
      int kl=-1;//defektuzko balioa
      Kotxea bilatuNahiDugunKotxea;//zerrendan pMatrikula duen kotxearen
                                   //helbidea gordetzeko aldagai lokala
      bilatuNahiDugunKotxea=this.bilatuKotxeaMatrikulaz(pMatrikula);
      if(bilatuNahiDugunKotxea!=null){
            kl=bilatuNahiDugunKotxea.zenbatKlEginDitu();
      }
      return kl;
}
```


Galderaren bat?