

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

Helburuak

- Klaseak definitu, objektuak sortu eta erabili
- Ataza bat burutzeko, dagokion metodoa inbokatu
- Primitibo vs. erreferentzia (edo erakusle) motak
- Klase predefinituak erabili listak kudeatzeko
- Klaseen diseinua eta euren arteko erlazioak

Aurkibidea



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

Kontzeptuak (gogoratzin)

- Klase = Objektu zehatzen abstrakzioa
 - ❖ Atributuak: forma edo egoera inplementatzen dute
 - ❖ Metodoak: jokaera inplementatzen dute
- Objektu = elementu errealaren nortasun bereziarekin
 - ❖ Intantzia aldagaiak: atributuen instantziak

Analogia (adibidea)

- Suposatu **kotxe bat gidatu nahi dugu, are gehiago, nahi dugu kotxe hori azkar joatea, beraz azeleragailua erabiliko dugu**
- ❖ **Kotxe bat gidatu baino lehen, norbaitek diseinatu behar izan du (azeleradore barne), hau da, norbaitek in behar izan ditu diseinu-planak**



Analogia (adibidea)

- Azeleradoreak ***ezkututzen*** du mekanismo konplexu bat kotxea azkarrago joan dadila
 - ❖ Erreza da azeleratzea, mekanikaz ezer jakin gabe
- Kotxeak ez du bere kabuz azeleratzen
 - ❖ Norbaitek sakatu behar du azeleradoreak

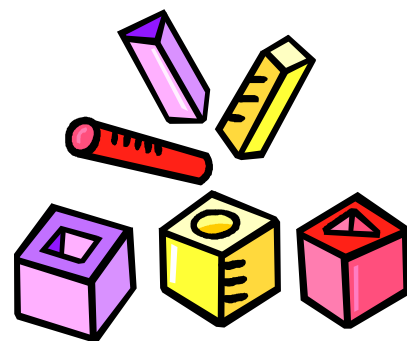


Klase eta objektuak

- Objektu bat klase baten errealizazioa da, kotxe bat diseinu-plano baten errealizazioa den bezala
 - ❖ Kotxe zehatzak diseinu-plano baten oinarrituta eraikitzen dira
 - ❖ Ataza konkretuak objektu konkretu baten gainean gertatzen dira
 - Azeleratzen da benetako azeleradorea sakatuz, ez planoko azeleradorean gainean sakatuz

Atributuak

- Objektuaren forma edo egoeraren balioak gordetzeko erabiltzen dira, ez funtzionalitatea.
- ❖ Adibidez (kotxe): kolore, kilometroak, matrikula...



Instantzia aldagaiak

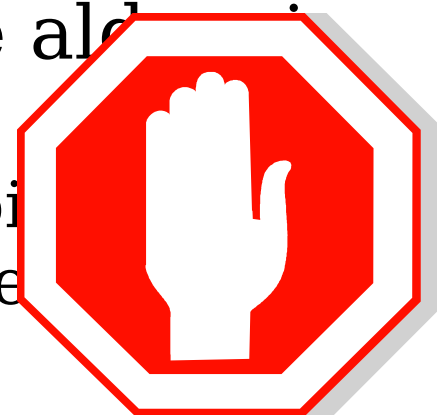
- Objektu baten atributuen balio zehatzak dira, hau da, objektu hori (eta ez beste) definitzen duten balioak
- ❖ Kotxe zehatz bakoitzak (objektu) atributuen balio zehatz bat izango du: kolore gorria, 20000 km, etab.



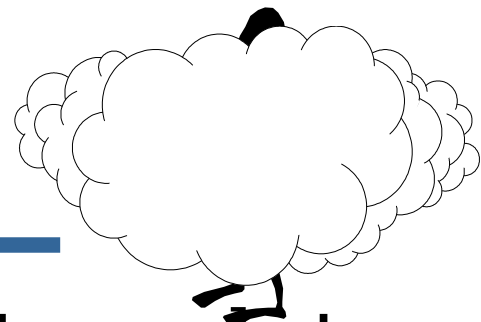
**Batzuetan ikusiko dugu instantzia
aldagaiari atributu ere deitzen
zaia**

Aldagai lokalak

- Atributu vs. Metodo baten aldagai lokala
 - ❖ Atributuak klase baten barruan definitzen dira (klasearen hasieran), eta ez metodo baten barruan.
 - ❖ Metodo batek eduki ditzake aldagai lokalak:
 - tarteko kalkuluak egiteko, adibidez
 - metodoak bueltatu beharko duen balioa gordetzeko.



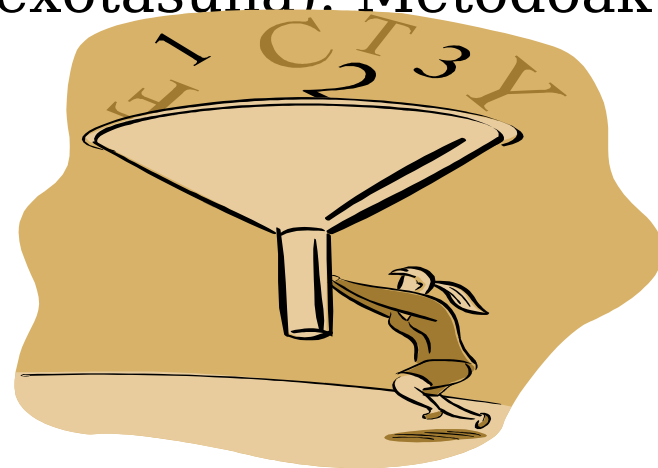
Metodoak



- Ataza edo jokaera bat burutzen dute
 - ❖ Ataza burutzeko pausuak inplementatzen dituzte
- Ez dira berez egikaritzen
 - ❖ Norbaitek deitu behar die
- Detaileak ezkututzen dituzte (datuak eta eragiketak)
 - ❖ Euren inplementazioa ezagutu gabe, erraza da erabiltzea

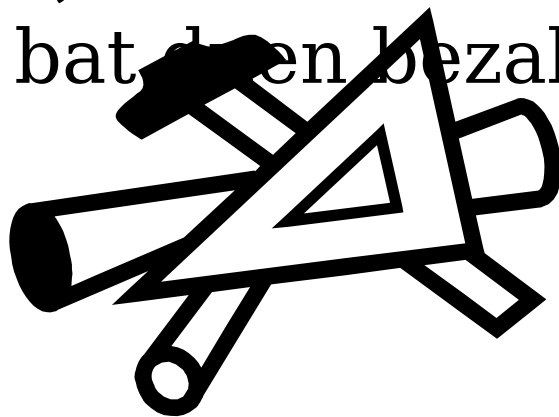
Metodoak

- Programazio prozedimentalean aplikatzen den ideia bera aplikatzen da hemen.
- ❖ Prozedimentalean, azpiprograma batek instrukzioak ezkutatzeko (barneko konplexotasuna). Metodoak \approx azpi-prog.
- ❖ Adibidez: `rem()`



Metodoak

- Java-ko klase bakoitzean, identifikatzen den jokaera edo funtzionalitate bakoitzeko metodo bat inplementatzen da
 - ❖ Horrela, kotxe klaseak *azeleratu()* metodo bat izango du, kotxe baten planoak azeleradore bat duen bezala



Metodo bati deitzen

- Objektu batek ez du ezer egingo, eskaera bat jaso arte
 - ❖ Objektu baten metodo bat egikaritzeko dei bat jaso behar du
 - Modu berdinean kotxe batek ez du azeleratzen norbaitek ez badu azeleradorea sakatzen.
 - ❖ Orduan metodoak zerbitzu bat emango du metodo zerbait bueltatuko du.



Aurkibidea





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

Eraikitzailea

- Klase baten metodo berezia, klase horretako objektuak sortzeko eta hasieratzeko balio dena
- ❖ Bat baino gehiago egon daiteke (parametro desberdinekin)
- ❖ Ez badugu bat ere sortzen, Javak defektuzkoa erabiliko du. Defektuzko eraikitzaileak objektu berriaren atributuak defektuzko balioekin hasieratzen ditu.

Adibidez: Pertsona klasea

```
public class  {  
    //atributuak  
    private double altuera;  
    private int adina;  
    private String izena;  
    private String egunHileJaiotza;  
  
    //eraikitzailea  
    public Pertsona() {  
        this.izen="IzenEzezaguna";  
    }  
}
```



Atributuak
BETI
pribatuak

Adibidea: Pertsona klasea

```
//eraikitzailea parametro posible guztiekin
public Pertsona(String plzen, double pAlt,
                 int pAdin, String pData)  {

    this.altuera=pAlt;
    this.adina=pAdin;
    this.egunHileJaiotza=pData;
    this.izen=plzen;
}
```

Objektuak eraikitzen

➤ Lehenengoz, erakusle bat deklaratu behar da

❖ Erakusle batek objektu bati *apuntatzen* dio

▪ Mota sinpleek, aldiz, oinarritzko balio bat a, erreala, karaktere edo

Pertsona personaBat;

personaBat —||

int osoBat;

Deklaratzen dugunean erakuslea ez dago inora apuntatzen (*null*)

osoBat

?

Objektuak eraikitzen

- **new** eragiketak objektu berri bat sortuko du, objektuarentzat existitzen diren eraikitzailearen bati deituz

```
Pertsona p1 = new Persona();
```

p1 →

altuera: 0.0
adina: 0
izena: "IzenEzezaguna"
egunHileJaiotza : null

```
Pertsona p2 = new Persona("Hegoa Ibarra", 167, 26, "2902");
```

p2 →

altuera: 167.0
adina: 26
izena: "Hegoa Ibarra"
egubHileJaiotza: "2902"

Puntero bakoitzak, memoria zati bati apuntatzen dio, eta hor instantzia aldagaia (objektua) gordetzen da

Adi, galdera

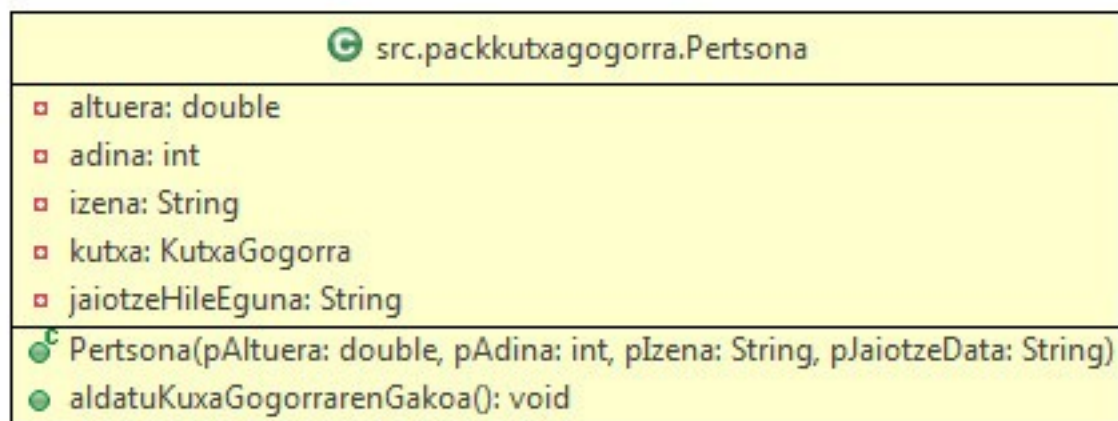
- Non eta noiz erabiltzen da **new** eragiketa?



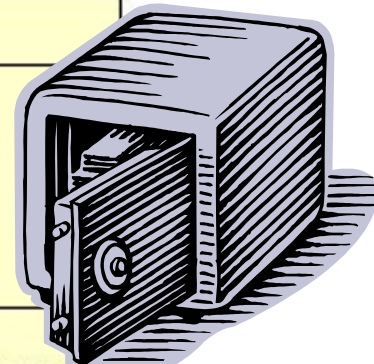
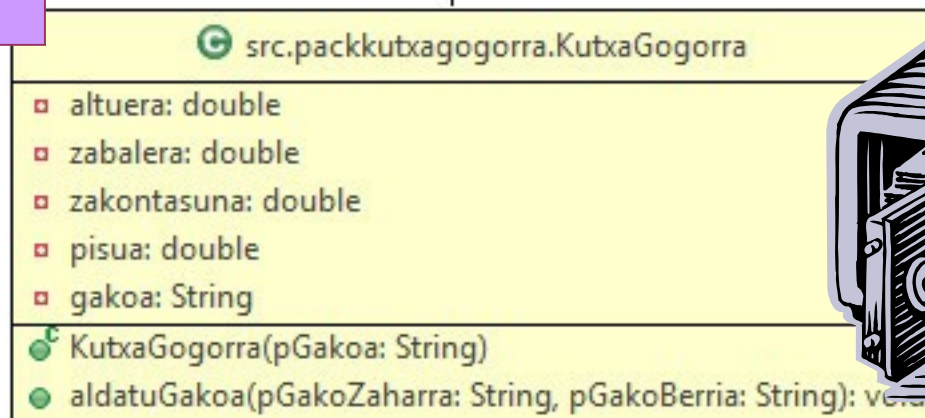
Ariketa ebatzia (zati bat)

- Pertsona klaseari KutxaGogorra klaseko atributu bat gehituko zaio
 - ❖ KutxaGogorra klase berri bat izango da, eta bere atributu bat gakoa String-a izango da

Klase diagrama



Klaseen arteko erlazioa
(elkartasun erl)



Eraikitzaileak




```
public KutxaGogorra (String pGakoa)
{
    this.gakoa = pGakoa;
    //TODO gainontzeko atributuak defektuzko balioekin hasieratu
}
```

```
public Pertsona(String plzen, double pAltuera, int pAdina, String pData)
{
    this.izena = plzen;
    this.altuera = pAltuera;
    this.adina = pAdina;
    this.egunHileJaiotza = pData;
    this.kutxa = new KutxaGogorra(pData);
    //kutxaren defektuzko gakoa pertsonaren jaiotzen data izango da
}
```

Programa nagusia (I)

```
public class Hasieratzailea
{
    public static void main(String[] args)
    {
        Pertsona p1, p2; //memoria alokatze estatikoa erakusleentzat
        p1 = new Pertsona("Isabel", 167,26, "1203");
        p2 = new Pertsona("Angel", 180, 20, "2908");
        // memoria alokatze dinamikoa
        // Pertsona klaseko 2 objektuentzat
    }
}
```

p1 →

altuera: 167.0
adina: 26
izena: Isabel
kutxa: 
egunHileJaiotza: 1203

altuera: 0.0
zabalera: 0.0
sakontasuna: 0.0
pisua: 0.0
gakoa: 1203

Programa nagusia (II)

Hasieratzailearen metodo auxiliarra

...

```
String gakoZaharra = eskatuGakoaTeklatutik();  
String gakoBerria  = eskatuGakoaTeklatutik();
```

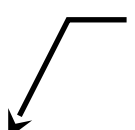
```
p1.norberarenKutxarenGakoaAldatu("1203", "8888");???  
p1.norberarenKutxarenGakoaAldatu(gakoZaharra, gakoBerria);
```

```
} // main bukaera
```

...

Pertsona klasearen metodo publikoa

Erabilitako metodoak



Hasieratzailearen metodo auxiliarra

```
private static String eskatuGakoaTeklatutik()
{
    String gako;
    Scanner sc = new Scanner(System.in);
    System.out.println("Idatzi konbinazio berria: ");
    gako = sc.nextLine();
    return gako;
}
```

Adi, galderak

- Ze nolako rola betetzen dute...?
 - ❖ *aldatuKutxaGogorrarenGakoa()*
 - ❖ *aldatuGakoa()*

