1 Gaia

Sarrera

Aurkibidea

- **Gaiaren helburuak**
- Software-ren eboluzioa
- Modularitatea
- Objektuei Bideratutako Prog.
- Prog. egituratua vs OBP

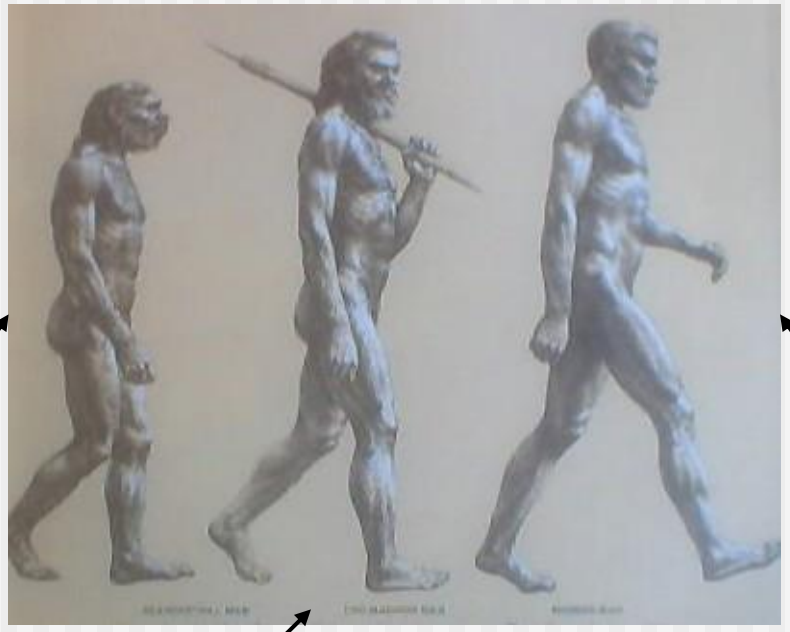
Helburuak

- Software garapenaren eboluzioa
  - Programazio EZ Egituratua
  - Programazio Egituratua
  - Objektuei Bideratutako Programazioa
- Modularitatearen garrantzia
- OBP, Prog. Egituratuaren hautabide bezala

Aurkibidea

- Gaiaren helburuak
- **Software-ren eboluzioa**
- Modularitatea
- Objektuei Bideratutako Prog.
- Prog. egituratua vs OBP

SW garapenaren eboluzioa



Programazio EZ egituratua

Prog. egituratua edo prozedimentala

Modularitatea

Objektuei Bideratutako Programazioa

Prog. EZ egituratua

- Behe mailako lengoaiak
- Kode bloke bakar bat
  - Aldagai guztiak globalak dira

Prog. EZ egituratua

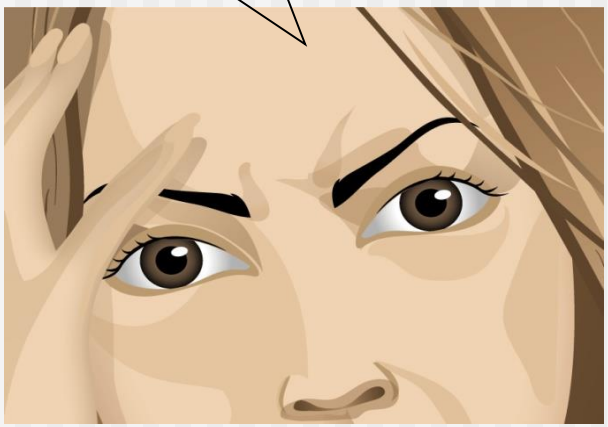
- Arazoak:
  - Ulermena: programaren logika jarraitzea ez da erreza
  - Mantentze lana: aldaketa bakoitza programa osoa gainbegiratzea suposatzen du

Programa EZ Egituratua

idatzi(“Sartu bi zenbaki: “);
irakurri(a, b);
(b = 0) balitz orduan *saltatuAra*(eti3)
c2 ←1
d ←a
eti2: (c2 < b) balitz orduan c1 ←0
bestela *saltatuAra*(eti4)
f ←d
d←0
eti1: d ←d + f
c1←c1 + 1
(c1 < a) balitz orduan *saltatuAra*(eti1)
c2 ←c2 + 1
saltatuAra(eti2)
eti3: d←1
eti4: *idatzi*(d)

Berreketa funtzioa

Ondo, orain ez dago etiketarik, ez saltorik, baina oraindik liatzen naiz hainbeste bukleekin…

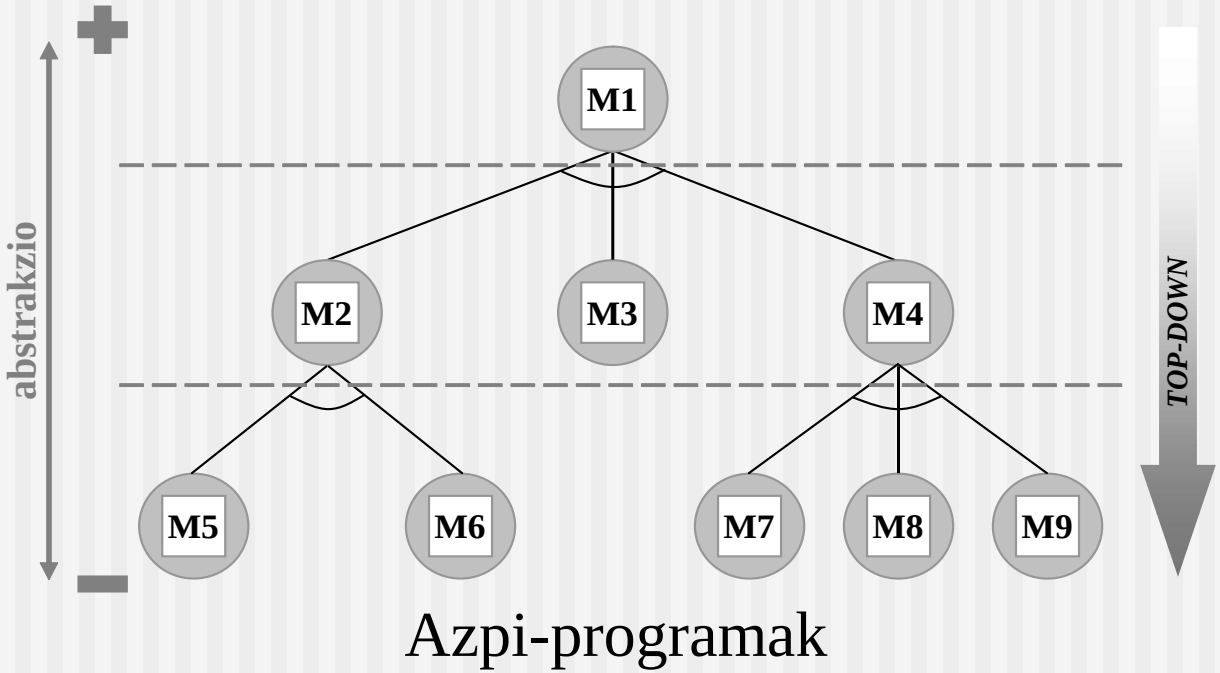


```
idatzi("Sartu bi zenbaki: ");
irakurri(oinarri, exp);
(exp = 0) balitz orduan
        resul←1;
bestela
        resul←oinarri; cont2←1;
(kont2 < exp) egia den bitartean egin
faktore←resul; resul←0; kont1←0;
(kont1 < base) egia den bitartean egin
      resul←resul + faktore;
      kont1←kont1 + 1;
bitartean amaiera
      kont2←kont2 + 1;
bitartean amaiera
baldin amaiera
idatzi(resul);
```

Programazio egituratua

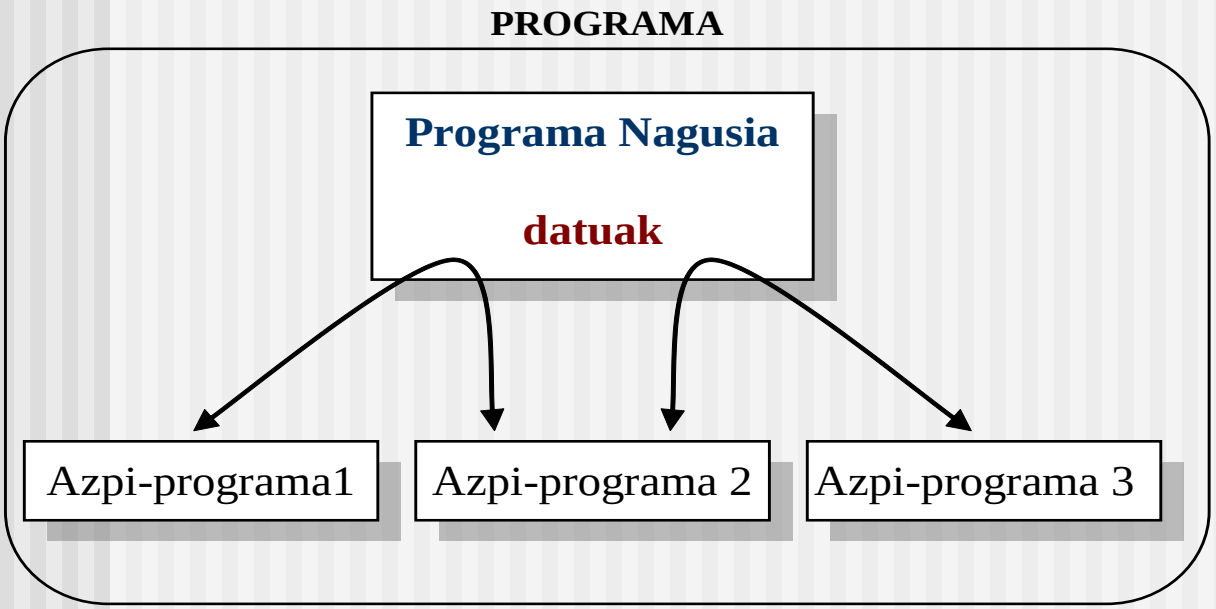
- Lehenengo aldiz, programak **diseinatzen** dira
  - Top-Down diseinua

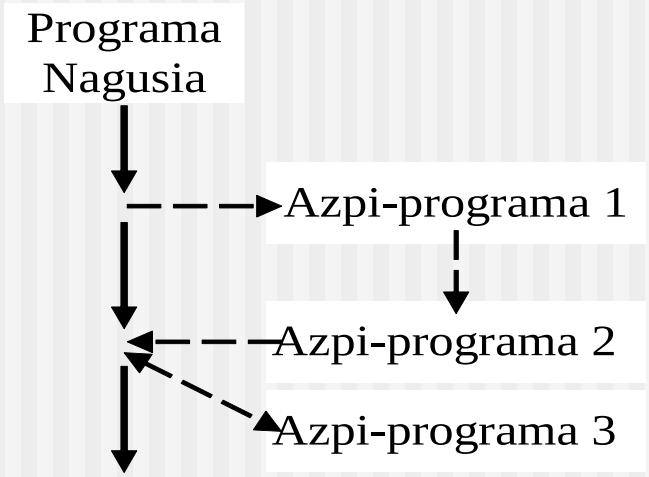
1. Pausua
2. Pausua
...
n. Pausua



Programazio egituratua

- Azpi-programa kontzeptua sortzen da eta noski parametroen trukaketa
  - Aldagai lokalak eta globalak desberdintzen dira





Programazio egituratua

```
Programa Nagusia
integer a, b, d;
eskatu_zenbaki(a);
eskatu_zenbaki(b);
d=berreketa(a, b);
idatzi(d);
```

Ulertzen da!

```
funtzio  berreketa (integer a, integer b) bueltatu integer
 integer c;
c←1
(b>0) egia den bitartean egin
     c←biderkatu(a, c);
     b←b-1;
bitartearen amaiera
bueltatu(c);
```

parametroak
emaitza
aldagai lokala

Aurkibidea

- Gaiaren helburuak
- Software-ren eboluzioa
- **Modularitatea**
- Objektuei Bideratutako Prog.
- Prog. egituratua vs OBP

Softwarraren garapena

- Kontutan izan behar diren oinarriak
  - Abstrakzioa
  - Modularitatea
  - Informazioaren ezkutaketa

Abstrakzioa

- **Funtsezko ezaugarrien** erauzketa eta beste ezaugarrien ezabaketa
  - Gakoa: azken erabiltzailearen lekuan jartzea eta datuak eta eragiketak ikusi bere ikuspuntutik

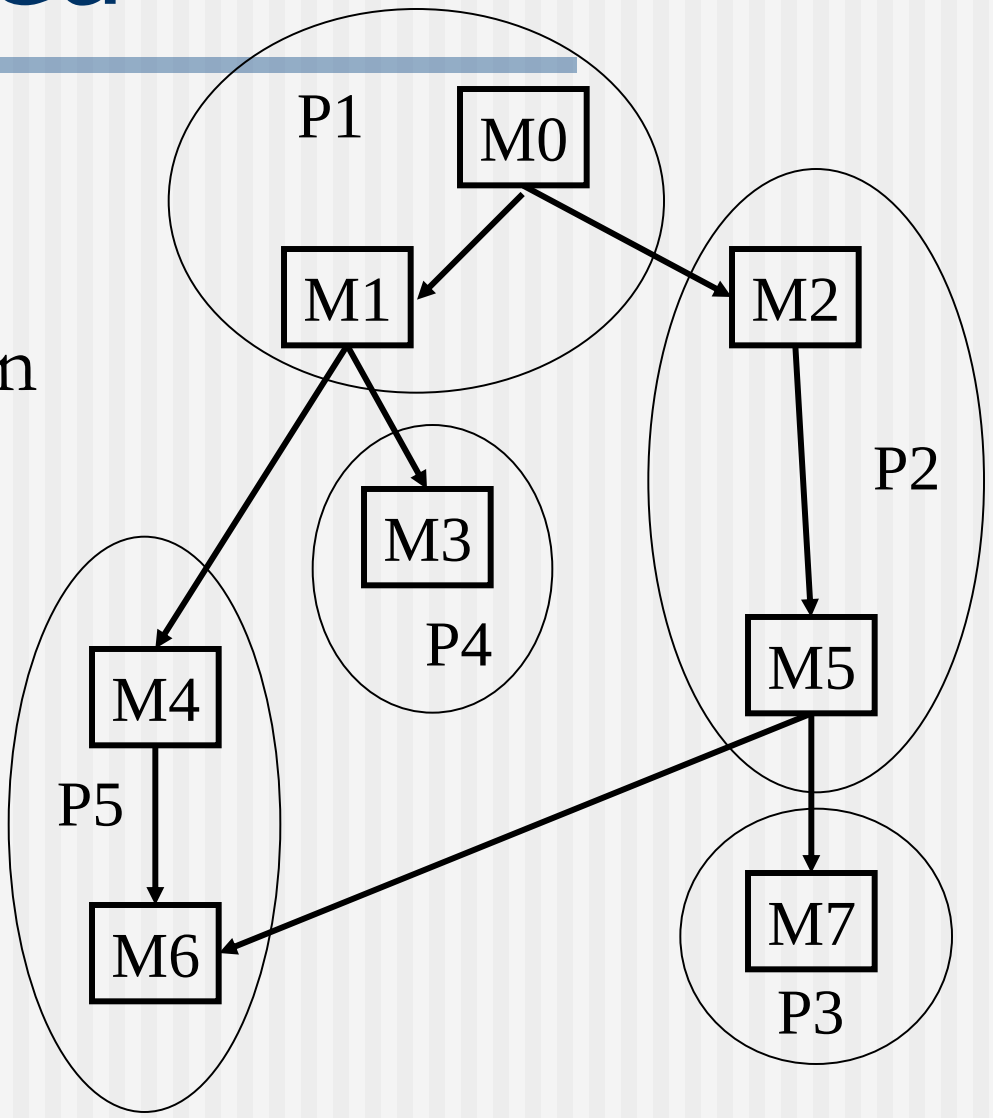
Abstrakzioa

- Adibidez: kotxe baten deskribapena ikuspuntu desberdinetatik

  - Gidari baten ikuspuntutik
  - Mekaniko baten ikuspuntutik
  - Kotxe-zatien diseinatzailearen ikuspuntutik

Modularitatea

- Aplikazio baten egituraketa, elkar-erlazionatuta dauden moduluetan



Informazioaren ezkutatzea

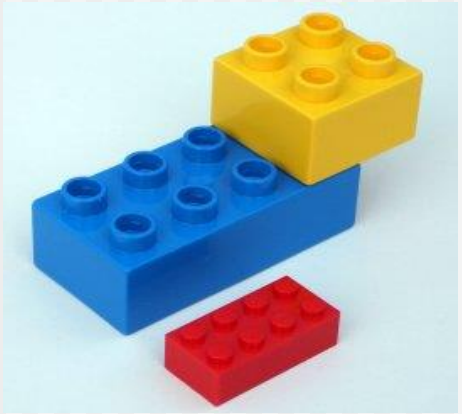
- Modulu bakoitzak, bakarrik behar beharrezkoa erakusten die beste moduluei, detaileak ezkutatuz
  - Barne informazioa, barne kalkuluak egiteko azpi-programak, datu-egiturak...

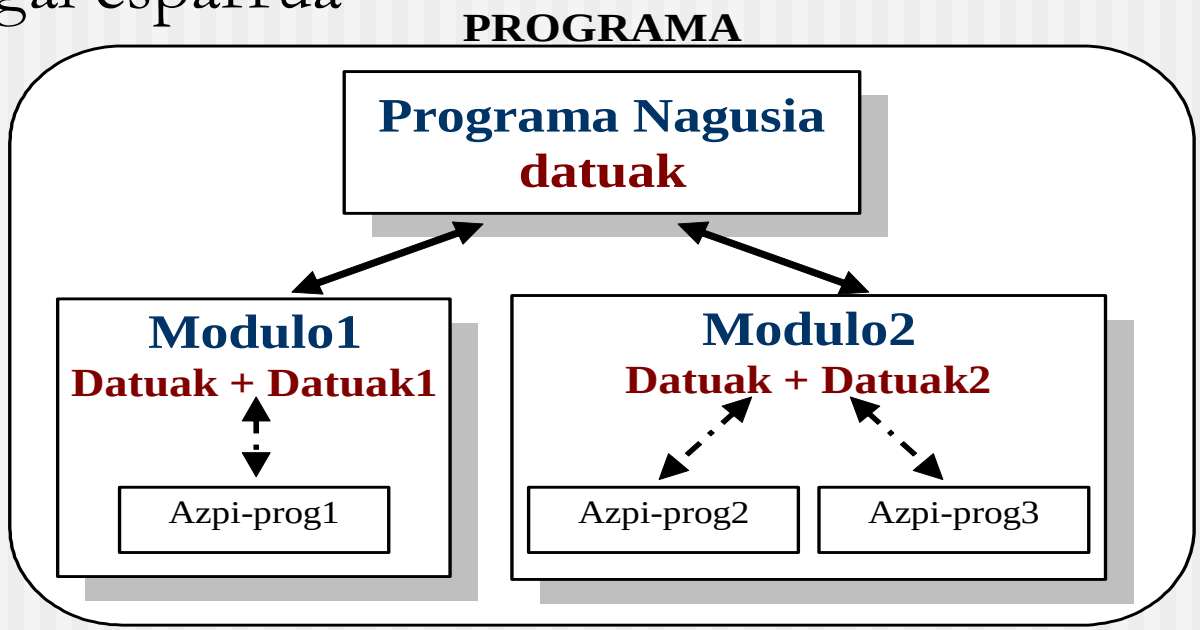
Informazioaren ezkutatzea

- Onura:
  - Ulermena erraztu
  - Aldaketak burutzea erraztu
  - Fidagarritasuna handitu

Programazio modularra

- Antzeko funtzionalitatea duten azpi-programak moduluetan multzokatzen dira
  - Modulu bakoitzak bere datuak izango ditu
  - Berezko aldagai esparrua





Programazio modularra

- Aplikazio handi bat = Σ moduluak
  - Diseinu ona → moduluen zatiketa egokia
- Moduluak izan behar dira:
  - Autonomoak eta autoedukiak
  - Elkar-konektatuta
  - Berrerabilgarriak

Modularitatearen abantailak

- Deskonposaketa modularra = Banandu eta irabazi
  - Modulu independente eta elkar-komunikatuak
- Ulergarritasuna
  - Sistema zatika ulertzeko erraztasuna eta ondorioz, bere osotasunean ere bai
- Akatsak eta aldaketak ez dute aplikazio osoarengan eraginik

Aurkibidea

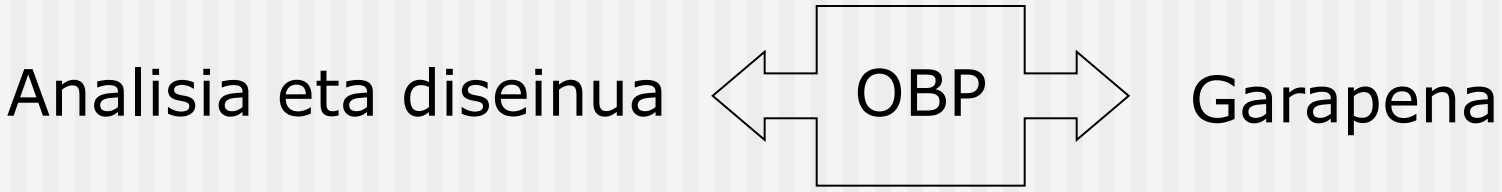
- Gaiaren helburuak
- Software-ren eboluzioa
- Modularitatea
- **Objektuei Bideratutako Prog.**
- Prog. egituratua vs OBP

OBP: historia apur bat

- 1967: simula67, OB lehenengo lengoaia
- 1970: Smalltalk, OB ingurune grafikoarekin
- 70. hamarkada: C, goi mailako lengoaia, behe mailako ezaugarriekin
- 80. hamarkada: C++, C-ri OB paradigmaren ahalmenaz hornitu
- 1991: Java, elektrodomestikoentzat softwarea garatzeko sortzen da.

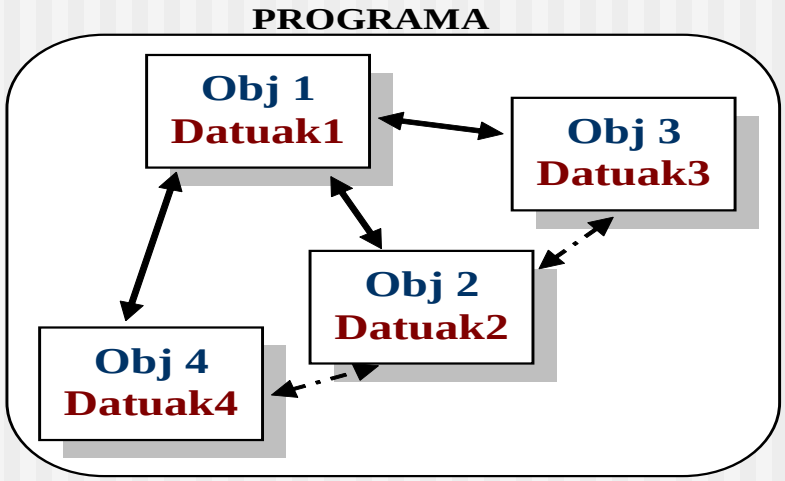
OBP

- Aplikazioen diseinua burutzeko programazio metodologia bat da
- Programazio egituratua da (modularra)
  - Prozesuetan baino, errealitatea modelatzean jartzen du arreta



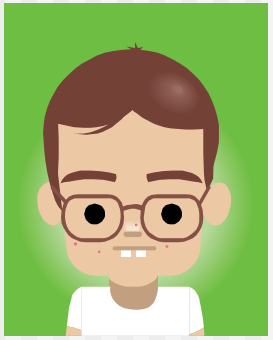
Objektuei Bideratutako ikuspuntua

- OB programazioa, programak objektuen ikuspuntutik garatzen dituen programazio paradigma bat da
  - Objektuak elkar komunikatzen dira, mezuak euren artean pasatuz



Objetuak eta klaseak

- Objektua = berezko nortasuna duen elementu erreala
  - Egoera batekin (atributuak) eta jokaera (metodoak) zehatzarekin
- Klase = objektu zehatzen abstrakzioa



AH!! klasea objetuarekiko litzateke, aldagai-mota aldagaiarekiko dena

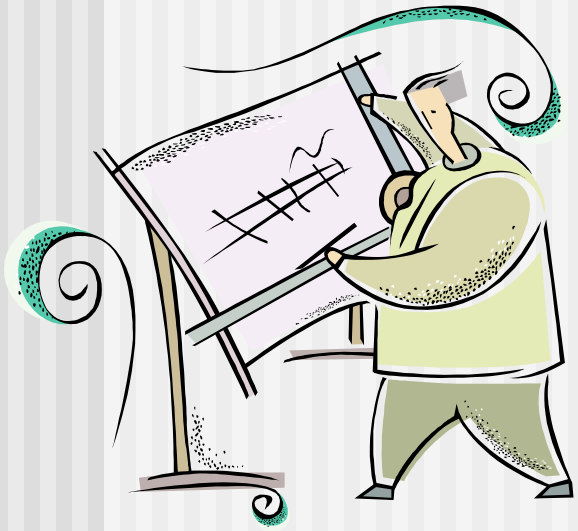
(integer-mota zenb-aldagaia)

Erne!! galdera

- Zein da *klase* eta *erregistro* arteko desberdintasuna?

Kotxe Klasea

- Kotxe guztien ezaugarri (atributu) eta funtzionalitateen jokaera (metodoak) *abstrakzioa*



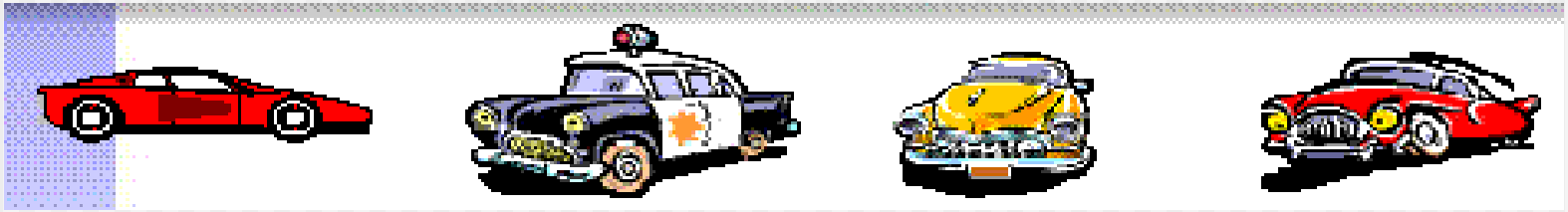
kolore String
modeloa String
matrikula String
abiaduraMax real
...

Emaitzaren mota

void abiatu()
void gelditu()
real azeleratu()
real frenatu()
void gainditu(Kotxe pAurrekoa)
...

Kotxe klaseko objektuak

Objektuak



objektu1
kolore gorria
modeloa deportiboa
matrikula 1234 ABC
abiaduraMax 300

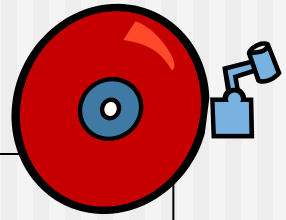
objektu2
kolore urdina
modeloa polizia
matrikula 0000 PM
abiaduraMax 250

objektu3
kolore horia
modeloa dotorea
matrikula 9876 DEF
abiaduraMax 150

objektu4
kolore gorria
modeloa erabilgarri
matrikula 9235 FCB
abiaduraMax 200

Objektuetan pentsatzen

- Klase bat diseinatzean objektuetan pentsatzea komeni da
  - Objektuen ezaugarriak edo **ezagutza**
  - Objektuak **egiten** dituen gauzak



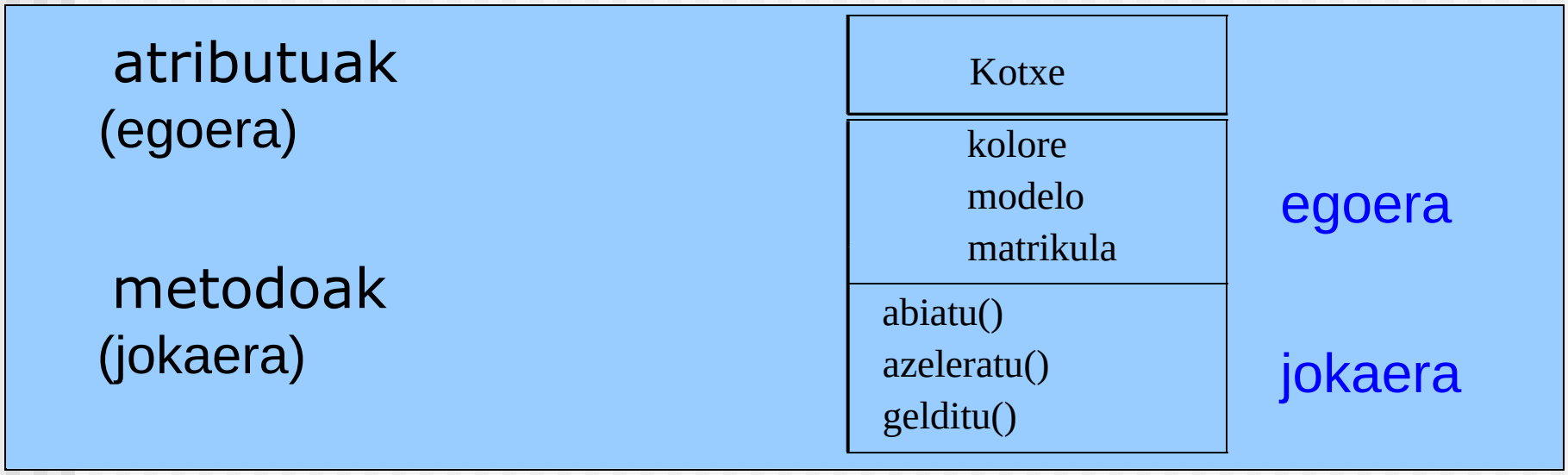
ErosketenKarroa
karroarenEdukia
karroanGehitu() karrotikKendu() kutxatikPasa() produktuakKontatu()

Botoia
etiketarenIrudia ertzenKolorea
kolorezAldatu() etiketaAldatu() sakatu() askatu()

Alarma
alarmarenEgoera alarmarenOrdua
orduanJarri() lortuOrdua() aktibatutaAlDago?() piztu()

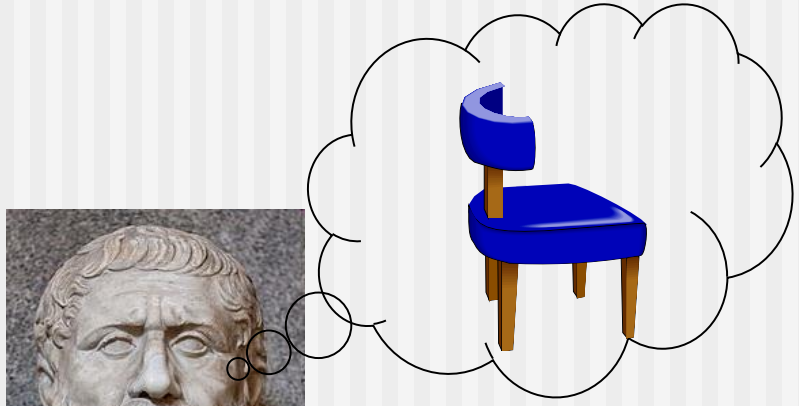
Ezagutu eta egin

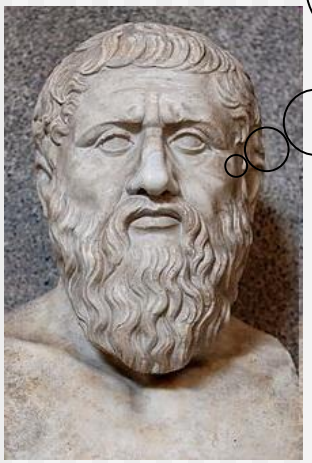
- Atributuak edo instantzia aldagaiak
  - Objektu batek bere buruaz dakien guztia
- Metodoak
  - Objektu batek egin dezakeena (jokaera)



klase/objektu arteko desberdintasunak

- Klase bat EZ DA objektu bat, baina objektuak eraikitzeko erabiltzen da
  - Nemoteknia → Gauza bera gertatzen da aldagai eta aldagaiaren mota artean





Platon-en ideien mundua bezala: ideia kontzeptuala (abstraktua, idilikoa) vs. objektu erreala (zehatza)

Klaseen sorrera

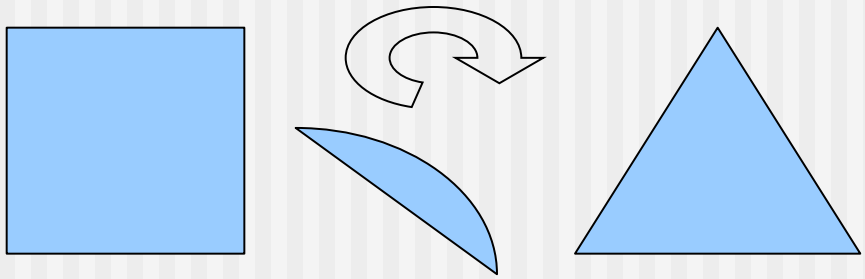
1) Klase horretako objektuak identifikatzea
   - Mundu erreala imitatu behar dute
   - Ez dute zertan objektu fisikoak izan behar
2) Atributuak definitu
3) Jokaera (funtzionalitatea) definitu
   - Objektuak eraikitzeko eragiketak ahaztu gabe

Aurkibidea

- Gaiaren helburuak
- Software-ren eboluzioa
- Modularitatea
- Objektuei Bideratutako Prog.
- **Prog. egituratua vs OBP**

Adibidea: espezifikazioa

- Demagun leiho grafiko batean hainbat forma ditugula: lauki bat, borobil bat eta hiruki bat. Erabiltzaileak formaren batetan sakatzean, forma horrek buelta emango du (buelta oso bat, 360º buelta, erlojuaren orratzen zentzuan). Eta forma horri esleitutako AIF formatuzko soinu berezi bat joko du



Egituratua vs OB

Lady Byron

```
bueltaEman (formaIdent) {
   //360°ko buelta eman
}

soinuaJo(formaIdent) {
  // formaIdent erabili
  // dagokion AIF
  // fitxategia bilatzeko
}
```

Alan Kay

Laukia

```
bueltaEman(){
  //Laukiak buelt
}

soinuaJo(){
 // Laukiaren AI
}
```

Borobila

```
bueltaEman(){
  // Borobilak bu
  // emateko kode
}

soinuaJo(){
  //Borobilaren A
}
```

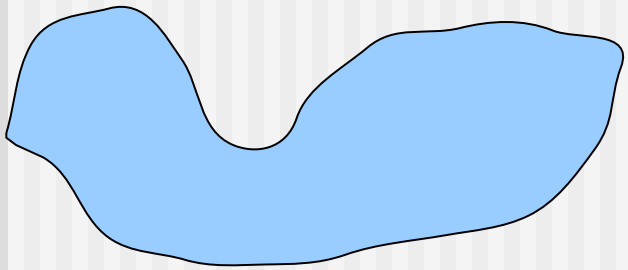
Hirukia

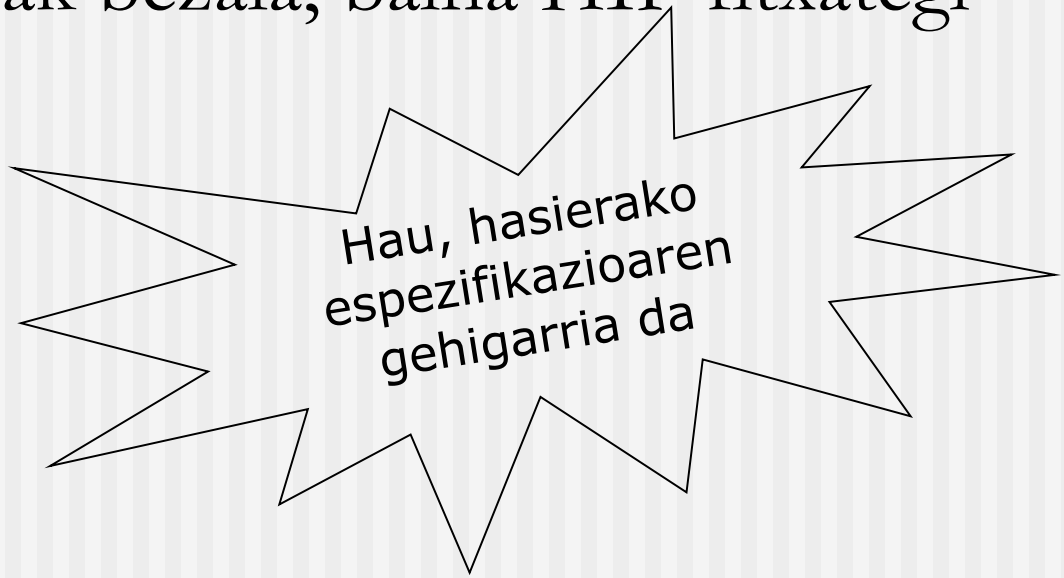
```
bueltaEman(){
  // hirukiak buelta
  //emateko kodea
}

soinuaJo(){
  // Hirukiaren AIF jo
}
```

Espezifikazio aldaketa

- Pantailan forma berri bat egongo da, besteen parean: amoeba bat. Honela, erabiltzaileak amoebaren gainean sakatzean, honek buelta osoa emango du besteak bezala, baina HIF fitxategi bat joko du





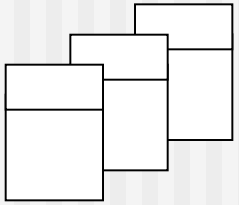
Egituratua vs OB

Lady Byron

```
BueltaEman (formaIdent) {
   // 360°ko buelta eman
}

soinuaJo(formaIdent) {
  //  forma ez bada amoeba bat
  //  formaIdent erabili AIF
  //  fitxategia topatzeko
  //  eta jotzen hasteko
  // bestela
  //   amoebaren .hif jo
}
```

Fidagarritasuna?
Bererapilpena?
Ulergarritasuna?



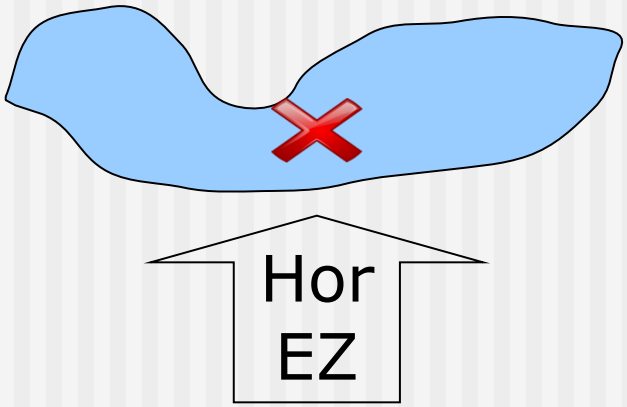
Amoeba

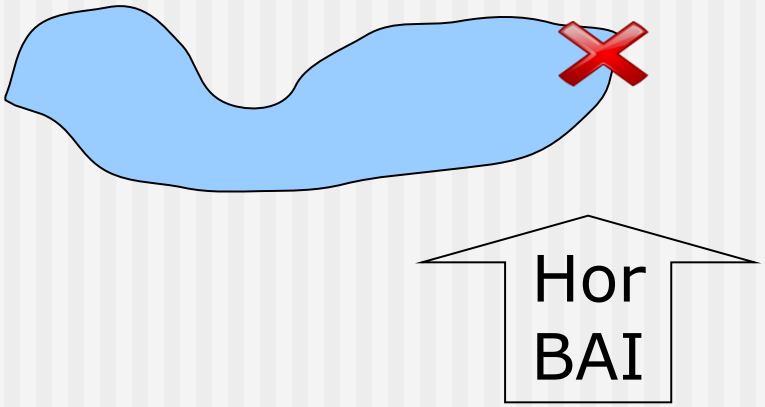
bueltaEman(){
//amoebak buelta emateko kodea
}
 	
soinuaJo(){
// .hif fitxategi bat jotzeko kodea
}

Alan Kay

Ui! A ze nolako despistea...

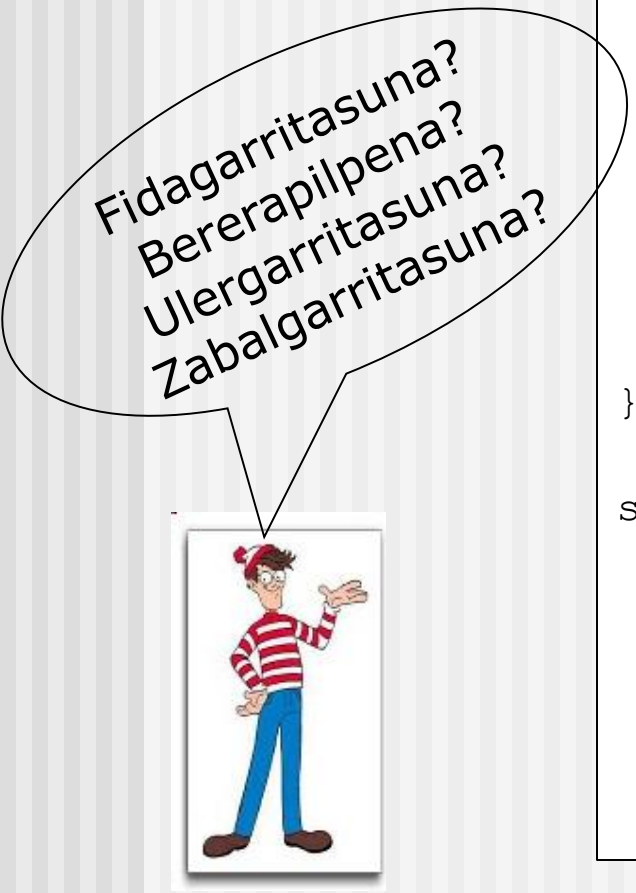
- Erabiltzaileari ahaztu zaio esatea amoebak ez duela erdiko puntua erabiltzen buelta emateko erreferentziazko puntu gisa. Beraz, ez Lady Byron ezta Mr. Kay ez dute horrela programatu





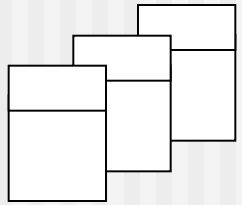
Egituratua vs OB

Lady Byron




```
bueltaEman (formaIden, xPt,
yPt) {
 // formaIdent ez bada amoeba
 // 360°ko buelta eman
 // erdiarekiko
 // bestela
 // erabili xPt,yPt buelta
 // emateko erreferentziazko
 // puntu gisa
}

soinuaJo(numForma) {
  //numForma ez bada amoeba
  //  erabili numForma AIF
  //  fitxategia bilatzeko
  //  eta egikaritzeko
  // bestela
  //   jo .hif }
```



Amoeba

int puntoX
int puntoY
 	
bueltaEman(){
// amoeba bati buelta emateko
// kodea X eta Y erabilita
}
 	
soinuaJo(){
// .hif soinua jotzeko
//kodea
}

Alan Kay

Herentzia kontzeptua

Laukia	Borobila	Hirukia	Amoeba
bueltaEman() soinuaJo()	bueltaEman() soinuaJo()	bueltaEman() soinuaJo()	puntuX puntuY ------- bueltaEman() soinuaJo()

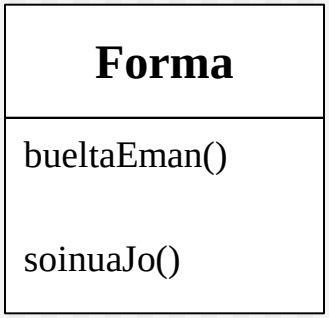
1 Zer daukate klase hauek amankomunean?

Forma
bueltaEman() soinuaJo()

2 abstrakzioa eta orokortasuna

Herentzia kontzeptua

superklase

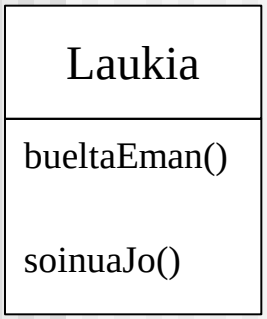


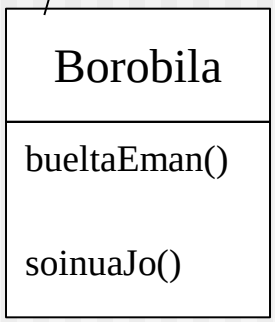
3

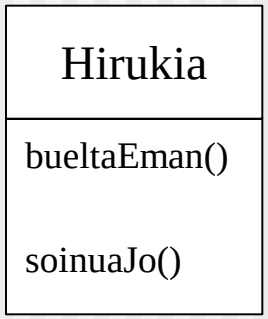
Bildu gure 4 klase, beste klase batean, HERENTZIA erlazio bat sortuz

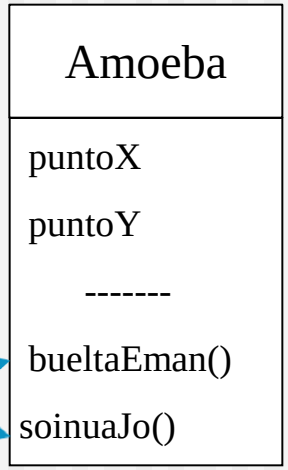
Fidagarritasuna!!!
Bererapilpena!!!
Ulergarritasuna!!!
Zabalgarritasuna!!!

azpiklaseak

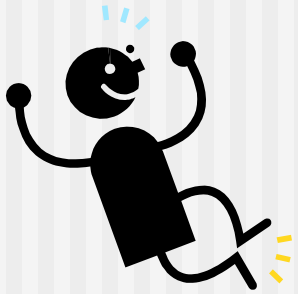








overriding




Galderarik ?