



# Programazioa

# Java egituratua

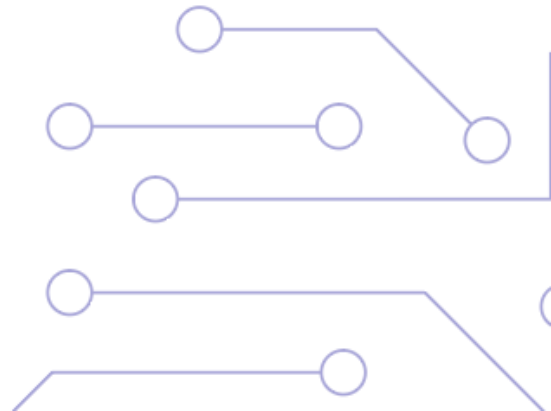
# Java lengoaia

- Objektuetara zuzendutako erabilera orokorreko programazio lengoaia bat da
- Plataformatik independentea
- Lengoaia erabilienetako bat da
  - <https://www.stackscale.com/es/blog/lenguajes-programacion-mas-populares/>
- Dokumentazioa:
  - <https://www.w3schools.com/java/default.asp>
  - <https://docs.oracle.com/en/java/javase/20/>



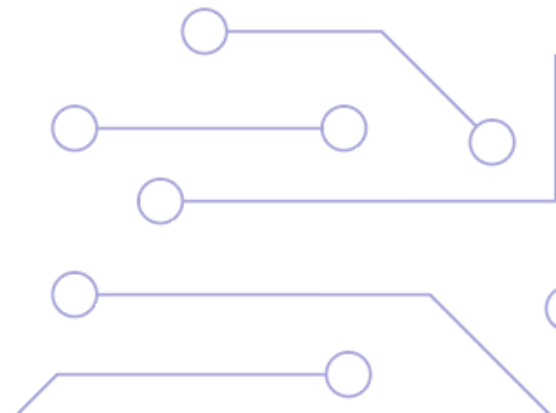
# Klaseak, paketeak eta metodoak

- Klase bat Javako oinarritzko fitxategi bat da
  - Objektu bat definitu dezake
  - Programazio egituratuan ere erabili daiteke
  - Izena letra maiuskula batez hasten da
    - Ariketa01.java
- Pakete batean antzeko helburu edota erabilera duten klaseak gordetzen dira
  - Izena letra minuskulaz idazten da
  - Hitz bat baino gehiago lotu daitezke puntu batez
    - zubiri.wag31.nagusia



# Klaseak, paketeak eta metodoak

- Metodoak klase baten barruan definitzen diren programak/funtzioak dira
  - Klase baten portaera definitzen dute
  - Izena minuskulaz hasten da
    - `public void batuSoldata()`
  - Aplikazio baten metodo nagusiak *Main* izena dauka eta hau izango da aplikazioa hasieratzen denean exekutatu den lehen programa





KONTUZ

Javak maiuskulak eta  
minuskulak  
desberdintzen ditu

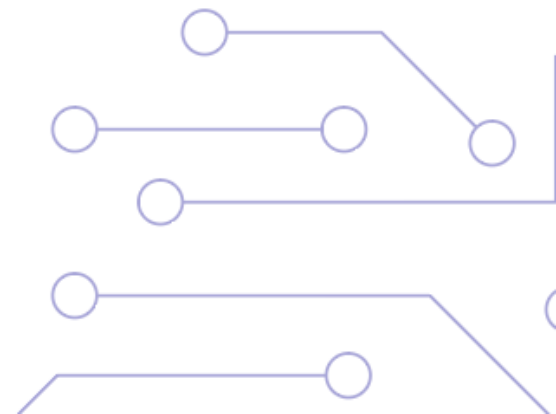
# Klaseak, paketeak eta metodoak

```
package zubiri.wag31.nagusia;
/**
 * @author Unai
 */
public class Ariketa01 {
    public static void main(String[] args) {
        System.out.println("Kaixo mundua!");
    }
}
```



# Java Development Kit (JDK)

- Javan programatzeko beharrezkoak diren baliabideak eskaintzen ditu:
  - Konpiladorea
  - Exekutatzeko makina birtuala (JRE)
  - Lanerako liburutegiak
  - ...
- Sistematik dependentea da, hau da, sistema bakoitzak berezko bertsioa du



# Java Development Kit (JDK)

JDK 21 **JDK 17** GraalVM for JDK 21 GraalVM for JDK 17

## JDK Development Kit 17.0.8 downloads

JDK 17 binaries are free to use in production and free to redistribute, at no cost, under the [Oracle No-Fee Terms and Conditions \(NFTC\)](#).

JDK 17 will receive updates under the NFTC, until September 2024. Subsequent JDK 17 updates will be licensed under the [Java SE OTN License \(OTN\)](#) and production use of limited free grants of the OTN license will require a fee.

Linux macOS **Windows**

Product/file description	File size	Download
x64 Compressed Archive	172.38 MB	<a href="https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip">https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip</a> ( sha256)
<b>x64 Installer</b>	153.48 MB	<a href="https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe">https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe</a> ( sha256)
x64 MSI Installer	152.27 MB	<a href="https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi">https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi</a> ( sha256)




# IntelliJ IDEA

- Java eta beste zenbait programazio lengoaiatan programatzeko ordaindu beharreko garapen ingurunea (IDE, *Integrated Development Environment*) da
  - Komunitateak mantentzen duen doako bertsioa dauka
  - Git biltegi bat sortzea ahalbidetzen du proiektuan egindako aldaketak kontrolatzeko
  - Adimen artifizialean oinarritutako laguntzaile bat du kodea aztertu eta hobekuntzak proposatzeko



# IntelliJ IDEA

- Beraien web orritik komunitateko bertsioa lor daiteke




Developer Tools Team Tools Education Solutions Support Store

IntelliJ IDEA

New UI What's New Features ▼ Resources Pricing Download

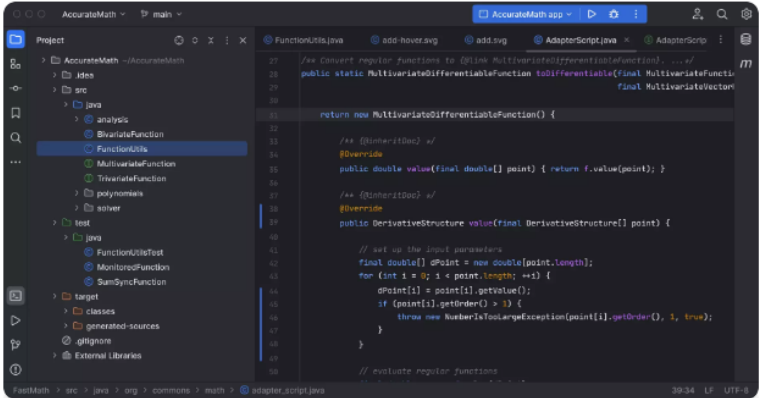
Windows macOS Linux



**IntelliJ IDEA Ultimate**  
The Leading Java and Kotlin IDE

Download .exe ▼

Free 30-day trial



Version: 2023.2.1  
Build: 232.9559.62  
23 August 2023

[System requirements](#)  
[Installation instructions](#)

[Other versions](#)  
[Third-party software](#)

# IntelliJ IDEA

## Other Versions

Version 2023.2

2023.2.1



### IntelliJ IDEA Ultimate

[2023.2.1 - Linux aarch64 \(tar.gz\)](#)

[2023.2.1 - Linux x86\\_64 \(tar.gz\)](#)

[2023.2.1 - Windows ARM64 \(exe\)](#)

[2023.2.1 - Windows x64 \(exe\)](#)

[2023.2.1 - Windows x64 ZIP Archive \(zip\)](#)

[2023.2.1 - macOS \(dmg\)](#)

[2023.2.1 - macOS Apple Silicon \(dmg\)](#)

### IntelliJ IDEA Community Edition

[2023.2.1 - Linux aarch64 \(tar.gz\)](#)

[2023.2.1 - Linux x86\\_64 \(tar.gz\)](#)

[2023.2.1 - Sources Archive \(zip\)](#)

[2023.2.1 - Windows ARM64 \(exe\)](#)

[2023.2.1 - Windows x64 \(exe\)](#)

[2023.2.1 - Windows x64 ZIP Archive \(zip\)](#)

[2023.2.1 - macOS \(dmg\)](#)

[2023.2.1 - macOS Apple Silicon \(dmg\)](#)

Version: 2023.2.1 ([Release notes](#))

Build: 232.9559.62

Released: 23 August 2023

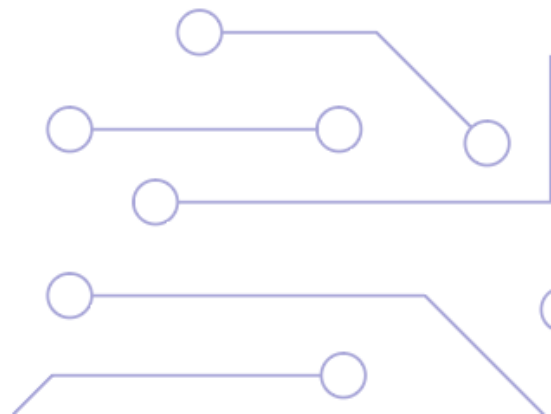
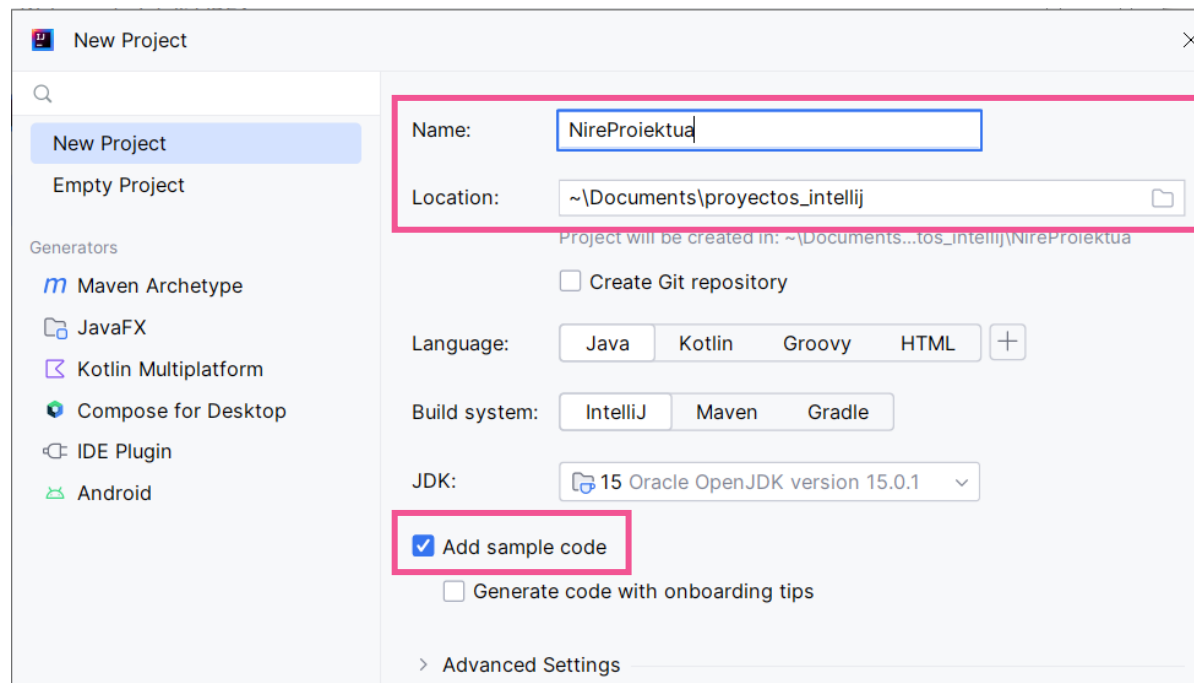
Major version: 2023.2

Released: 26 July 2023

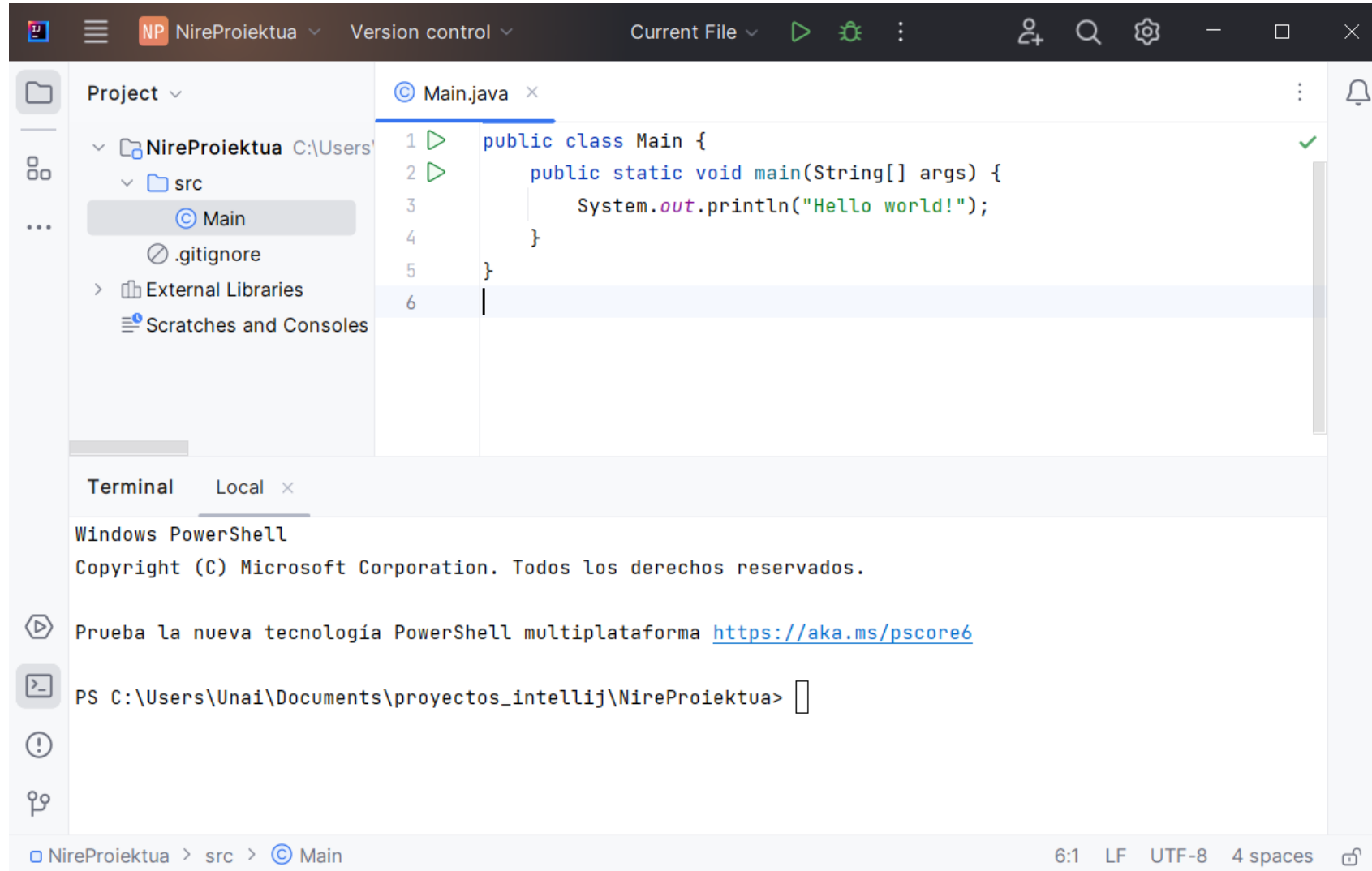
[IntelliJ IDEA Ultimate third-party software](#)  
[IntelliJ IDEA Community Edition third-party software](#)

# IntelliJ IDEA

- Proiektu berri bat sortzean honi izen bat jarri behar zaio eta balio lehenetsiak uztearekin nahikoa da
  - Beharbada *Add sample code* aukera aktibatzea egokia izan daiteke
  - Ondo adierazi proiektuaren kokapena

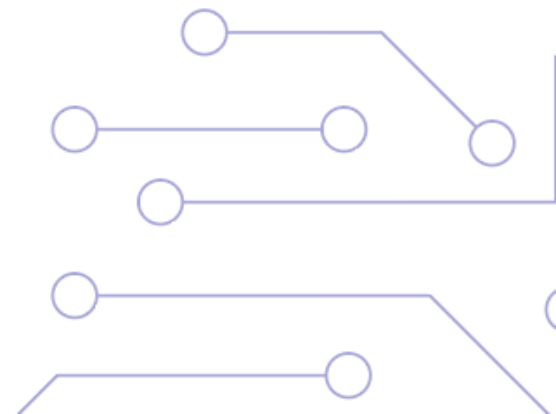


# IntelliJ IDEA



# Aginduak eta iruzkinak

- Javan agindu guztiek puntu koma ikur baten bidez (";") bukatu behar dute
- Metodo eta egituren hasiera eta bukaera adierazteko giltzak erabiltzen dira ("{}")
- Lerro bakarreko komentarioak bi barren ondoren ("//") adierazten dira
- Lerro bat baino gehiagoko komentarioak "/\*" eta "\*/" sinboloen artean adierazten dira





GOGORATU:

Lerroen indentazio argi bat  
egitea oso komenigarria da

# Aginduak eta iruzkinak

- Javadoc Java iturburu-kodean oinarrituta dagoen HTML formatuko dokumentazioa sortzeko funtzionalitatea da
- Javaren klaseak dokumentatzeko industriaren estandarra da
- Javadoc-ekin dokumentazioa sortzeko, HTMLren etiketak edo hitz erreserbatu batzuk erabili behar dira aurretik "@" karakterea gehituta
- Etiketa hauek klase edo metodo bakoitzaren hasieran idazten dira, deskribatu nahi den objektuaren arabera, "/\* \*/" rekin hasi eta "\*/" - rekin amaitzen den iruzkin baten bidez.
  - <https://es.wikipedia.org/wiki/Javadoc>
  - <https://docs.oracle.com/en/java/javase/20/docs/specs/javadoc/doc-comment-spec.html>



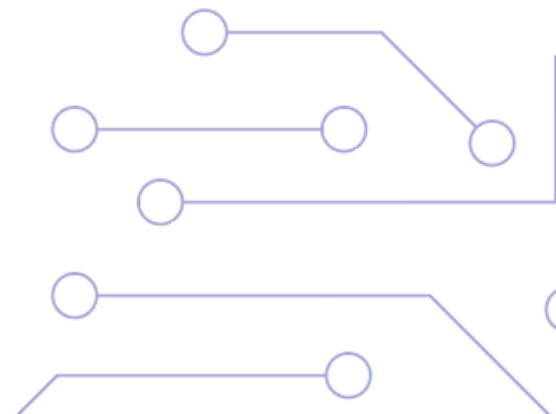
# Datu motak

Izena	Definizioa	Adibidea
byte	Zenbaki osoak (1 byte)	-23, 116
int	Zenbaki osoak (4 byte)	32, -1274
long	Zenbaki osoak (8 byte)	45, -7454645634
float	Zenbaki hamartarrak (4 byte)	3.5, -9.8675
double	Zenbaki hamartarrak (8 byte)	654.893, -3445.4234464
boolean	Bi balio: egia eta gezurra	true, false, 0, 1
char	Karakterek	'a', ':'
String	Karaktere kateak	"Kaixo", "Hau Java da"

# Aldagaien eta konstanteen definizioa

- Lehenik aldagaiaren mota adierazten da eta ondoren izena
- Erabili izen esanguratsuak!
- Nahi bada hasierako balio bat ezarri daiteke

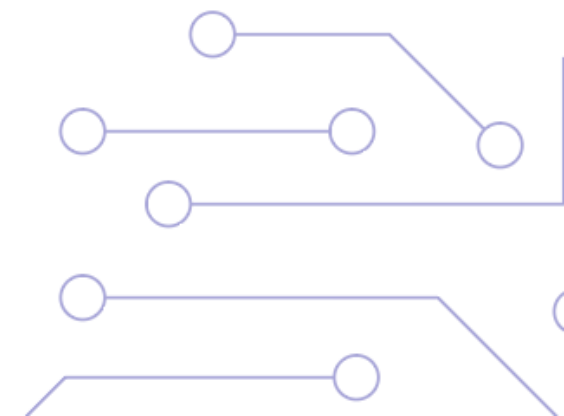
```
int zenb;  
boolean aurkitua = false;  
String katea = "Kaixo";  
float PI = 3.14159
```



# Esleipena

- "=" karakterea erabiltzen da
- Ezkerrean helburu aldagai edo konstantea jartzen da eta eskuinean gordeko den balioa adierazten da

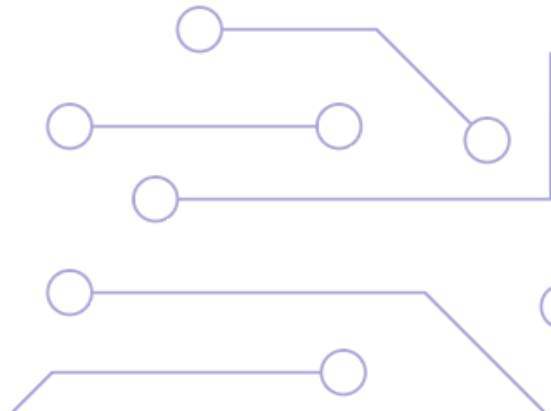
```
int zenb = 5;  
zenb = zenb / 4;  
String katea = "Kaixo";  
katea = katea + "denoi"
```



# Datuen irteera

- Agindurik sinpleena **System.out.println** da, lerro berri batean parentesien artean adierazten zaiona idazten du
- Bere aldaera den **System.out.print** lerro berdinean idazten du
- Zenbakiak, karaktere kateak eta aldagaiak adierazi daitezke idazteko
  - Aldagaien kasuan beraien edukia inprimatzen da

```
int zenb = 5;  
System.out.println("Hau testu bat da");  
System.out.println(17);  
System.out.println(zenb);
```



# Datuen irteera: Karaktere bereziak

- Badaude zuzenean komilen artean inprimatu ezin diren karaktere batzuk: ", ', \...
- Badaude ere karaktere ez diren baina batzuetan esplizituki adierazi behar diren beste egitura batzuk: lerro jauzia, tabuladorea...
- Guzti hauek adierazteko aurretik atzeranzko barra erabili behar da

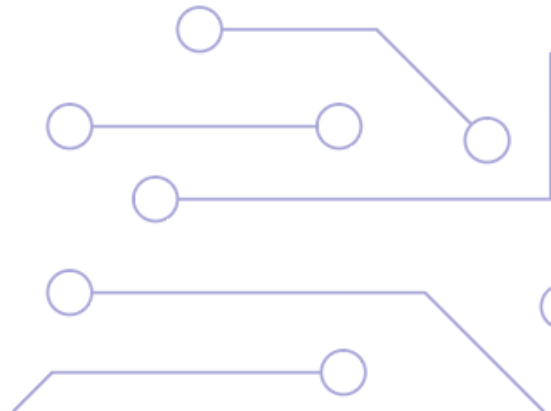
Karakterea/ Egitura	Nola adierazi	Karakterea/Egitura	Nola adierazi
"	\"	Lerro jauzia	\n
'	\'	Tabuladorea	\t
\	\\	Atzera ezabatzea	\b

# Datuen sarrera

- Errazena **Scanner** objektu bat inportatu eta hau erabiltzea da
  - Inportazioaren agindua objektua sortzen denean automatikoki gehitzen da
- Ondoren scanner objektua definitu eta sortu behar da
- Garrantzitsua: Bukaeran scanner objektua itxi behar da

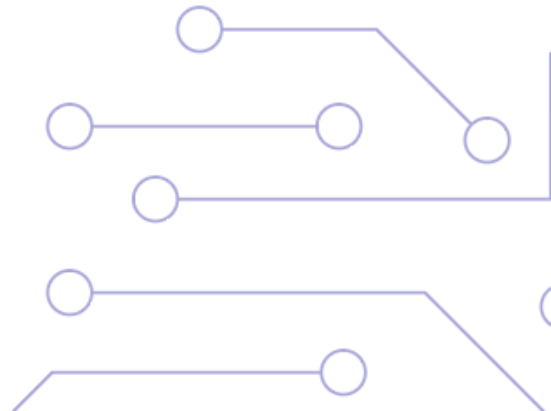
```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ...
        sc.close();
    }
}
```



# Datuen sarrera

- Sortu berri den objektua datu mota desberdinak irakurtzeko erabili daiteke
  - **int**: `sc.nextInt();`
  - **long**: `sc.nextLong();`
  - **float**: `sc.nextFloat();`
  - **double**: `sc.nextDouble();`
  - **char**: Ez dauka
  - **String**: `sc.nextLine();`



# Eragile aritmetikoak

- Matematikan jarraitzen den eragiketen lehentasuna mantentzen da
  - berreketa/erroketa -> biderketa/zatiketa -> batuketa/kenketa


Eragilea	Esanahia	Adibidea
+	Batuketa String motako balioak lotzeko ere erabili daiteke	$a + b$ String esaldi = "Kaixo" + "Unai"
-	Kenketa	$c - 5$
*	Biderketa	$8 * a$
/	Zatiketa	$b / c$
%	Hondarra/Modulua	$b \% c$
++	Bateko gehikuntza	$i++$
--	Bateko gutxikuntza	$i--$





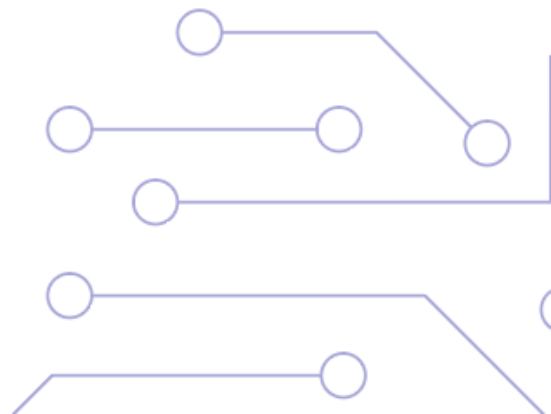
# Nire lehen programa

```
public static void main(String[] args) {  
    Scanner sc = new Scanner(System.in);  
    System.out.println("Sartu zenbaki bat: ");  
    int zenb1 = sc.nextInt();  
    System.out.println("Sartu beste zenbaki bat: ");  
    int zenb2 = sc.nextInt();  
    int bat = zenb1 + zenb2;  
    System.out.println("Baturaren emaitza hau da: " + bat);  
    sc.close();  
}
```



# Barne funtzioak

- Javak liburutegi asko ditu dagoeneko sortuta dituen funtzioak erabiltzeko
- Nahikoa da funtzioak definituta dauden paketeak inportatzearekin hauek erabili ahal izateko
  - Normalean aginduak idazten hastean hauek automatikoki gehitzen dira
  - Beste batzuetan garapen inguruneak atzetik egiten duen pauso bat da



# Barne funtzioak: matematikoak

Funtzioa	Esanahia	Adibidea	Emaitza
Math.PI	Pi-ren balioa	<code>double pi = Math.PI; System.out.println(pi);</code>	3.14159265358
Math.floor(double x)	x-en balioa beheraka borobiltzen du	<code>double a = Math.floor(5.67); System.out.println(a);</code>	5.0
Math.ceil(double x)	x-en balioa goraka borobiltzen du	<code>double z = Math.ceil(5.34); System.out.println(z);</code>	6.0
Math.round(float x)	x-en balioa borobiltzen du	<code>int zenb = Math.round(5.67); System.out.println(zenb);</code>	6
Math.pow(double x, double y)	x ber y eragiketa egiten du	<code>double y = Math.pow(3.0, 2.0); System.out.println(y);</code>	9.0
Math.random()	[0, 1) tarteko ausazko zenbaki bat itzultzen du	<code>double k = Math.random(); System.out.println(k);</code>	0.8730583

[https://www.w3schools.com/java/java\\_ref\\_math.asp](https://www.w3schools.com/java/java_ref_math.asp)

# Barne funtzioak: testuak

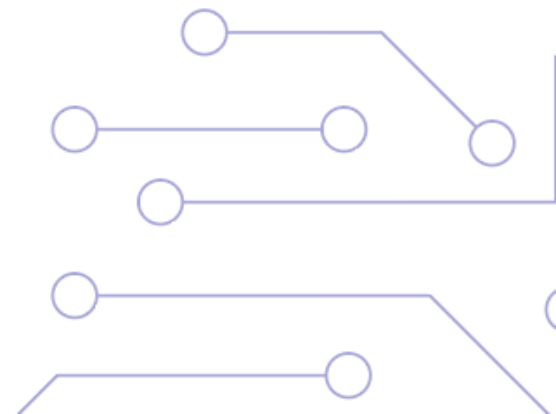
Funtzioa	Esanahia	Adibidea	Emaitza
s.charAt(int index)	Index posizioan dagoen karakterea itzultzen du	String s = "Kaixo"; char c = s.charAt(1); System.out.println(c);	'a'
s.indexOf(String s2)	s katean s2-ren lehen agerpenaren posizioa itzultzen du	String s = "Kaixo"; int i = s.indexOf("o"); System.out.println(i);	4
s.isEmpty()	Egia itzultzen du hutsik badago eta gezurra kontrako kasuan	String s = "Kaixo"; boolean b = s.isEmpty(); System.out.println(b);	false
s.length()	s katearen luzera itzultzen du	String s = "Kaixo"; int i = s.length(); System.out.println(i);	5

# Barne funtzioak: testuak

Funtzioa	Esanahia	Adibidea	Emaitza
s.substring(int a, int b)	a eta b posizioen arteko kate zatia itzultzen du	String s = "Kaixo"; String s2 = s.substring(2, 4); System.out.println(s2);	"ix"
s.contains(String s2)	s2 katea s katean barruan dagoen egiaztatzen du	String s = "Kaixo"; boolean b = s.contains("ai"); System.out.println(b);	true
s1.equals(String s2)	Egiazkoa itzultzen du bi kateak berdinak badira eta faltsua kontrako kasuan	String s1 = "etxea"; String s2 = "Etxea"; System.out.println(s1.equals(s2));	false
s.split(String regex)	s katea regex katearen arabera zatitzen du eta zatiak array batean itzultzen ditu	String s = "kaixo:ikasleak" String[] array = s.split(":");	{"kaixo", "ikasleak"}

# Moten arteko transformazioak

- *Casting* ere deitzen zaio
- Batzuetan datu mota batetik beste baterako aldaketa beharrezkoa dela ikusiko da
  - Adibidez, bi zenbaki oso irakurri eta zatiketaren emaitza hamartarra lortu nahi bada
  - Adibidez, testu motako zenbaki bat irakurri eta zenbaki bezala erabili nahi bada
- Bi casting mota daude:
  - Mota zabaltzeko: char -> int -> long -> float -> double
  - Mota estutzeko: double -> float -> long -> int -> char



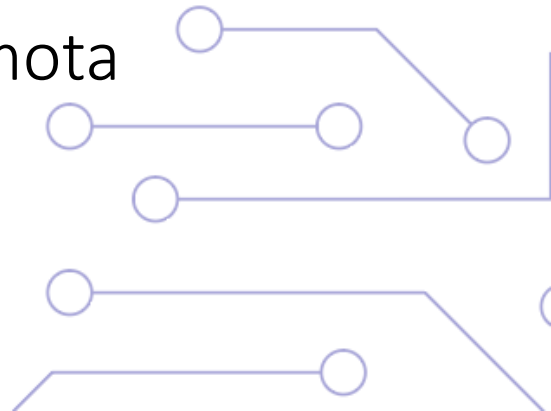
# Moten arteko transformazioak

- Zabaltzeko transformazioa automatikoki egiten da mota handiagoko aldagai batean mota txikiagoko balio bat gordetzen bada

```
int nireOsoa = 9;  
double nireHamartarra = nireOsoa;  
  
System.out.println(nireOsoa); // Irteera 9  
System.out.println(nireHamartarra); // Irteera 9.0
```

- Hau ere balio bat Scanner objektu baten bidez irakurtzean mota handiago batean gordetzean betetzen da

```
float nireZenb = sc.nextInt();
```

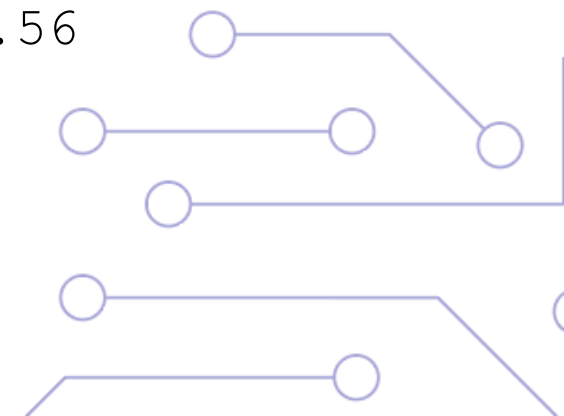


# Moten arteko transformazioak

- Estutzeko transformazioa egiteko bi modu erabili daitezke:
  - Mota aldatu nahi den aldagaiaren aurretik helburu mota parentesi artean adieraziz
    - *Math.round()* funtzioak *double* motatik *int* motara aldaketa ere egiten du

```
double nireHamartarra = 4.56;  
int nireOsoa = (int)nireHamartarra;
```

```
System.out.println(nireHamartarra); // Irteera 4.56  
System.out.println(nireOsoa); // Irteera 4
```





# Moten arteko transformazioak

- Estutzeko transformazioa egiteko bi modu erabili daitezke:

- Objektu datu motak erabilita

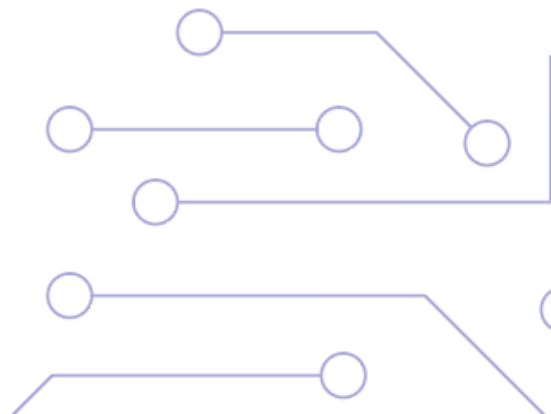
- Math eta String objektuek bezala funtzioak dituzte
    - Zenbaki osotik komadun zenbakira

```
Integer a = sc.nextInt();  
float f = a.floatValue(); // edo double d = a.doubleValue();
```

- Komadun zenbakitik zenbaki osora

```
Float a = sc.nextFloat();  
int i = a.intValue();
```

```
Double a = sc.nextDouble();  
int i = a.intValue();
```



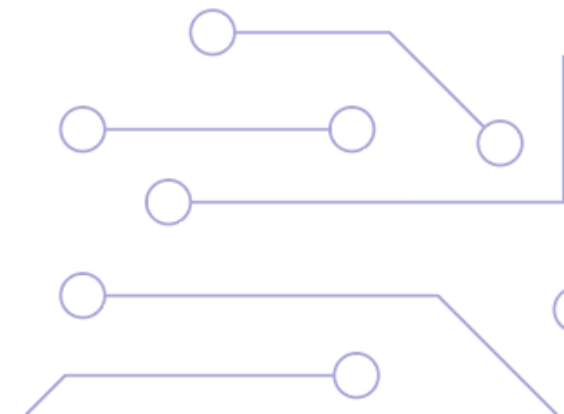
# Moten arteko transformazioak

- Estutzeko transformazioa egiteko bi modu erabili daitezke:
  - Objektu datu motak erabilita
    - Karaktere-kate motatik zenbakira

```
String s1 = "5";  
int a = Integer.parseInt(s1);
```

```
String s2 = "5.78";  
float b = Float.parseFloat(s2);
```

```
String s3 = "5.78";  
double c = Double.parseDouble(s3);
```

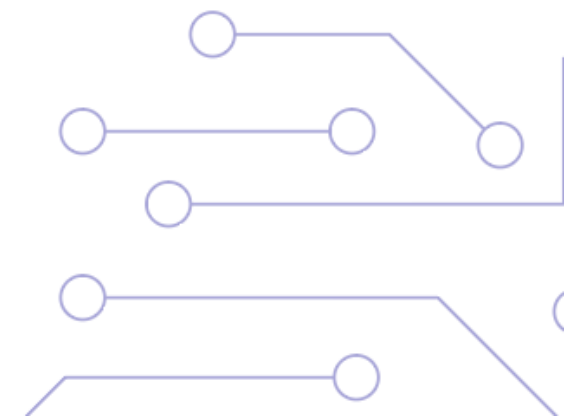


# Moten arteko transformazioak

- Estutzeko transformazioa egiteko bi modu erabili daitezke:
  - Objektu datu motak erabilita
    - Zenbaki motetatik eta karakteretik karaktere-katera

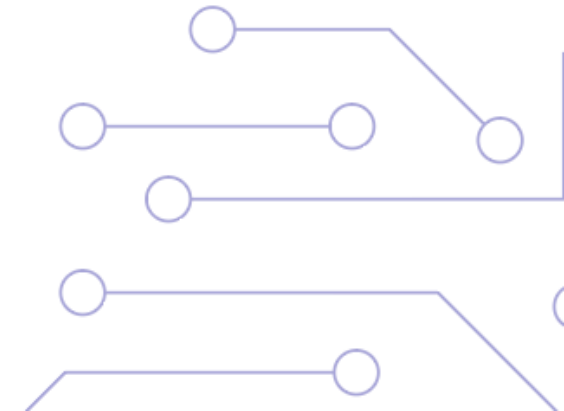
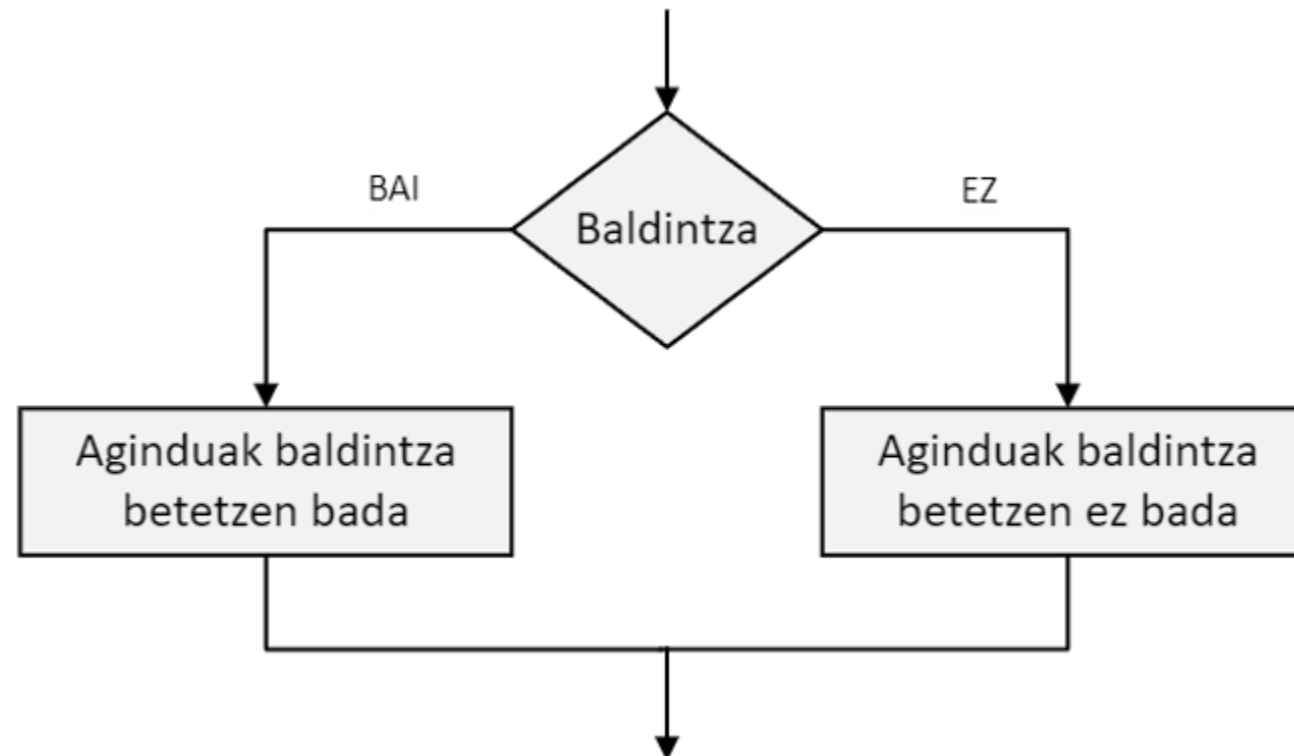
```
int a = 4;  
float b = sc.nextFloat();  
double c = 5.9037847;  
char d = 'd'
```

```
String s1 = String.valueOf(a);  
String s2 = String.valueOf(b);  
String s3 = String.valueOf(c);  
String s4 = String.valueOf(d);
```



# Baldintza egiturak

- Baldintza egiturak programak aldagai baten edo batzuen balioen arabera bide bat edo beste bat har dezan erabiltzen dira



# Eragile erlazionalak

- Baldintza egituretan eta egitura errepikakorretan erabiltzen dira baldintzak adierazteko

Eragilea	Esanahia	Adibidea
>	Handiago	3 > 2
<	Txikiago	"ABC" < "abc"
==	Berdina	b == 3
>=	Handiago edo berdina	4 >= 5
<=	Txikiago edo berdina	"xyz" <= "abc"
!=	Desberdina	c != "b"
>	Handiago	3 > 2

# Eragile logikoak

- Baldintza bat baino gehiagoko egituretan erabiltzen dira baldintzak lotzeko
- Eragiketen lehentasuna: NOT -> AND -> OR

Eragilea	Esanahia	Adibidea
&&	Egia itzultzen du baldintza guztiak egiazkoak badira eta gezurra beste kasuetan	(7>4) && (2==1) //gezurra
	Egia itzultzen du baldintza bat gutxienez betetzen bada eta gezurra denak gezurrezkoak badira	(1==1    2==1) //egia
!	Kontrako balioa itzultzen du	!(2<5) //gezurra



# if egitura

- Baldintzak sortzeko egitura orokorra
- Hiru zatitan banatzen da
  - Hasierako baldintza betetzen denean egin behar dena
  - Hurrengo baldintzetako bat betetzen bada egin behar dena
  - Adierazitako baldintzarik ez bada betetzen egin behar dena
  - Behar beharrezkoa hasierako zatia da bakarrik

if

# if egitura

```
if (baldintza)
{
    //ekintzak baldintza betetzen bada
}
else if (beste baldintza)
{
    //ekintzak bigarren baldintza betetzen bada
}
else //ez bada baldintzarik betetzen
{
    //ekintzak baldintzarik ez bada betetzen
}
```

if



# if egitura

```
public static void main(String args[]) {  
    int a = 30, b = 30;  
    if (b > a)  
    {  
        System.out.println("b handiagoa da");  
    }  
    else if(a > b)  
    {  
        System.out.println("a handiagoa da");  
    }  
    else  
    {  
        System.out.println("a eta b berdinak dira");  
    }  
}
```

# switch-case egitura

- Aldagai bakar batek jaso ditzakeen balioak finituak eta jakinak direnean erabiltzeko baldintza egitura
  - Zenbaki oso edo karaktere motako aldagaia
- Aldagai horrek hartu ditzakeen balioetarako agindu desberdinak adieraziko dira
- Oso erabilgarria da menuak sortzeko
  - Adibidez, dei batean 1 sakatzen denean zerbitzu teknikoarekin hitz egiteko, 2 komertzial batekin...

switch

# switch-case egitura

```
switch (aldagaia)
{
case balio_1:
    // egin beharreko ekintzak aldagaiak balio_1 balioa duenean
    break;
case balio_n:
    // egin beharreko ekintzak aldagaiak balio_n balioa duenean
    break;
default:
    // aldagaiak aurretik adierazitako baliorik hartzen ez duenean
    egin beharreko ekintzak, normalean erroreak adierazteko
    erabiltzen da
    break;
}
```

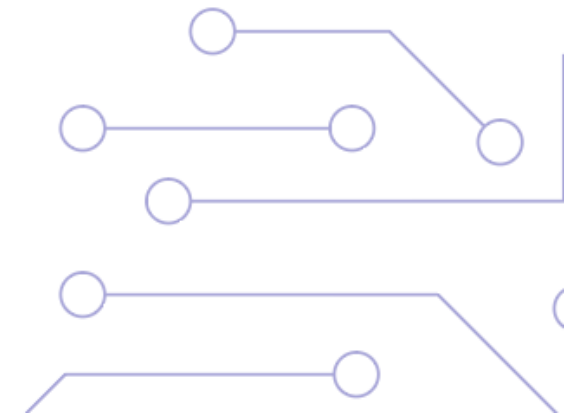
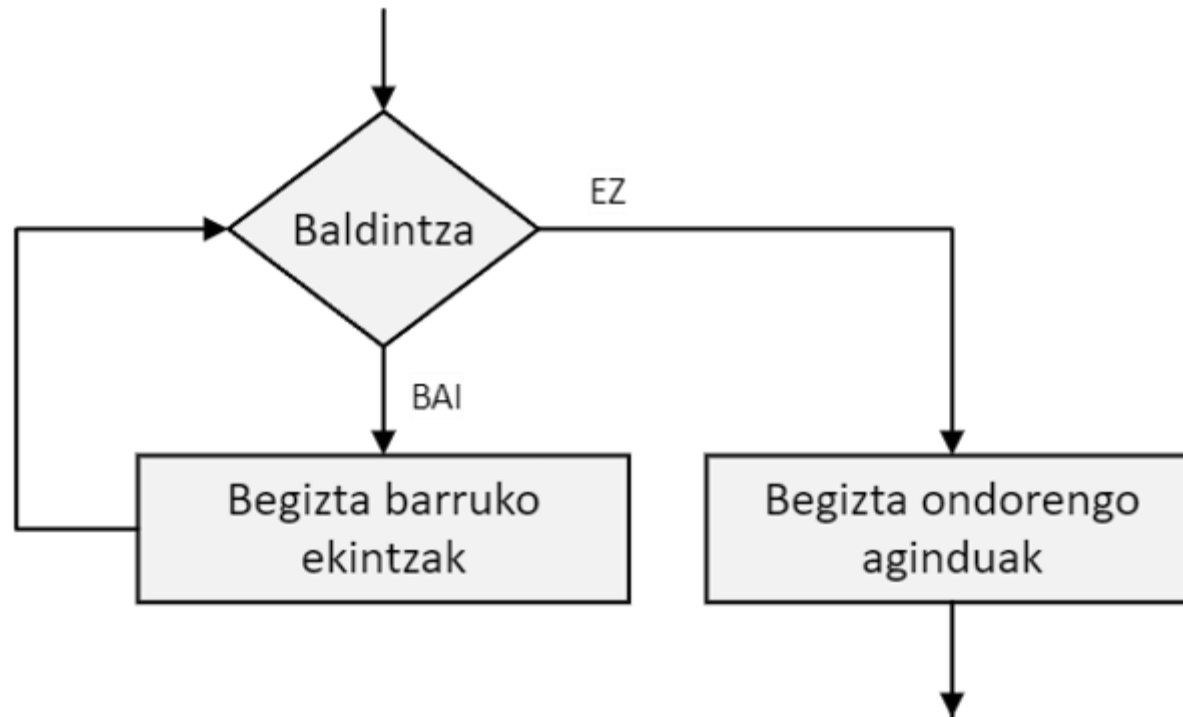
switch

# switch-case egitura

```
public static void main(String args[]) {  
    char c = 'b';  
    switch (c)  
    {  
        case 'a':  
            System.out.println("A letra sartu duzu");  
            break;  
        case 'b':  
            System.out.println("B letra sartu duzu");  
            break;  
        default:  
            System.out.println("Sarrerak ez du balio");  
            break;  
    }  
}
```

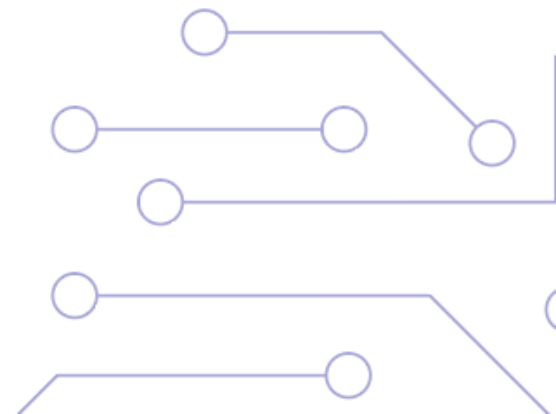
# Egitura errepikakorrak

- Baldintza bat betetzen den bitartean behin eta berriz errepikatu behar diren aginduak exekutatzeko erabiltzen diren egiturak dira
  - Begiztak (*bucle* gaztelaniaz) sortzen dituzte



# Egitura errepikakorrak

- Baldintza egituretan erabilitako eragile erlazional eta logiko berdinak erabiltzen dira
- Egituraren iterazio edo buelta bakoitzean kontagailu bat inkrementatzen (batzuetan dekrementatzen) doa, noizbait baldintza betetzera iristeko





## GARRANTZITSUA:

Begiztatik ateratzeko  
baldintza ondo adierazi  
behar da, bestela begizta  
amaigabe batean utzi  
daiteke programa!

# while egitura

- Egitura errepikakorrak definitzeko agindu orokorra
  - Edozein beste egitura errepikakor *while* batekin sortu daiteke
- Begiztako baldintza betetzen den bitartean aginduak errepikatzen ditu
- Irteera-baldintza hasieran ebaluatzen da
  - Gerta daiteke begizta barruko aginduak inoiz ez exekutatzea (hasieratik baldintza betetzen ez bada)
- Begizta kontrolatzeko kontagailu bat behar bada begiztan sartu aurretik definituta egon beharko da

while



# while egitura

```
while (baldintza)
{
    //ekintzak baldintza betetzen den bitartean
}
```

```
public static void main(String args[]) {
    int n = 1;
    while (n <= 5)
    {
        System.out.println("Iterazio kop.: " + n);
        n++;
    }
}
```



# for egitura

- Ekintza batzuk kopuru jakin bat errepikatzeko erabiltzen den egitura
- Errepikapen kopurua ezaguna denean egitura egokia da
- Begizten kontrola zenbakizko aldagai baten bidez egiten da
  - Normalean inkrementatzea edo dekrementatzea 1eko urratsarekin egiten da

for



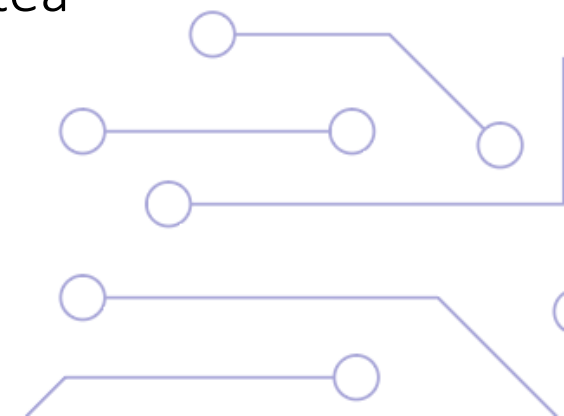
# for egitura

```
for (hasieratzea;baldintza;urratsa)
{
    //ekintzak baldintza betetzen den bitartean
}
```

```
public static void main(String args[]) {
    for (int n = 5; n >= 1; n--)
    {
        System.out.println("Iterazio kop.: " + n);
    }
}
```

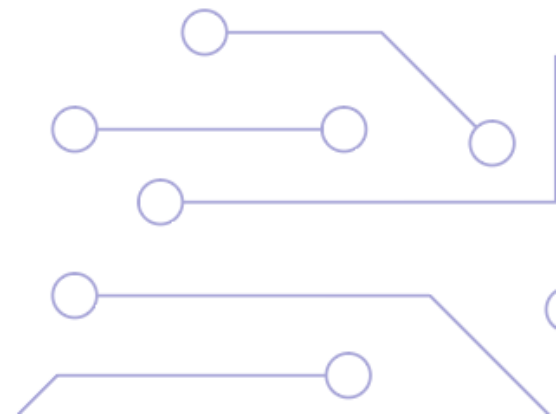
# Azpiprogramak

- Ebatzi behar den arazoa konplexuegia denean ideia ona da arazo txikiagoetan zatitzea
  - Arazoa zenbat eta zati txikiagoetan egin errazagoa izango da ebaztea
- Zati bakoitza azpiprograma baten bidez ebatzi daiteke
- Azpiprograma batek **parametro** izeneko datuak hartzen ditu sarrera bezala
  - Azpiprograma definitzen denean parametroen datu mota adieraztea beharrezkoa da



# Azpiprogramak: Abantailak

- Modularizazioa:
  - Azpiprograma bakoitzak helburu jakin bat du
  - Kodearen berrerabilpena ahalbidetzen da
- Garapen denbora murriztea:
  - Programen lerro kopuru osoa murrizten da
  - Erroreak egiteko probabilitatea ere jaisten da
- Datuen independentzia eta ezkutatzea:
  - Programaren beste zatiekiko independentea da
  - Azpiprograma erabiltzen duen programak ez du nola eginda dagoen jakin behar



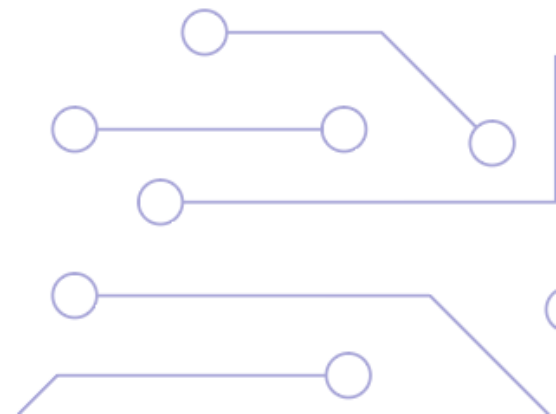


# Azpiprogramak: Adibidea

- a eta b bi zenbaki oso irakurrita eta kontutan hartuta  $a \geq b$  dela, ebatzi hurrengo espresio matematikoa:

$$f(a, b) = \frac{a!}{b! (a - b)!}$$

- Kasu honetan hiru faktorial kalkulatu behar dira eta hirurak kalkulatzeko dira modu berdinean
  - Faktorialaren kalkulua behin definitzearekin nahikoa da



# Azpiprogramak: Adibidea

- Sasikodea:

```
Algoritmoa espresioFaktoriala
```

```
  Irakurri a
```

```
  Irakurri b
```

```
  Baldin eta  $a \geq b$  Orduan
```

```
    emaitzaA <- faktoriala(a)
```

```
    emaitzaB <- faktoriala(b)
```

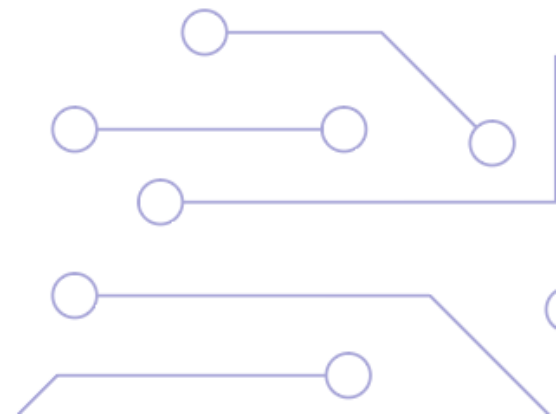
```
    emaitzaAB <- faktoriala(ab)
```

```
    emaitza <- emaitzaA / (emaitzaB * emaitzaAB)
```

```
    Idatzi emaitza
```

```
  Bukatu baldin
```

```
Bukatu algoritmoa
```





# Azpiprogramak: Adibidea

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Sartu zenbaki oso bat: ");
        int a = sc.nextInt();
        System.out.println("Sartu beste zenbaki oso bat: ");
        int b = sc.nextInt();
        sc.close();
        if (a < b)
        {
            System.out.println("Lehen zenbakia ezin da izan bigarrena baino txikiagoa!");
        }
        else
        {
            long aFakt = faktoriala(a);
            long bFakt = faktoriala(b);
            long abFakt = faktoriala(zenbaki: a - b);
            long emaitza = aFakt / (bFakt * abFakt);
            System.out.println("Espresioaren emaitza " + emaitza + " da.");
        }
    }
}
```

```
public static long faktoriala(int zenbaki)
{
    long emaitza = 1;
    for (int i = 1; i <= zenbaki; i++)
    {
        emaitza = emaitza * i;
    }
    return emaitza;
}
```