

# JAVA

1. Programa baten itxura
2. Hitz erreserbatuak
3. Aldagaiak eta konstanteak
4. Teklatutik irakurri (Scanner)
5. Baldintzak
6. Errepikapenak
7. Funtzioak
8. Fitxategiak irakurri

# 1. Programa baten itxura

```
public class KaixoMundua {  
    /* KaixoMundua programa */  
    public static void main(String[] args) {  
        /*programa honek gauza bakarra egiten du:  
        "KaixoMundua" katea pantailan atera*/  
        System.out.println("Kaixo, mundua!");  
    }  
}
```

## 2. Hitz erreserbatuak

abstract	char	double	for	int
private	strictfp	throws	assert	boolean
class	else	goto	interface	protected
super	transient	enum	break	const
extends	if	long	public	switch
try	byte	continue	final	implements
native	return	synchronized	void	case
default	finally	import	new	short
this	volatile	catch	do	float
instanceof	package	static	throw	while

### 3. Aldagaiak eta Konstanteak: Datu mota simpleak

Datu mota	Adierazitako informazioa	Heina	Azalpena
byte	Zenbaki osoak	- 128 ----- +127	8 bit erabiltzen dira datua gordetzeko
short	Zenbaki osoak	- 32768 ----- +32767	16 bit erabiltzen dira datua gordetzeko
int	Zenbaki osoak	- 2147483648 ----- +2147483647	32 bit erabiltzen dira datua gordetzeko
long	Zenbaki osoak	- 9223372036854775808 ----- + 9223372036854775807	64 bit erabiltzen dira datua gordetzeko
char	UNICODE karaktereak	0 ----- 65535	Zenbakiak UNICODE bidez adierazteko erabiltzen da.
float	Koma mugikorrekoko datuak (32 bit)	7 digituko zehaztasuna gutxi gorabehera.	Bit 1 zeinurako, 11 bit berretzailerako eta 24 mantisarako.
double	Koma mugikorrekoko datuak (64 bit)	16 digituko zehaztasuna gutxi gorabehera.	Bit 1 zeinurako, 11 bit berretzailerako eta 52 mantisarako.
boolean	Balio logikoak	true/false	Adierazpen boolear bat egia (true) edo faltsua (false) den adierazteko erabiltzen da.

### 3. Aldagaiak eta Konstanteak deklaratzeko

Adibidea Aldagaia:

```
int zenbakia = 0;
```

```
int zbk1=10, zbk2;
```

Adibidea Konstantea:

```
final int    ZENBAKIA=10;
```

### 3. Aldagaiak eta Konstanteak: Eragile Aritmetikoak

<i>Eragilea</i>	<i>Eragiketa Javan</i>	<i>Adierazpena Javan</i>	<i>Emaitza</i>
+	Batuketa	$1.2 + 9.3$	10.5
-	Kenketa	$312.5 - 12.3$	300.2
*	Biderketa	$1.7 * 1.2$	2.04
/	Zatiketa	$0.5 / 0.2$	2.5
%	Hondarra	$25 \% 3$	1

### 3. Aldagaiak eta Konstanteak: Esleipen-eragileak

<i>Eragilea</i>	<i>Adibidea Javan</i>	<i>Adierazpen baliokidea</i>
<code>+=</code>	<code>op1 += op2</code>	<code>op1 = op1+op2</code>
<code>-=</code>	<code>op1 -= op2</code>	<code>op1 = op1-op2</code>
<code>*=</code>	<code>op1 *= op2</code>	<code>op1 = op1*op2</code>
<code>/=</code>	<code>op1/= op2</code>	<code>op1 = op1/op2</code>
<code>%=</code>	<code>op1 %= op2</code>	<code>op1 = op1%op2</code>
<code>=</code>	<code>op1 = op2</code>	<code>op1 = op2</code>

### 3. Aldagaiak eta Konstanteak: Erlaziozko-eragileak

Eragilea	Adibidea Javan	Esanahia
<	$A < B$	A txikiagoa da B baino
>	$A > B$	A handiagoa da B baino
<=	$A \leq B$	A txikiagoa da B baino edo berdina
>=	$A \geq B$	A handiagoa da B baino edo berdina
!=	$A \neq B$	A eta B desberdinak
=	$A = B$	A eta B berdinak



## 4. Teklatutik irakurri (Scanner)

Oraindik ez dakigu klaseak eta paketeak zer diren. Horrek ez gaitu arduratu behar: *Scanner* klasea teklatu bidez sartutako datuak irakurtzeko erabiltzen da, eta erabili ahal izateko pakete bat inportatu behar da.

1. **Paketea inportatu:** *Scanner* klasea **java.util** paketearen barruan dago; beraz, programaren hasieran hau idatziko dugu: **import java.util.Scanner;**
2. **Scanner objektu bat sortu:** *Scanner* objektu bat sortu eta sarrerako gailuarekin erlazionatu behar dugu. Sarrerako gailua teklatua bada, hau idatziko dugu:

**Scanner sc = new Scanner(System.in);**

Egin dugunarekin sc objektua sortu dugu, eta teklatuarekin erlazionatuta dago (***System.in***).

## 4. Teklatutik irakurri (Scanner): Adibidea

```
Scanner sc = new Scanner(System.in);  
double x; // x aldagaia sortuko dugu  
System.out.print("Sartu zenbaki erreal bat: "); // pantailan mezu bat aterako  
dugu  
x = sc.nextDouble(); erabiltzaileak sartutako datua n aldagaian gordeko dugu.
```

## 4. Teklatutik irakurri (Scanner): Funtzionamendua

### Nola funtzionatzen du Java Scanner klaseak?

Teklatu bidez karaktereak sartzen direnean, *Scanner* objektuak osorik hartzen du sartutako katea eta osagaitan zatitzen du: *token-ak*.

Token-ak banatzen dituen karakterea zuriunea da.

Testu hau sartzen badugu:

Hau adibidea da. Datuak irakurtzea.

Scanner objektuak katea zatituko du, eta token hauek sortuko:

Hau  
adibidea  
da.  
Datuak  
irakurtzea.

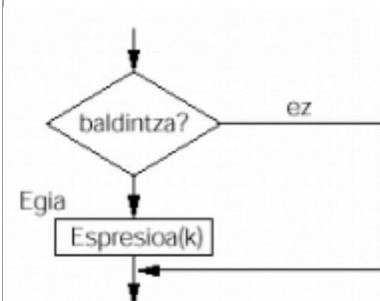
## 4. Teklatutik irakurri (Scanner): Beste metodo batzuk

Metodoa	Esanahia
nextXXX()	Osteko tokena itzultzen du oinarrizko datu mota moduan. xxx datu mota da. Adibidez, nextInt() zenbaki oso bat irakurtzeko, nextDouble double bat irakurtzeko, etab.
next()	Hurrengo tokena itzultzen du String moduan.
nextLine()	Lerro osoa itzultzen du String moduan. Intro karakterea garbitzen du.
hasNext()	Boolear bat itzultzen du. Irakurtzeko beste tokenik geratzen den ala ez adierazten du.
hasNextXXX	Boolear bat itzultzen du. Irakurtzeko beste tokenik geratzen den ala ez adierazten du, baina datu mota jakin batekoa. Adibidez: hasNextDouble()

## 5. Baldintzak: IF eta IF-ELSE

Adierazpen logiko bat ebaluatzen du, eta emaitzaren arabera, sententzia bat edo sententzia multzo bat exekutatzen du.

IF		IF-ELSE	
if (adierazpen logikoa) sententzia 1;	if (adierazpen logikoa){ sententzia 1; sententzia 2; ... ; sententzia N; }	if (adierazpen logikoa) sententzia 1; else sententzia2;	if (adierazpen logikoa){ sententzia 1; ... ; sententzia N; } else{ sententzia 1; ... ; sententzia N; }
Adierazpen logikoa egia bada, sententzia 1 exekutatuko da edo dagokion sententzia-blokea. Faltsua bada, ez da hautazko egiturako sententziarik exekutatuko.		Adierazpen logikoa egia bada, sententzia 1 edo lehenengo sententzia-blokea exekutatuko da. Faltsua bada, berriz, sententzia 2 edo bigarren sententzia-blokea exekutatuko da.	



## 5. Baldintzak: SWITCH

Zer egin dezakegu gure programak bi aukera baino gehiagoren artean aukeratu behar duenean? Aukera bat *if* egiturak kateatzea izan liteke; baina, askotan, aukera hori ez da eraginkorra. Kasu horietan, *switch* egitura erabiliko dugu.

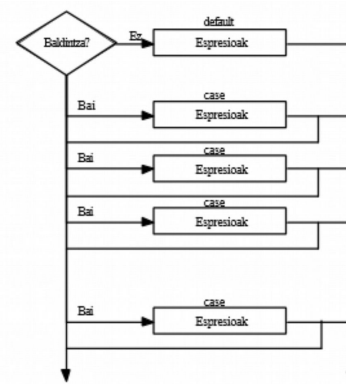
### SWITCH

```
switch (adierazpen logikoa){  
  case balioa1:  
    sententzia 1_1;  
    sententzia 1_2;  
    ...      ;  
    break;  
  case...  
    ....;  
    break;  
  case balioaN:  
    sententzia N_1;  
    sententzia N_2;  
    ...      ;  
    break;  
  default:  
    default-sententziak;  
}
```

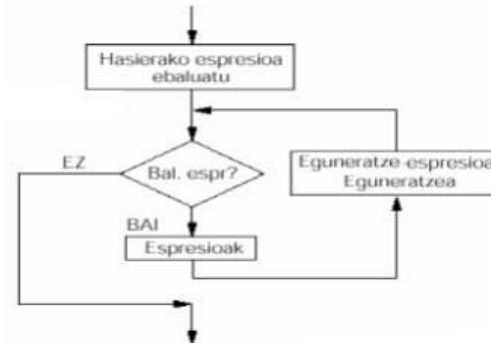
Adierazpena parentesi artean jartzen da.

- **case** bakoitzak balio bat izan behar du eta bi punturekin (:) amaituko da.

- **default**: sententziak batekin amaitu daitezke ala ez.



## 6. Errepikapenak: FOR egitura



### FOR

for (hasieratzea; baldintza; eguneratze-espresioa)

    sententzia 1\_1;

**sententzia bakarreko for egitura**

for (hasieratzea; baldintza; eguneratze-espresioa){

    sententzia 1;

    sententzia 2;

    ...

    sententzia N;

}

**sententzia blokea duen for egitura**

#### Hasieratzea:

da. Aldagai honek kontrolatuko du begiztaren eragina noiz amaituko den.

#### Baldintza:

Baldintza egia den bitartean, begizta errepikatuko da. Faltsua denean, begizta amaitu egingo da.

#### Eguneratze-espresioa:

balioa aldatzen doa begizta exekutatzen den bakoitzean. Balio horrek gora edo behera egin dezake.

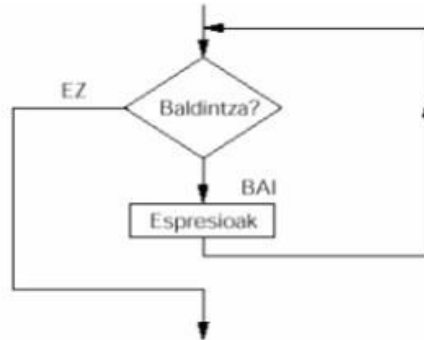
## 6. Errepikapenak: While egitura

### WHILE

while (baldintza)

sententzia 1;

**sententzia bakarreko while egitura**



```
while (baldintza){
```

```
sententzia 1;
```

```
sententzia 2;
```

```
...
```

```
sententzia N;
```

```
}
```

**sententzia-blokea duen while egitura**

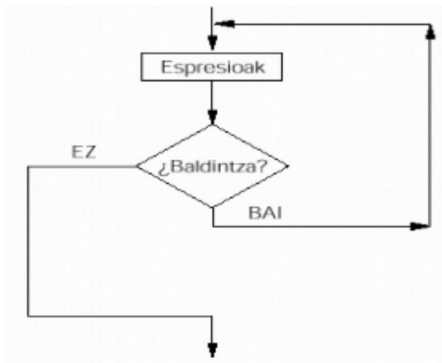
Baldintza egia den bitartean, begizta errepikatu egingo da eta gorputzeko sententziak exekutatuko ditu.

Baldintza faltsua izatera pasatzen den unean, programaren fluxua sententziara pasatuko da.

Baldintza hasieran ebaluatuko da beti, eta gerta liteke haren gorputzean dauden aginduak inoiz ez exekutatzea, hasieratik baldintza faltsua bada.



## 6. Errepikapenak: do While

DO WHILE		
<pre>do   sententzia 1; while (baldintza);</pre>	 <pre>do{   sententzia 1;   sententzia 2;   ...   sententzia N; } while (baldintza);</pre>	
<b>sententzia bakarreko do/while egitura</b>		<b>sententzia-blokea duen do/while egitura</b>

Gorputzeko kodea behin exekutatu da, eta gero, baldintza ebaluatuko da. Baldintza egia den bitartean, begizta errepikatu egingo da eta gorputzeko sententziak exekutatu ditu.

Baldintza faltsua izatera pasatzen den unean, programaren fluxua dagoen sententziara pasatuko da.

Gorputza baldintza ebaluatu aurretik exekutatzen da, beraz, ezin liteke gertatu gorputzean dauden aginduak inoiz ez exekutatzea.

## 7. Funtzioak

Programa modulutan deskonposatuko da, horretarako prozedurak eta funtzioak erabiliko dira.

### **ONURAK:**

- **Programa ulertzea**
- **Modularitatea**
- **Errezago mantentzia**
- **Ziurtasuna**
- **Berrerabilgarritasuna**

### **Funtzio bat sortu:**

```
public class Adibidea {  
    public static void main (String [] args){  
        sentzentziak;  
        //ez du ezer bueltatzen  
    }  
  
    public void kaixoMundua (atributuak){  
    }  
    public int balioaBuelatu(atributuak){  
        sententziak ...  
        //buelatu behar du zbk oso bat  
        return int  
    }  
}
```

## 8. Fitxategiak irakurri

//fitxategiak irakurtzeko eta idazteko clasea inportatu

```
import java.io.File;  
import java.util.Scanner;
```

```
public class Adibidea {  
    public static void main ... {  
        File f = new File ("fitxategirako bidea"); // adi ruta jartzerakoan "\\" bikoitza!  
        Scanner sc = new Scanner (f);  
        while (sc.hasNext()){  
            sententziak 1  
        }  
        sc.close() //konexioa itxi  
    }  
}
```