- Constructors
- Sobrecàrrega de mètodes
- Operador this

1.- Constructors a Java.

Un constructor inicialitza un objecte quan es crea. Té el mateix nom que la seua classe i és sintàcticament similar a un mètode. Quan es construeix un objecte és necessari inicialitzar els seus atributs amb valors coherents.

Totes les classes tenen constructors. Si no en definim, **Java crea** automàticament un constructor **predeterminat**. En aquest cas, els atributs no inicialitzats tenen els seus valors predeterminats, que són **zero** (per a tipus numèrics), **null** (tipus de referència) **i false** (booleans).

Quan definim un constructor, el constructor predeteminat ja no s'usa.

Característiques dels constructors:

- **Es diu igual que la classe.**
- **No retorna res**, ni tan sols void.
- Poden existir més d'un, però seguint les regles de la **sobrecàrrega de funcions**.
- D'entre els constructors que existisquen, **NOMÉS s'executarà un** en crear un objecte de la classe.

Exemple :

Definició de la classe PERSONA

```
public class Persona {

// Atributs de la classe Persona

   int edat;
   String nom;

// Constructors de la classe Persona
   public Persona(){
     edat = 0;
     nom = "anònim";
   }

   public Persona(String nouNom){
     edat = 0;
     nom = nouNom;
   }

   public Persona(String nouNom, int no-
vaEdat){

     nom = nouNom;
     edat = novaEdat;

   }
}
```

Creació i ús d'OBJECTES (instàncies) de la classe PERSONA

```
public class Main {

   public static void main(String[] args) {

       Persona P1 = new Persona();

       Persona P2 = new Persona("Maria");

        Persona P3 = new Persona("Anna", 26);


  System.out.println("Em dic " + P1.nom + " i
    tinc "+ P1.edat + " anys");

  System.out.println("Em dic " + P2.nom + " i
    tinc "+ P2.edat + " anys");

  System.out.println("Em dic " + P3.nom + " i
    tinc "+ P3.edat + " anys");


  }
}
```

Eixida:

```
Em dic anònim i tinc 0 anys
Em dic Maria i tinc 0 anys
Em dic Anna i tinc 26 anys
```

És important definir un constructor o més que controlen el que ha de passar quan es crea un objecte.

2.- Sobrecàrrega de Mètodes i Constructors.

La definició d'un mètode és la combinació del **nom** i els tipus dels **paràmetres o arguments**.

```
int calculaSuma(int x, int i, int z)    { ... }
```

La sobrecàrrega de mètodes és la creació de diversos mètodes amb el mateix nom dins de la classe, però amb diferent número o tipus d'arguments .

Java utilitza el número i tipus de paràmetres per a seleccionar quin definició de mètode executar.

Java diferencia els mètodes sobrecarregats amb base en el **número** i **tipus** de paràmetres o arguments que té el mètode i **no pel tipus que retorna**.

/* Mètodes sobrecarregats */	/* Error: aquests mètodes no estan sobrecarregats */

`int calculaSuma(int x, int i, int z){`
`…`
`}`
 	
`int calculaSuma(double x, double i, double z){`
`…`
`}`
	`int calculaSuma(int x, int i, int z){`
	`…`
	`}`
	
	`double calculaSuma(int x, int i, int z){`
	`…`
	`}`

També existeix la **sobrecàrrega de constructors.** Algunes voltes podem tindre la necessitat d'inicialitzar un objecte de diferents maneres. La **sobrecàrrega de constructors** ens permetrà construir objectes de diverses maneres. Com podem veure al següent exemple, segons com creem l'objecte, es crida a un constructor o altre

Constructors Sobrecarregats	Us sobrecàrrega de constructors

`class MiClase{`
`  int x;`
 	
`  MiClase(){`
 	
`    System.out.println("Dins de MiClase().");`
`    x=0;`
`  }`
 	
`  MiClase(int i){`
 	
`    System.out.println("Dins de MiClase(int).");`
`    x=i;`
`  }`
	`class SobrecargaConstructor{`
	`public static void main(String[] args) {`
	`/* En definir diversos constructors podem crear objectes de diverses maneres */`
	` MiClase t1 = new  MiClase();`
	` MiClase t2 = new  MiClase(28);`
	` MiClase t3 = new  MiClase(15.23);`
	` MiClase t4 = new  MiClase(2,4);`
	
	` System.out.println("t1.x: "+ t1.x);`
	` System.out.println("t2.x: "+ t2.x);`
	` System.out.println("t3.x: "+ t3.x);`
	` System.out.println("t4.x: "+ t4.x);`
	` }`
	`}`

```
  MiClase(double d){

    System.out.println("Dins de MiClase(double).");
    x=(int)d;
  }

  MiClase(int i, int j){

    System.out.println("Dins de MiClase(int,
int)");
     x=i*j;
  }
}
```

EIXIDA:

Dins de MiClase().
Dins de MiClase(int).
Dins de MiClase(double).
Dins de MiClase(int, int).
t1.x: 0
t2.x: 28
t3.x: 15
t4.x: 8

3.- Operador this.

L'operador **this** fa referència als atributs de l'objecte. I cal utilitzar-lo, encara que no siga obligatori, per tal de distingir entre atributs i variables locals.

Definició de la classe PERSONA

```
public class Persona {

// Atributs de la classe Persona
   int edat;
   String nom;

// Constructors de la classe Persona
   public Persona(){
     this.edat = 0;
     this.nom = "anònim";
   }

   public Persona(String nouNom){
     this.edat = 0;
     this.nom = nouNom;
   }

   public Persona(String nom, int edat){

    this.edat = edat;
    this.nom = nom;
   }
}
```

Creació i ús d'OBJECTES (instàncies) de la classe PERSONA

```
public class Main {

   public static void main(String[] args) {

       Persona P1 = new Persona();

       Persona P2 = new Persona("Maria");

        Persona P3 = new Persona("Anna", 26);


  System.out.println("Em dic " + P1.nom + " i
    tinc "+ P1.edat + " anys");

  System.out.println("Em dic " + P2.nom + " i
    tinc "+ P2.edat + " anys");

  System.out.println("Em dic " + P3.nom + " i
    tinc "+ P3.edat + " anys");


  }
}
```

Eixida:

```
Em dic anònim i tinc 0 anys
Em dic Maria i tinc 0 anys
Em dic Anna i tinc 26 anys
```

Dels tres constructors de persona, només en el tercer és obligat l'us de **this**. Ja que els paràmetres del constructor: **public** Persona(String nom, **int** edat) tenen el mateix nom que els atributs de la classe:

```
int edat;
String nom;
```

Nota: Si no posem **this**, no donarà error ni d'execució ni de compilació, però els resultats no seran correctes.

Per exemple, en el següent constructor de classe:

```
public Persona(String nom, int edat) {
      edat = edat;
      nom = nom;
}
```

I si al **main()** posem:

```
Persona P3 = new Persona("Anna", 26);
System.out.println("Em dic " + P3.nom + " i  tinc "+ P3.edat + " anys");
```

L'eixida seria:

```
Em dic null i tinc 0 anys
```

Quan esperàvem:

```
Em dic Anna i tinc 26 anys
```