

Programació I - T1 - Introducció



UNIVERSITAT DE
BARCELONA

Grau en Enginyeria Informàtica
Facultat de Matemàtiques i Informàtica
Curs 18-19



Tema 1: Introducció

- 1.1. Conceptes: programa, computador, llenguatge de programació
- 1.2. Conceptes bàsics de programació amb Java (OO)

Punt actual

- 1 Tema 1: Introducció
 - 1.1. Conceptes: programa, computador, llenguatge de programació
 - 1.2. Conceptes bàsics de programació amb Java (OO)

1.1. Conceptes bàsics

Programa d'ordinador

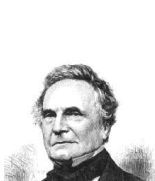
- Problema.
- Estratègia per a solucionar-lo.
- Programa d'ordinador:
 - Conjunt d'instruccions (i dades) que poden ser executades ordenadament en el temps per un computador, per a **solucionar un problema** concret.
 - Successió d'estats definits pels valors que les dades van prenent.

1.1. Conceptes bàsics

Computador

- Computador: màquina electrònica capaç de realitzar càlculs i tractar grans quantitats de dades de forma automàtica, seguint un conjunt d'instruccions.
 - Charles Babbage (1792-1871): "Màquina analítica".
 - Ada Lovelace (1815-1852): Primera programadora d'ordinadors.
 - Alan Turing (1912-1954): Màquina de Turing i Test de Turing.

<https://plato.stanford.edu/entries/computing-history/>



1.1. Conceptes bàsics

Computador - Parts d'un computador: maquinari (Hardware)

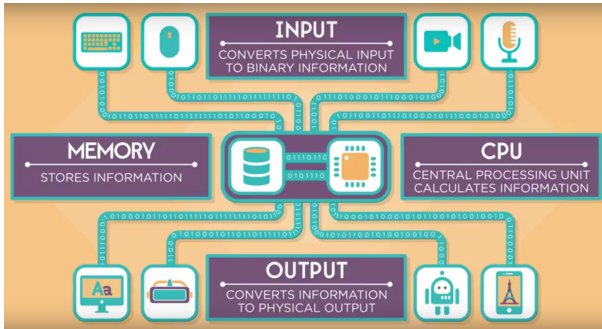


Image from 'How computers work', code.org

<https://www.youtube.com/watch?v=DKGZlaPlVLY&list=PLzdnOP1liJNcsRwJhvksEoltJqjIqWbN-&index=5>

1.1. Conceptes bàsics

Computador - Memòria

- Memòria: Com es guarden les dades?
 - Es tenen dades de diferents **tipus de dades**, com nombres, caràcters, strings (cadena de caràcters) que es codifiquen com a sèries de bits (zeros i uns).
 - Per exemple, la lletra 'J' es representa per el byte (8 bits) 01001010.
 - Si es necessita més d'un byte, el computador guarda el nombre adjacent de bytes que es necessitin.
 - Mesures derivades del byte: KB, MB, GB, TB, etc.

Memory address	Memory content	
.	.	
.	.	
.	.	
2000	01001010	Encoding for character 'J'
2001	01100001	Encoding for character 'a'
2002	01110110	Encoding for character 'v'
2003	01100001	Encoding for character 'a'
2004	00000011	Encoding for number 3

1.1. Conceptes bàsics

Computador - Parts d'un computador: programari (Software)

- Programari és el conjunt de programes que s'executen a un ordinador.

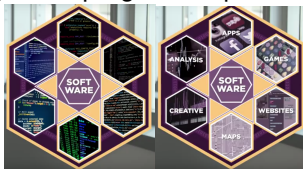
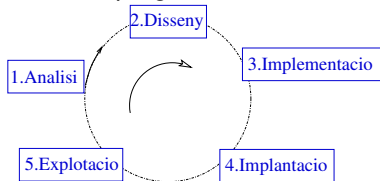


Image from 'How Computers Work: Hardware and Software', code.org

<https://www.youtube.com/watch?v=xnyFYiK2rSY&list=PLzdnOP1liJNcsRwJhvkseoltJqjIqWbN-&index=6>

- Etales de desenvolupament de programari.



[https://ca.wikipedia.org/wiki/Proc%C3%A9s_de_desenvolupament_de_programari#Etales_del_desenvolupament_de_programari\[3\]](https://ca.wikipedia.org/wiki/Proc%C3%A9s_de_desenvolupament_de_programari#Etales_del_desenvolupament_de_programari[3])

1.1. Conceptes bàsics

Llenguatge de programació

- Formalisme mitjançant el qual es poden crear programes d'ordinador.



- Evolució dels llenguatges: <http://www.levenez.com/lang/>

1.1. Conceptes bàsics

Llenguatge de programació

- Paradigmes de llenguatges de programació: funcionals, imperatius, lògics, orientats a objecte (OO)...
- Compiladors i intèrprets: eficiència versus portabilitat.
 - Codi font, codi objecte, codi executable.
 - Bytecode i màquina virtual.

1.1. Conceptes bàsics

Llenguatge de programació

- Plataforma Java (Sun-Oracle): James Gosling (1991)
 - Codificació en llenguatge de programació Java
 - Compilació a llenguatge intermig bytecode (**javac**)
 - Execució en la màquina virtual (**java**)
 - ▶ Muntatge de tots els blocs necessaris
 - ▶ Interpretació o Traducció al llenguatge màquina real

Punt actual

- 1 Tema 1: Introducció
 - 1.1. Conceptes: programa, computador, llenguatge de programació
 - 1.2. Conceptes bàsics de programació amb Java (OO)

1.2. Conceptes bàsics de programació amb Java (OO)

Conceptes de classe, objecte, mètode



1.2. Conceptes bàsics de programació amb Java (OO)

Conceptes de classe, objecte, mètode

- **Una classe representa totes les 'coses' d'un mateix tipus.**
- La classe Gos representa a tots els gosos, la classe Cotxe representa a tots els cotxes, etc.

1.2. Conceptes bàsics de programació amb Java (OO)

Conceptes de classe, objecte, mètode

- Una classe ens serveix per a representar els **atributs** i **accions/comportaments** comuns a una serie de 'coses'.
- Una vegada definida la classe, es poden crear tants objectes d'aquesta classe com es vulgui. Per exemple:
 - La classe Gos defineix com és (propietats i atributs) i què pot fer (accions) qualsevol Gos.
 - La classe Gos serveix per a crear objectes 'concrets' d'aquesta classe, cadascun amb els seus atributs (raça, altura, ...). Per exemple, elMeuGos, elTeuGos, etc.



- A un **objecte** d'una classe també se'l coneix com 'instància d'una classe'.

1.2. Conceptes bàsics de programació amb Java (OO)

Conceptes de classe, objecte, mètode

- **Mètode:** Acció que pot realitzar un objecte d'una classe.
- Per ara parlarem de *Mètode d'objecte*: es un mètode que es **crida** per a què un objecte concret d'una classe realitzi una acció.
- **Sintaxis:** `nom_objecte.nom_metode()` ;
- Per exemple, els mètodes `lladrar()`, `dormir()`, de la classe `Gos` s'utilitzen per a fer lladrar i dormir a un gos concret:

- `elMeuGos.lladrar()` ; //Es demana lladrar a elMeuGos (es crida el mètode `lladrar()`)
- `elTeuGos.dormir()` ; //Es demana dormir a elTeuGos (es crida el mètode `dormir()`)

1.2. Conceptes bàsics de programació amb Java (OO)

Conceptes de classe, objecte, mètode

Exemple a Greenfoot. Mètodes de la classe HedgeHogs

The screenshot shows the Greenfoot IDE interface. The main window displays a green grid world with several hedgehog actors. A context menu is open over one of the hedgehogs, listing methods inherited from the Actor class:

- void act()
- boolean canMove()
- void eatApple()
- boolean foundApple()
- int getApplesEaten()
- void move()
- void turnLeft()
- void turnRandom()

Below the list are the options "Inspect" and "Delete".

On the right side, the class hierarchy is shown:

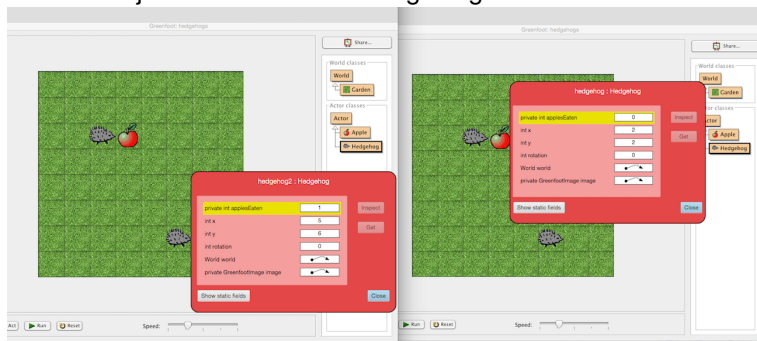
- World classes
 - World
 - Garden (inherits from World)
- Actor classes
 - Actor
 - Apple (inherits from Actor)
 - Hedgehog (inherits from Actor)

At the bottom, there are buttons for "Act", "Run", and "Reset", along with a "Speed" slider.

1.2. Conceptes bàsics de programació amb Java (OO)

Conceptes de classe, objecte, mètode

Ex. Dos objectes de la classe HedgeHogs. Cadascun amb els seus atributs!!



1.2. Conceptes bàsics de programació amb Java (OO)

Conceptes de classe, objecte, mètode

- Fins ara, hem parlat de la crida als mètodes, però per a cridar (les vegades que es vulgui!!) un mètode, aquest ha d'estar definit.
- Formalment, un mètode és un bloc de sentències (instruccions i dades) etiquetat amb un identificador, que pot rebre dades i retornar dades.

1.2. Conceptes bàsics de programació amb Java (OO)

Conceptes de classe, objecte, mètode

- **Sintaxi Java de la definició d'un mètode:**

```
<tipus_retorn> <nom_mètode>(<paràmetres>) {  
.....  
..... //Sentències que defineixen el mètode  
.....  
    <return <expressió>;>  
}
```

- **<nom_mètode>:** Nom del mètode.
- **<paràmetres>:** Valors que pot rebre el mètode per a utilitzar-ne dintre. Pot no rebre cap.
- **<tipus_retorn>:** Tipus de dades del valor/s que retorna un mètode.

1.2. Conceptes bàsics de programació amb Java (OO)

Conceptes de classe, objecte, mètode

- `<tipus_retorn>`: Un mètode pot:
 - retornar un valor, s'indica amb una paraula clau que indica **el tipus de dades** que retorna, per exemple (**int**, **boolean**, **float**, ...). S'utilitza `return <expressió>;` per a fer-ho.
 - ▶ Per exemple, per a retornar l'edat d'un gos aquest valor serà enter (int), per a retornar si un gos està vacunat o no aquest valor serà veritat o fals (boolean).
 - NO retornar res, s'indica a `<tipus_retorn>` amb la paraula clau que indica buit, **void**. En aquest cas, no es necessari utilitzar `return <expressió>;`.
- Exemple:

```
int calcEdatHuma( ) {  
    return edat*7;  
}
```

1.2. Conceptes bàsics de programació amb Java (OO)

- Un mètode Java sempre pertany a una classe. Ho podem veure'l a l'estructura general d'un programa Java (Generic.java):

```
public class Generic {  
    //Només la classe principal, Generic, pot contenir el mètode  
    main()  
    //La classe principal sempre és 'public'  
    //Al main() comença l'execució de tot programa Java  
    public static void main(String[] args) {  
        //Es declaren entitats  
        //(ex. objectes de la classe Exemple)  
        <Declaracions Entitats (Dades)>  
        //A partir d'aquí s'utilitzen les entitats  
        //declarades abans  
        <Sentències (Instruccions)>  
    } //Aquí termina l'execució del programa  
    <Definicions de més mètodes de la classe Generic>  
}  
// Altre classe  
class Exemple{  
    <Declaracions atributs>  
    <Definicions mètodes>  
}
```

1.2. Conceptes bàsics de programació amb Java (OO)

Mètodes Constructors

- Mireu la classe `Gos`: `Gos()` i `Gos(String raca, int edat)` són mètodes 'especials' (anomenats **CONSTRUCTORS**) que només es criden per a crear un objecte.
- Mireu la classe `DemoGos`: la crida per a crear objectes d'una classe sempre es fa amb la paraula reservada `new` seguida de la crida al constructor.

```
public class DemoGos{  
  
    public static void main(){  
        int edatH;  
        Gos gos1 = new Gos("Pastor", 7);  
        Gos gos2 = new Gos("Westie", 5);  
        Gos gos3 = new Gos();  
  
        edatH = gos1.calcEdatHuma();  
        System.out.println(edatH);  
  
        edatH = gos2.calcEdatHuma();  
        System.out.println(edatH);  
  
        edatH = gos3.calcEdatHuma();  
        System.out.println(edatH);  
    }  
}
```

```
class Gos{  
  
    String raca;  
    int edat;  
  
    Gos () {  
    }  
  
    Gos (String raca, int edat){  
        this.raca = raca;  
        this.edat = edat;  
    }  
    int calcEdatHuma(){  
        return edat * 7;  
    }  
}
```

1.2. Conceptes bàsics de programació amb Java (OO)

Mètodes Constructors i funcionament de les crides a mètodes

- **IMPORTANT:** NO es pot utilitzar un objecte sense haver-lo creat (amb `new`) abans!
- Nosaltres hem definit un constructor buit, però encara que no s'hagués definit aquest constructor existeix un constructor per defecte!
- **Recordeu!** La crida a altres mètodes definits a la classe es fa amb la sintaxis:
`nomObjecte.nomMetode();` (veure crida `acalcEdatHuma()`)


```
public class DemoGos{  
  
    public static void main(){  
        int edatH;  
        Gos gos1 = new Gos("Pastor", 7);  
        Gos gos2 = new Gos("Westie", 5);  
        Gos gos3 = new Gos();  
  
        edatH = gos1.calcEdatHuma();  
        System.out.println(edatH);  
  
        edatH = gos2.calcEdatHuma();  
        System.out.println(edatH);  
  
        edatH = gos3.calcEdatHuma();  
        System.out.println(edatH);  
    }  
}
```

```
class Gos{  
  
    String raza;  
    int edat;  
  
    Gos () {  
    }  
  
    Gos(String raza, int edat){  
        this.raza = raza;  
        this.edat = edat;  
    }  
  
    int calcEdatHuma(){  
        return edat * 7;  
    }  
}
```



```
public class DemoGos{  
  
    public static void main(){  
        int edatH;  
        Gos gos1 = new Gos("Pastor", 7);  
        Gos gos2 = new Gos("Westie", 5);  
        Gos gos3 = new Gos();  
  
        edatH = gos1.calcEdatHuma();  
        System.out.println(edatH);  
  
        edatH = gos2.calcEdatHuma();  
        System.out.println(edatH);  
  
        edatH = gos3.calcEdatHuma();  
        System.out.println(edatH);  
    }  
}
```

```
class Gos{  
  
    String raza;  
    int edat;  
  
    Gos () {  
    }  
  
    Gos (String raza, int edat){  
        this.raza = raza;  
        this.edat = edat;  
    }  
  
    int calcEdatHuma(){  
        return edat * 7;  
    }  
}
```



```
public class DemoGos{  
  
    public static void main(){  
        int edatH;  
        Gos gos1 = new Gos("Pastor", 7);  
        Gos gos2 = new Gos("Westie", 5);  
        Gos gos3 = new Gos();  
  
        edatH = gos1.calcEdatHuma();  
        System.out.println(edatH);  
  
        edatH = gos2.calcEdatHuma();  
        System.out.println(edatH);  
  
        edatH = gos3.calcEdatHuma();  
        System.out.println(edatH);  
    }  
}
```

```
class Gos{  
  
    String raza;  
    int edat;  
  
    Gos () {  
  
    }  
  
    Gos(String raza, int edat){  
        this.raza = raza;  
        this.edat = edat;  
    }  
  
    int calcEdatHuma(){  
        return edat * 7;  
    }  
}
```

```
public class DemoGos{  
  
    public static void main(){  
        int edatH;  
        Gos gos1 = new Gos("Pastor", 7);  
        Gos gos2 = new Gos("Westie", 5);  
        Gos gos3 = new Gos();  
  
        edatH = gos1.calcEdatHuma();  
        System.out.println(edatH);  
  
        edatH = gos2.calcEdatHuma();  
        System.out.println(edatH);  
  
        edatH = gos3.calcEdatHuma();  
        System.out.println(edatH);  
    }  
}
```

```
class Gos{  
  
    String raca;  
    int edat;  
  
    Gos () {  
    }  
  
    Gos (String raca, int edat){  
        this.raca = raca;  
        this.edat = edat;  
    }  
    int calcEdatHuma(){  
        return edat * 7;  
    }  
}
```

IDEM AMB CRIDES A MÈTODES QUE RESTEN

1.2. Conceptes bàsics de programació amb Java (OO)

Referència this

- Mireu el mètode `Gos(String raca, int edat)`. Dintre d'aquest mètode, `this` referència a l'objecte que ha cridat al mètode.
- En qualsevol mètode, `this` sempre referència al objecte que ha cridat al mètode. La referència `this` sempre està, ja sigui implícita o explícita, com al nostre exemple.
- Al mètode `calcEdatHuma()`, podríem posar `this.edat` enlloc de `edat`? A on referència `this` amb la crida `gos1.calcEdatHuma()`, i amb `gos2.calcEdatHuma()`?

```
public class DemoGos{  
  
    public static void main(){  
        int edatH;  
        Gos gos1 = new Gos("Pastor", 7);  
        Gos gos2 = new Gos("Westie", 5);  
        Gos gos3 = new Gos();  
  
        edatH = gos1.calcEdatHuma();  
        System.out.println(edatH);  
  
        edatH = gos2.calcEdatHuma();  
        System.out.println(edatH);  
  
        edatH = gos3.calcEdatHuma();  
        System.out.println(edatH);  
    }  
}
```

```
class Gos{  
  
    String raca;  
    int edat;  
  
    Gos () {  
    }  
  
    Gos (String raca, int edat){  
        this.raca = raca;  
        this.edat = edat;  
    }  
    int calcEdatHuma(){  
        return edat * 7;  
    }  
}
```