Introducció al Java

PROP 2020-21 Q1

Mario Martin – Ignasi Gómez – Jordi Urmeneta – Bernat Orellana

Característiques

- Orientat a Objectes
- Sintaxi procedural similar al C
- Multiplataforma
- No compilat

NetBeans

- Entorn per desenvolupar aplicacions en Java
- Eina d'ajuda que minimitza errors
- Dona documentació de les classes

- RECOMANABLE però no imprescindible

Característiques

- **Orientat a Objectes**
- Sintaxi procedural similar al C
- Multiplataforma
- No compilat

Orientat a Objectes

- Tres elements:
  - Classes: Descripció de classes genèriques
  - Objectes: Instancies particulars de les classes
  - Aplicacions: Programa principal que utilitza objectes
- Veurem conceptes relacionats a la POO avançada
  - Encapsulació
  - Herència (sobre-escriptura, sobre-càrrega,...)
  - Polimorfisme
  - Descripció de interfícies
  - ....

Classes

- Descriuen objectes amb propietats comuns (és com una plantilla)
- Una classe té:
  - Atributs
    - Característiques que defineixen l'objecte
    - Diferencia objectes de la mateixa classe
    - Defineixen l'estat de l'objecte
    - Poden ser variables o mètodes que calculin els valors
  - Mètodes: Accions que es poden dur a terme amb les dades de l'objecte.

Objectes i Classes

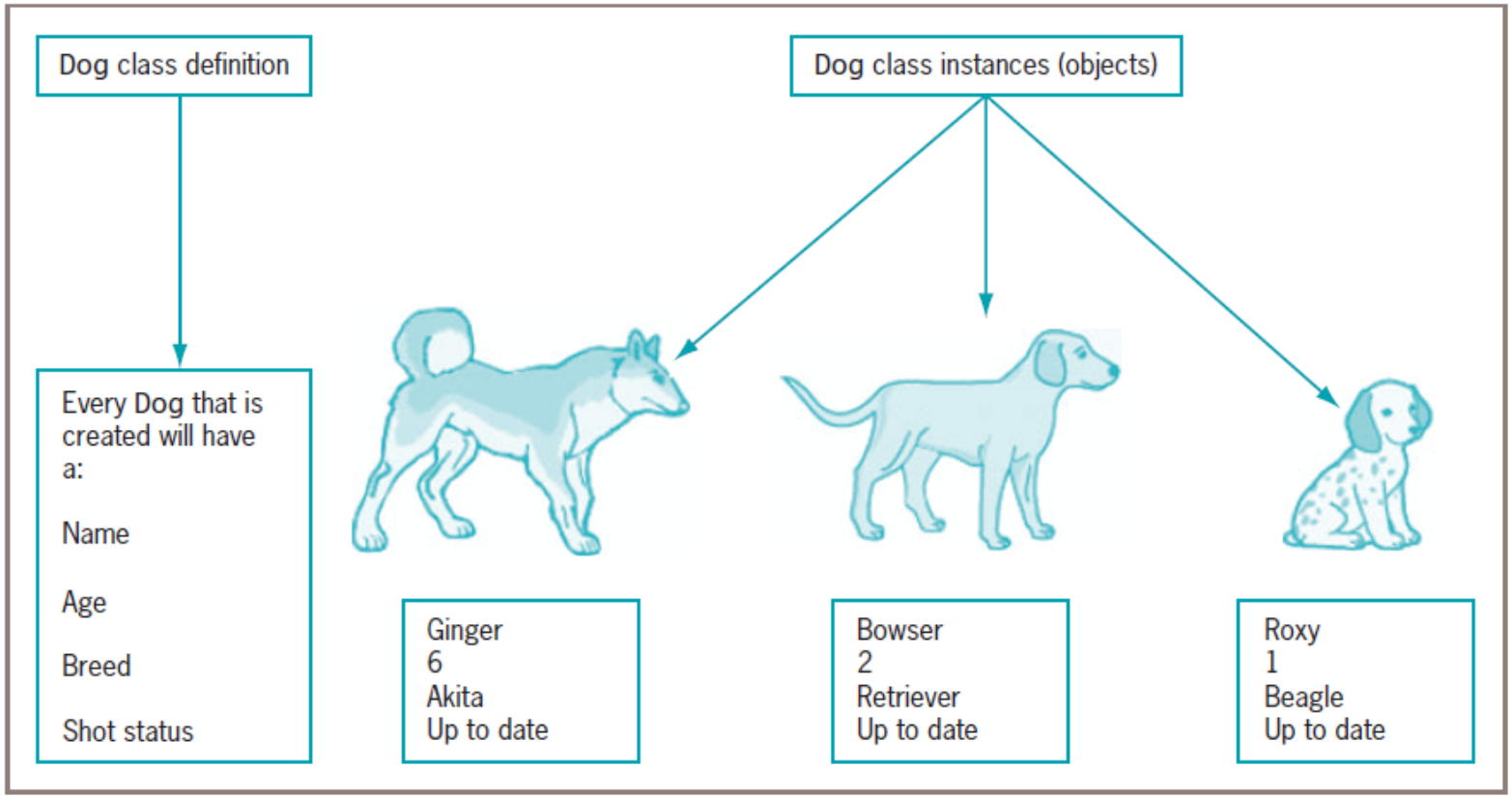


Figure 1-2 A class definition and some objects created from it

Objectes

- Objectes
  - Instàncies especifiques i concretes de les classes
  - Per crear un objecte usem l'operador

    new {nomClasse}()

    , que crea una instància de la classe indicada. L'operador new pot requerir de paràmetres segons el cas:

```
// creació d'un objecte de la classe Integer
Integer i  = new Integer(10);

// creació d'un objecte de la classe Persona
Persona maria = new Persona( 10, "11111111H", "Maria", "Gómez Soler" );
```

Aplicacions

- Tot en Java ha de pertànyer a una **classe**. Les aplicacions s'implementen com el mètode *static main* en les classes
- Permet execució procedural de codi

```
public class AnyClassName
{
    public static void main(String[] args)
    {
        /******/
    }
}
```

Aplicacions en una classe

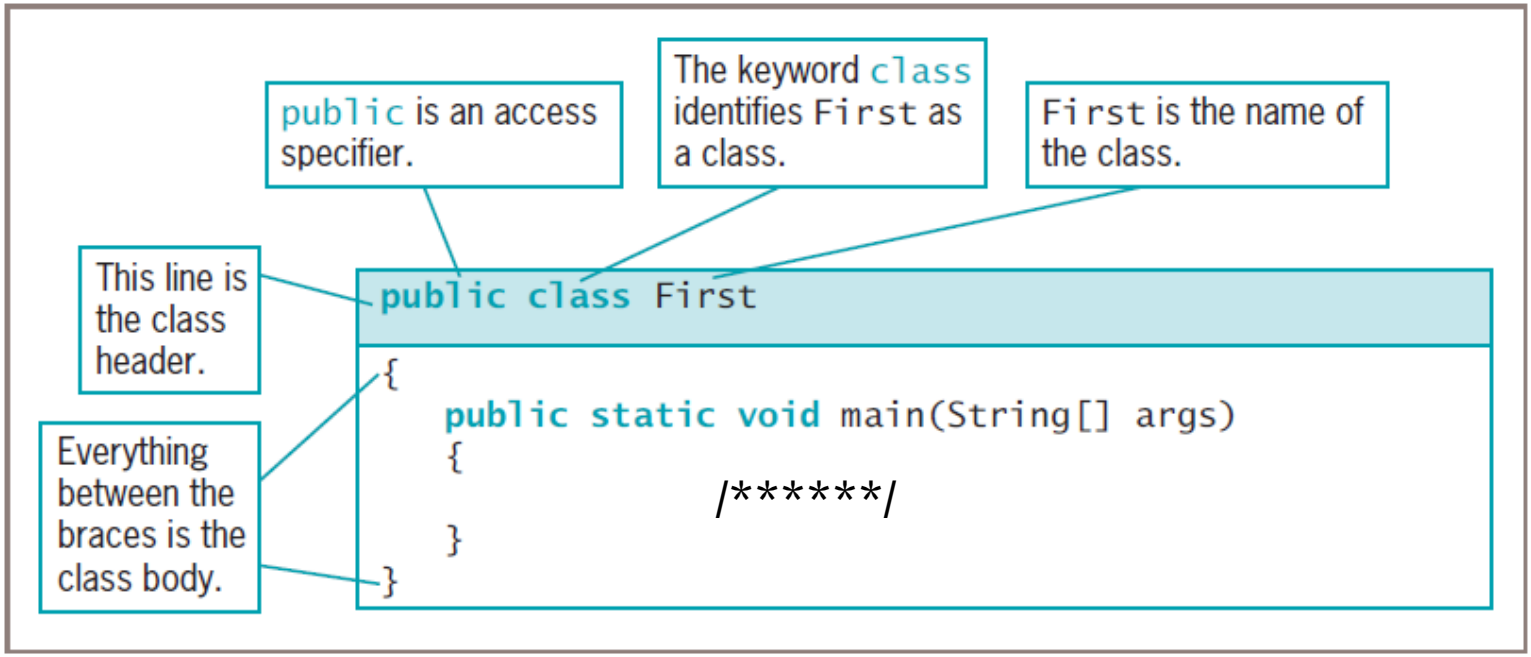


Figure 1-6 The parts of a typical class

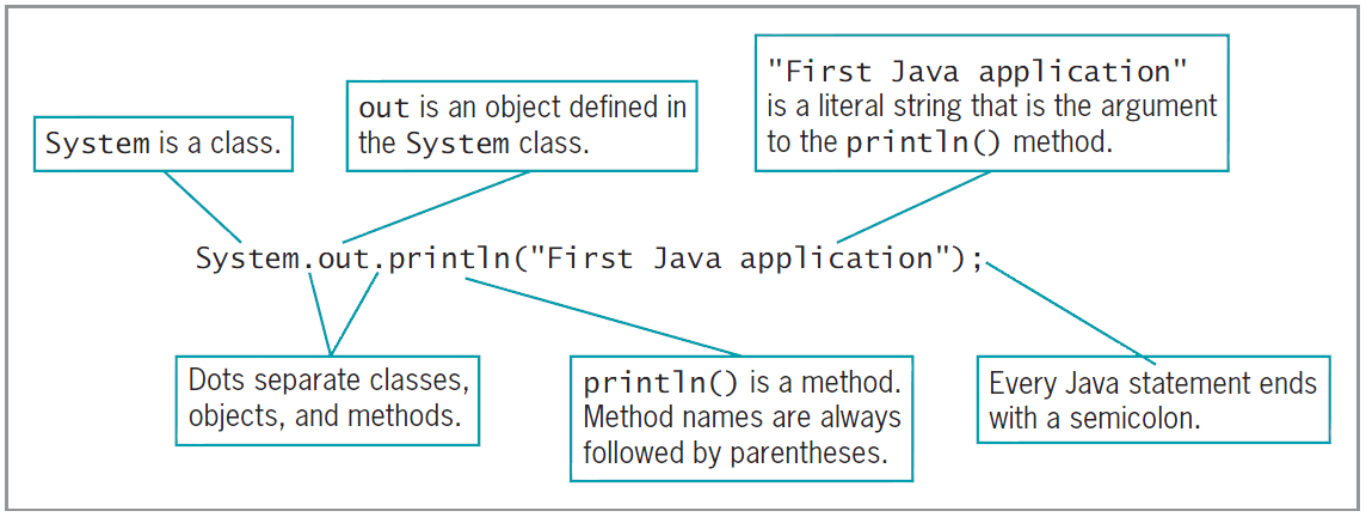
main()

- `static`
  - Significa que (1) és accessible i usable encara que no existeixin objectes de la classe i (2) no es lligat als objectes de la classe.
  - És també la manera de declarar ***funcions*** no lligades a classes
- `void`
  - Indica que el mètode (main) no retorna res.

```
public class AnyClassName
{
   public static void main(String[] args)
   {
      /******/
   }
}
```

Exemple main()

```
public class First
{
   public static void main(String[] args)
   {
      System.out.println("First Java application");
   }
}
```



Característiques

- Orientat a Objectes
- Sintaxi procedural similar al C
- Multiplataforma
- Combina compilació i interpretació.

Característiques

- Orientat a Objectes
- **Sintaxi procedural similar al C**
- Multiplataforma
- Combina compilació i interpretació.

Característiques

- Orientat a Objectes
- **Sintaxi procedural similar al C**
- Multiplataforma
- Combina compilació i interpretació.

```
public static int residu(int dividend, int divisor) {
    while(dividend>=divisor) {
        dividend -= divisor;
    }
    return dividend;
}
```

Tipus de dades

- Numèrics (primitius)

Type	Storage	Min Value	Max Value
byte	8 bits	–128	127
short	16 bits	–32,768	32,767
int	32 bits	–2,147,483,648	2,147,483,647
long	64 bits	–9,223,372,036,854,775,808	9,223,372,036,854,775,807
float	32 bits	Approximately –3.4E+38 with 7 significant digits	Approximately 3.4E+38 with 7 significant digits
double	64 bits	Approximately –1.7E+308 with 15 significant digits	Approximately 1.7E+308 with 15 significant digits

Tipus de dades

- Booleans
  - Tipus boolean
  - Literals: `true, false`
- Caràcters
  - Tipus char
  - Literals: 'x'
- Strings
  - Classe String
  - Indexades amb base zero
  - Literals: "paraula"
  - **És immutable (** el valor no es modifica, per canviar una cadena sempre se'n crea una de nova**)**
  - Operador de concatenació: **+**
  - Múltiples mètodes de manipulació:
- Dates
  - Classe Date
  - Valors: Date d = new Date();

```
//  0123   Strings are 0-indexed
String name="Paco";
int l = name.length();              // 4
int i = name.indexOf('c');          // 2
char c = name.charAt(1);            // 'a'
String s = name.substring(1, 2);    // "ac"
```

Tipus de dades

- Classe String:

```
String (String str)
   Constructor: creates a new string object with the same characters as str.

char charAt (int index)
   Returns the character at the specified index.

int compareTo (String str)
   Returns an integer indicating if this string is lexically before (a negative return
   value), equal to (a zero return value), or lexically after (a positive return value),
   the string str.

String concat (String str)
   Returns a new string consisting of this string concatenated with str.

boolean equals (String str)
   Returns true if this string contains the same characters as str (including
   case) and false otherwise.

boolean equalsIgnoreCase (String str)
   Returns true if this string contains the same characters as str (without
   regard to case) and false otherwise.
```

Tipus de dades

`int length ()`
Returns the number of characters in this string.

`String replace (char oldChar, char newChar)`
Returns a new string that is identical with this string except that every occurrence of `oldChar` is replaced by `newChar`.

`String substring (int offset, int endIndex)`
Returns a new string that is a subset of this string starting at index `offset` and extending through `endIndex-1`.

`String toLowerCase ()`
Returns a new string identical to this string except all uppercase letters are converted to their lowercase equivalent.

`String toUpperCase ()`
Returns a new string identical to this string except all lowercase letters are converted to their uppercase equivalent.

Arrays

- Definicions:

```
tipus[] nomtaula = new tipus[entermida];
```

```
tipus[][] nommatriu = new tipus[entermida1] [entermida2];
```

```
int notes[] = new int[10];
notes[0] = 6;
int notaSegonAlumne = notes[1];
for(int i=0;i<notes.length;i++) {
    notes[i]=i;
}
```

Sintaxi procedural

- Condicionals:

```
if (cond) {
 ....
}
else {
...
}
```

```
if (cond1) { ...}
else if (cond2) {...}
else if (cond3) {....}
....
else {....}
```

- Iteracions:

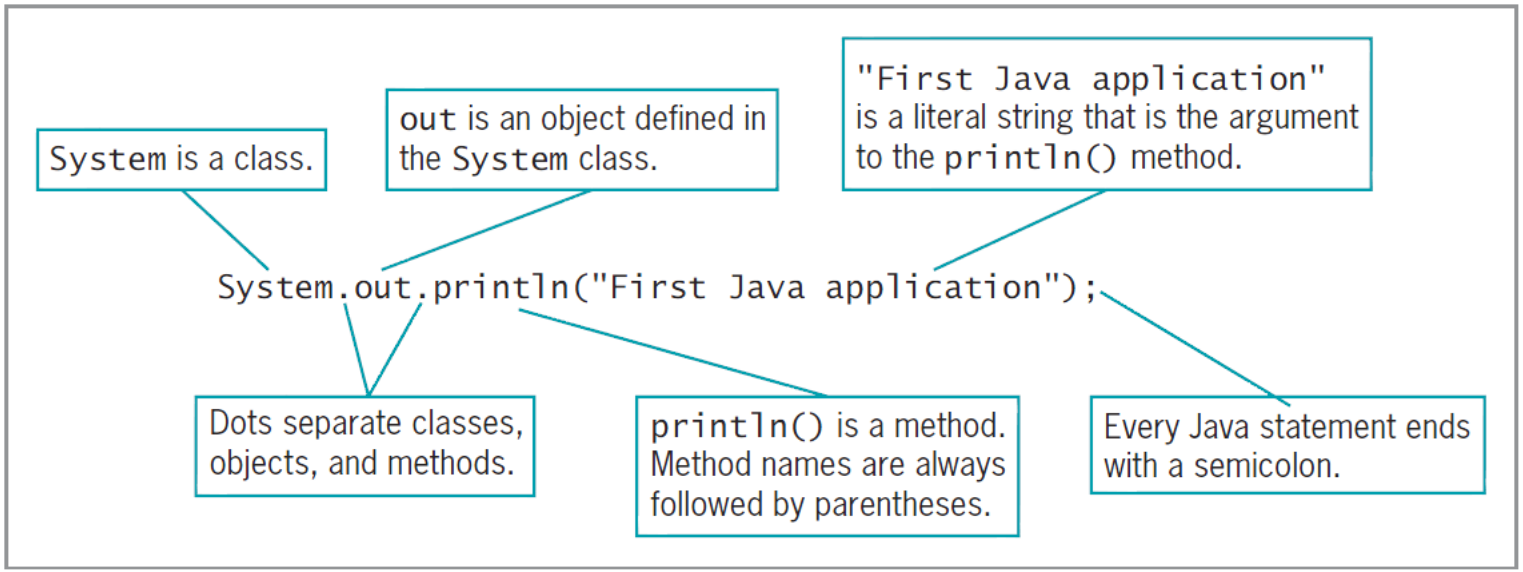
```
while (cond) {
 ....
}
```

```
for (int var= ini; var<fin; var++){
...
}
```

```
do {
...
} while (cond);
```

Entrada/Sortida

- Print:



Entrada/Sortida

- Classe Scanner.
  - Mètodes:

```
Scanner (InputStream source)
Scanner (File source)
Scanner (String source)
```
Constructors: sets up the new scanner to scan values from the specified source.

```
String next()
```
Returns the next input token as a character string.

```
String nextLine()
```
Returns all input remaining on the current line as a character string.

Entrada/Sortida

```
boolean nextBoolean()
byte nextByte()
double nextDouble()
float nextFloat()
int nextInt()
long nextLong()
short nextShort()
```

Returns the next input token as the indicated type. Throws `InputMismatchException` if the next token is inconsistent with the type.

```
boolean hasNext()
```

Returns true if the scanner has another token in its input.

Entrada/Sortida

```
Scanner useDelimiter (String pattern)
Scanner useDelimiter (Pattern pattern)
        Sets the scanner's delimiting pattern.

Pattern delimiter()
        Returns the pattern the scanner is currently using to match delimiters.

String findInLine (String pattern)
String findInLine (Pattern pattern)
        Attempts to find the next occurrence of the specified pattern, ignoring delimiters.
```

- Creació objecte per fer lectura de teclat

```
Scanner scan = new Scanner (System.in);
```

Entrada/Sortida

- Exemple:

```
import java.util.Scanner;

public class GetUserInfo
{
    public static void main(String[] args)
    {
        String name;
        int age;
        Scanner inputDevice = new Scanner(System.in);
        System.out.print("Please enter your name >> ");
        name = inputDevice.nextLine();
        System.out.print("Please enter your age >> ");
        age = inputDevice.nextInt();
        System.out.println("Your name is " + name + " and you are " + age + " years old.");
    }
}
```

Package

- Els *package* permeten organitzar les classes en estructures d'arbre.
- El *package* és com un nom de domini invertit al que pertany la classe. Per exemple:

```
package edu.upc.epsevg.prop.lab.sessio_0;
public class Persona {}
```

- Si no poseu package, s'assumeix el package per defecte (buit)
- El nom complet de la classe es composa del nom del *package* concatenat amb el nom de la classe.
  - edu.upc.epsevg.prop.lab.sessio_0.Persona

Característiques

- Orientat a Objectes
- Sintaxi procedural similar al C
- Multiplataforma
- **Combina compilació i interpretació.**

Característiques

- Orientat a Objectes
- Sintaxi procedural similar al C
- **Multiplataforma**
- **Combina compilació i interpretació.**

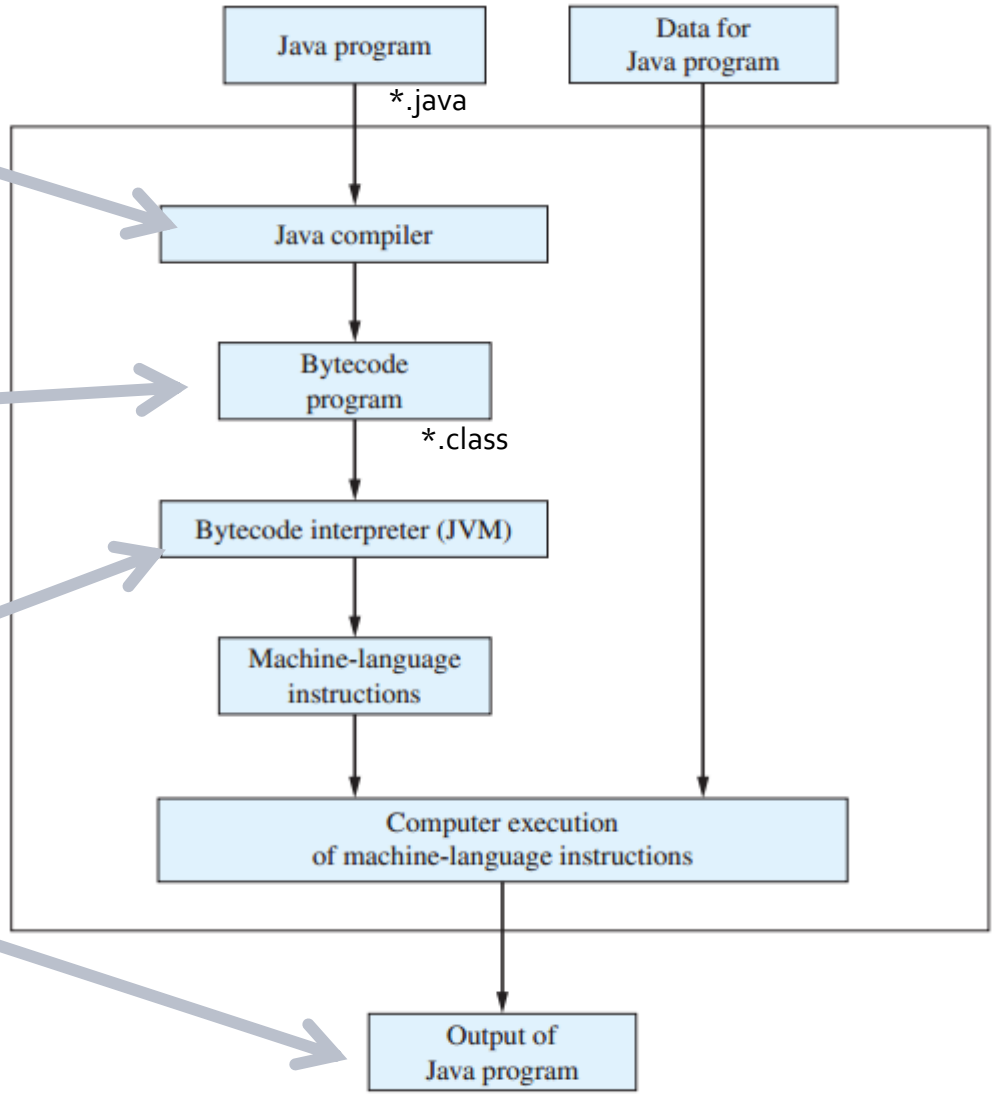
Combina compilació i interpretació.

```
>javac Salutacio.java
```

```
C:\Users\Hume\Desktop\PROP Java>javac Salutacio.java

C:\Users\Hume\Desktop\PROP Java>ls Salutacio*
Salutacio.class  Salutacio.java
```

```
C:\Users\Hume\Desktop\PROP Java>java Salutacio
Saluton mondo!
```



Exercicis (1)

- Llegir un sencer, N, pel teclat.
- Llegir N paraules pel teclat, **sense repeticions**. Si es repeteix una paraula s'ignora.
- Mirar quines d'aquestes parelles de paraules son "encastables" (totes les parelles possibles) i quina paraula en sortiria d'encastar-les. Dos paraules són encastables quan existeix un sufix d'una que al temps és prefix de l'altre. Per exemple:
  - Gat i tortuga són *encastables* i en sortiria "tortugat"
    - Un altra opció es combinar-les al revés per tenir "gatortuga"
  - Timbal i Taula no ho són
- La sortida del programa seran totes les paraules encastades (les dues possibilitats en cada cas), una per línia.
- El programa **no** ha de ser case-sensitive.

Exercicis (2)

- Llegir un sencer, N, pel teclat.
- Llegir una matriu NxN de l'entrada estàndard, usant un format:
  - numero, numero,..... numero [enter$_0$]
  - numero, numero,..... numero [enter$_1$]
  - ...
  - numero, numero,..... numero [enter$_{N-1}$]
- Mostreu per sortida estàndard la matriu transposada.
- Mostreu per sortida estàndard si la matriu és simètrica (igual a la seva transposada)
- Mostra per la sortida estàndard si la matriu és ortogonal → la seva inversa coincideix amb la seva transposada. → $M*M^T = I$

$$\begin{bmatrix} -3 & 1 & 5 \\ 1 & 0 & -2 \\ 5 & -2 & 4 \end{bmatrix}$$

Matriu simètrica

$$\begin{bmatrix} \frac{2}{3} & \frac{1}{3} & \frac{2}{3} \\ -\frac{2}{3} & \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} & -\frac{2}{3} \end{bmatrix}$$

Matriu ortogonal