


Programació



UT4.2 Declaració i invocació de
mètodes. Variables locals i globals



Mètodes

- En Java, les funcions/procediments s'anomenen **mètodes**
- Arribats a este punt, es pot ara comprendre millor que "**main**" és el **mètode principal** d'un programa, i que de moment ha sigut l'únic mètode definit per nosaltres.
- Quan es parla d'**invocació d'un mètode** (per exemple de tractament d'Strings), es tracta d'**executar un conjunt d'instruccions amb un objectiu comú**: "obtindre una cadena transformada", "obtindre una subcadena" ...

Declaració de mètodes

- Igual que les variables, els **mètodes també s'han de definir per a poder ser usats**. De moment optarem per definir un mètode de la següent manera:

```
public void nomMetode() {  
    //Aquí dins aniran les seves instruccions  
    //...  
}
```

- Esta declaració pot realitzar-se en qualsevol punt del programa, dins del bloc de la classe on es defineix (**public class NomClasse{...}**) sempre fora del mètode main.

Declaració de mètodes

```
public class OrdenarDescendent {  
    public static void main(String[] args) {  
        //Instruccions del mètode principal (problema general)  
        //...  
    }  
    //Mètode que resol el subproblema de llegir la llista.  
    public void llegirLlista() {  
        //Instruccions del mètode  
        //...  
    }  
    //Mètode que resol el subproblema d'ordenar la llista.  
    public void ordenarLlista() {  
        //Instruccions del mètode  
        //...  
    }  
    //Mètode que resol el subproblema de mostrar la llista per pantalla.  
    public void mostrarLlista() {  
        //Instruccions del mètode  
        //...  
    }  
}
```


Canvis al mètode principal

- Aplicarem un canvi a l'hora de desenvolupar qualsevol programa. Seguirem (de moment) esta estructura:

Nom de la classe

```
public class OrdenarDescendent {  
    public static void main (String[] args) {  
        //Aquí cal usar el nom de la classe que esteu creant.  
        OrdenarDescendent programa = new OrdenarDescendent();  
        programa.inici();  
    }  
    public void inici() {  
        //Instruccions del mètode principal (problema general)  
        //...  
    }  
    //Resta de mètodes  
    //...  
}
```


Àmbit de variables

- En este punt, **reprèn més força el concepte d'àmbit d'una variable**, ja que estem definint blocs a través de cada mètode.
- Cal tindre clar dos conceptes:
 - **Variable local**, és aquella que es restringeix al àmbit de la funció (mètode) on s'ha declarat.
 - **Variable global**, és aquella que serà accessible des de qualsevol funció del programa.

Variables globals


- La manera en la que es definirà una variable global serà la següent:

1) Utilitzant esta sintaxi:


```
private tipus nomVariable = valorInicial;
```

I ubicant SEMPRE esta declaració **just en la línia següent al bloc que defineix la classe del programa.**

Variables globals: Exemple



```
public class OrdenarDescendent {  
    //Variable global  
    private int[] llistaEnters = new int[10];  
    public static void main (String[] args) {  
        OrdenarDescendent programa = new OrdenarDescendent();  
        programa.inici();  
    }  
    public void inici() {  
        //Instruccions del mètode principal (problema general)  
        //...  
    }  
    //Mètode amb les instruccions per llegir la llista.  
    public void llegirLlista() {  
    }  
    //Mètode amb les instruccions per ordenar la llista.  
    public void ordenarLlista() {  
    }  
    //Mètode amb les instruccions per mostrar la llista per pantalla  
    public void mostrarLlista() {  
    }  
}
```



Mètode ordenarLlista()

```
//Mètode amb les instruccions per ordenar la llista.  
public void ordenarLlista() {  
    for (int i = 0; i < llistaEnters.length - 1; i++) {  
        for(int j = i + 1; j < llistaEnters.length; j++) {  
            //La posició tractada té un valor més alt que el de la cerca... Els  
            intercanviem.  
            if (llistaEnters[i] > llistaEnters[j]) {  
                //Per intercanviar valors cal una variable auxiliar  
                int canvi = llistaEnters[i];  
                llistaEnters[i] = llistaEnters[j];  
                llistaEnters[j] = canvi;  
            }  
        }  
    }  
}
```


Pràctica 1

- **Còpia el programa** d'ordenació de llistes d'enters.
- **Completa el programa**, dissenyant i implementant els mètodes **llegirLlista()** i **mostrarLlista()**. De moment no sabem provar-ho per a vore si funciona.
- Realitza una traça manual amb un parell d'exemples per a verificar el seu funcionament.

Invocació de mètodes

- Per a poder executar totes les operacions definides a un mètode, cal invocar-lo.
- Per a invocar (cridar) un mètode s'utilitza la següent sintaxi:

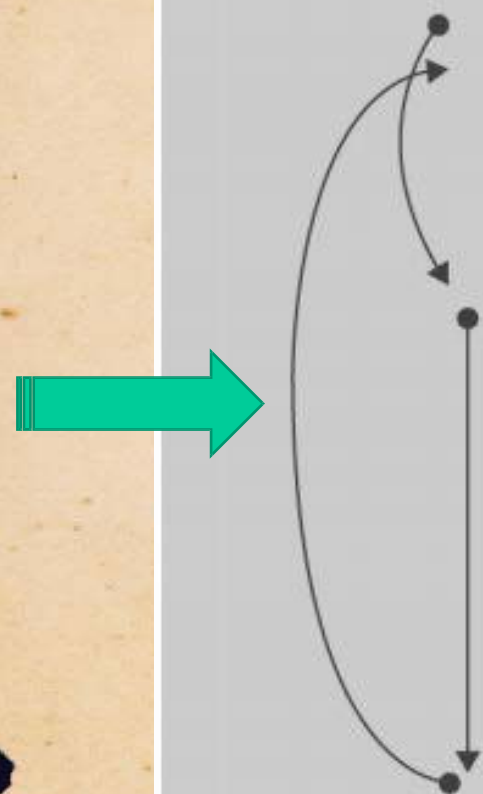
```
nomMètode ( ) ;
```


Invocació de 3 mètodes.

Exemple:

```
public void inici() {  
    llegirLlista();  
    ordenarLlista();  
    mostrarLlista();  
}
```


Flux de control en la invocació del primer mètode



```
public void inici() {  
    llegirLlista();  
    ordenarLlista();  
    mostrarLlista();  
}
```

```
public void llegirLlista() {
```

```
    // ... (blurred code) ...
```

```
    }  
    lector.nextLine();  
}
```


Pràctica 1 (continuació)

- Actualitza el programa, i ara sí, prova el programa *in situ*, per a veure si l'eixida és correcta.
- En cas d'existir algun problema, rectifica els mètodes implementats per a que funcione com cal.

Pràctica 2.

- Modifica el programa d'exemple perquè faci el següent:
 - Després de mostrar la llista ordenada, en una nova línia, ha de dir **quants dels valors són inferiors** a la meitat del valor més gran emmagatzemat.
 - Aplica **disseny descendent** per afegir esta nova tasca, declarant i invocant els nous mètodes que calguen.
 - Recorda que pots utilitzar variables globals.

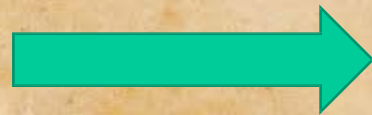
Inicialització diferida de variables

- Com ja saps, una variable pot ser declarada i inicialitzada al mateix temps o bé a posteriori.
- Per al cas dels tipus de dades primitius, el valor de inicialització provisional sol ser **0**, si és de tipus numèric.
- Per als tipus de dades compostos, s'utilitzarà com a valor d'inicialització provisional el valor especial **null**.
- Invocar a un mètode d'una variable de tipus compost que siga null sempre derivarà en un **error del programa**.

Inicialització a null

```
String missatge;
```

```
missatge.length();
```



ERROR

Pràctica 3

- A partir del programa de la pràctica 1, inicialitza ara el array d'enters a "null".
- El programa ara preguntarà a l'usuari, quants números vol introduir.
- En tenir esta dada, crea l'array amb la mida adequada

El programa ha de funcionar igual, excepte augmentar/disminuir el nombre de dades a tractar.