UEF 1 : Informatique & Programmation

Faculté des Sciences de Nice

DEUG 2000-2001

Jérôme DURAND-LOSE

Sandrine JULIA

Jean-Paul ROY

COURS 3

Les « classes »

et leurs

« méthodes de classe »

L'unité de programme Java est la *classe* !

```
/* fichier Essai1.java */

class Essai1
{  public static void main(String[] args)
   {  System.out.println("Salut la fac !");
   }
}
```

*la **méthode** principale **main(...)** de la classe **Essai1***

Qu'est-ce que programmer en Java ?

- C'est rédiger avec précision un **ensemble de classes** qui vont coopérer à la résolution d'un problème.
- Parfois même **une seule classe**. Par exemple, la classe `Essai1` précédente, qui ne contenait qu'une seule méthode, la méthode `main(…)`.
- En général, une classe contiendra **plusieurs méthodes**. Chaque méthode est une portion de la classe spécialisée dans une tâche bien déterminée.
- Dans ce cours : **une seule classe par fichier** !

Il y a beaucoup de classes prédéfinies !...

♦ Par exemple, la classe `Math` avec ses méthodes de calcul, avec la constante `PI`, etc.

```
public class Math
{  public static final double PI = 3.14159…;

   public static double sqrt(double x)
   {. . . . .}

   . . .
}
```

la fonction x --> √x
double --> double


```
double deuxPi = 2 * Math.PI;
double rac2 = Math.sqrt(2);
```

Envoi d'un message...

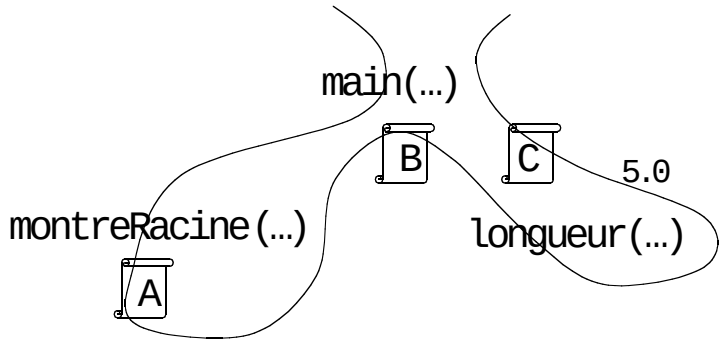
invocation de la méthode `sqrt(…)` de la classe `Math` sur l'argument 2

♦ On ne peut pas demander seulement `sqrt(2)`…

♦ Il faut s'adresser à la classe elle-même : il s'agit d'une *méthode de classe*.

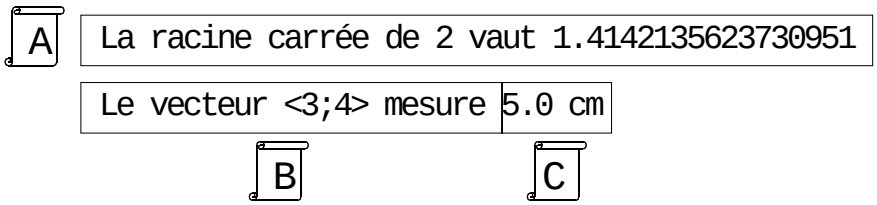
Mot-clé : `static`


```
class Essai2
{  public static void main(String[] args)
   {  montreRacine(2);
      System.out.print("Le vecteur <3;4> mesure ");
      System.out.println(longueur(3,4) + " cm");
   }

   static void montreRacine (int n)
   {  double racine = Math.sqrt(n);
      System.out.print("La racine carrée de " + n);
      System.out.println(" vaut " + racine);
   }

   static double longueur (double x, double y)
   {  return Math.sqrt(x*x + y*y);
   }
}
```

♦ Exécution du programme après compilation et production du fichier `Essai2.class` :

```
La racine carrée de 2 vaut 1.4142135623730951
Le vecteur <3;4> mesure 5.0 cm
```



♦ La méthode principale `main(…)` est obligatoire si cette classe est destinée à être exécutée comme classe principale. Elle utilise les deux autres méthodes statiques [inutile de les préfixer par le nom de la classe courante] :

obligatoire pour `main(…)` *!*

méthode de classe

sans résultat

inutilisé…

```
public static void main(String[] args)
   {  montreRacine(2);
      System.out.print("Le vecteur <3;4> mesure ");
      System.out.println(longueur(3,4) + " cm");
   }
```

Essai2.montreRacine(2);

Essai2.longueur(3,4)


```
class Essai2
{
   public static void main(String[] args)
   ...

   static void montreRacine (int n)
   ...

   static double longueur (double x, double y)
   ...
}
```


♦ La méthode `longueur(…)` se comporte comme une fonction : *double x double* ---> *double* :

méthode de classe

dont le résultat est un double

```
static double longueur (double x, double y)
{  return Math.sqrt(x*x + y*y);
}
```

elle « retourne » un résultat…

elle attend deux arguments de type « `double` *»*

♦ La méthode `montreRacine(…)` n'a pas de résultat à retourner ! Elle retourne « `void` » [*du vent*]...

méthode de classe sans résultat

elle attend un argument de type « `int` »

```
static void montreRacine (int n)
{  double racine = Math.sqrt(n);
   System.out.print("La racine carrée de " + n);
   System.out.println(" vaut " + racine);
}
```

Aucun « `return` » *!*

FONCTIONS et PROCEDURES

Par convention, nous dirons que :

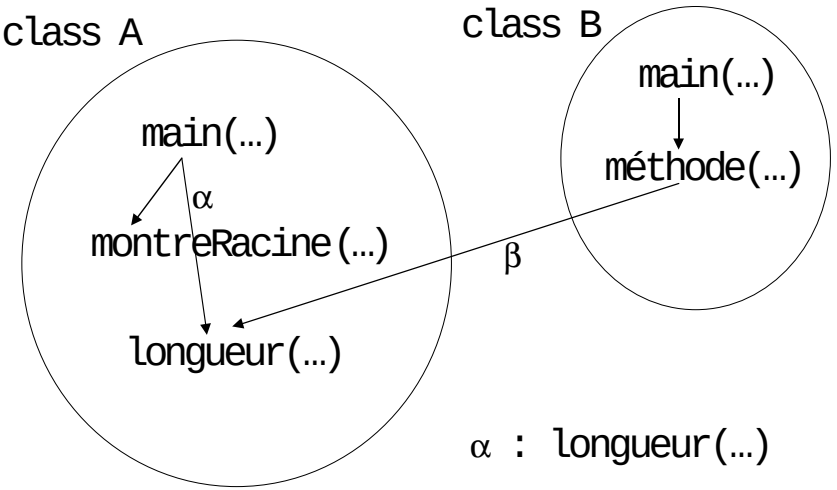
♦ La méthode `longueur(…)` est une FONCTION dont la *signature* est : *double* x *double* --> *double*

♦ La méthode `montreRacine(…)` est une PROCEDURE dont la *signature* est : *int* --> *void*

♦ Une FONCTION se borne à calculer et à retourner un résultat : c'est une « boîte noire » !

♦ Au contraire, une PROCEDURE agit, elle a des effets résiduels, mais ne retourne aucun résultat !

♦ Activation d'une méthode *static* d'une classe A depuis l'intérieur d'une classe B du même répertoire :



α : `longueur(…)`

β : `A.longueur(…)`


```
class ActivationEssai2
{  public static void main(String[] args)
   {  System.out.println("Allo Essai2 ?");
      Essai2.montreRacine(3);
      System.out.println("Ok, bien reçu.");
   }
}
```

Nécessité de préfixer par Essai2

```
Allo Essai2 ?
La racine carrée de 3 vaut 1.732...
Ok, bien reçu.
```

♦ *MORALE : se constituer des bibliothèques de classes ré-utilisables. Soyez « modulaires » !*