# R2.01 : cours n°3

I.Borne

- La surcharge des méthodes
- Données constantes
- Utiliser des packages standards
- Tableau des principaux packages
- Utilisation de `java.util.Scanner`
- Utilisation de la classe `Math`



# Rappels des concepts déjà vus

- Tout objet est instance d'une classe.
- Une classe définit les attributs et les méthodes des objets qu'elle représente.
- Les objets sont des références.
- Les objets communiquent par envois de messages
- Les types primitifs ne sont pas des objets



# La surcharge

- La surcharge d'une méthode ou d'un constructeur permet de définir plusieurs fois une méthode ou un constructeur avec des arguments différents dans une même classe.

- La méthode qui sera appelée le sera en fonction du nombre et du type des paramètres.



# Surcharge d'un constructeur

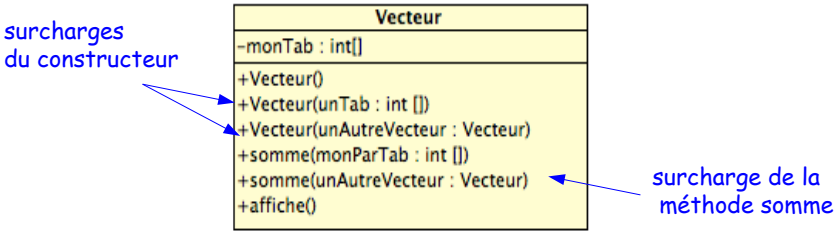
```
public class Manager {
  private String nom;
  private String prenom;

  public Manager(String nom, String prenom) {
    this.nom = nom;
    this.prenom = prenom;
  }

  public Manager(String nom) {
    this.nom = nom;
    this.prenom = "Richard";
  }
  public Manager() {
    this.nom = "Coeur de Lion";
    this.prenom = "Richard";
  }
}
```

C'est une illustration,
cela ne signifie pas qu'il faut
faire de telles initialisations !

# Exemple de la classe Vecteur






# Retour sur le concept de variable

- Une variable est une donnée (un objet ou un type primitif) repérée par son nom, et qui pourra être lue ou modifiée lors de l'exécution du programme.
- En Java les variables sont typées.
- Un nom de variable ne peut pas commencer par un chiffre et ne doit comporter que des lettres et des chiffres.
- Un nom de variable ne peut pas être un nom réservé du langage (boolean, if, else, true, false, void etc ...)



# Le concept de constante

- Une constante est une donnée dont la valeur ne doit pas changer lors de l'exécution du programme.
- En Java on ne peut véritablement parler de constantes que pour les types primitifs.
- La référence d'un objet peut ne pas être modifiable mais les attributs de l'objet peuvent être modifiés malgré tout.



# Les données constantes

- Quand une donnée ne doit pas changer de valeur alors on l'appelle une constante.
- Une constante doit être initialisée avec une valeur et ne peut pas être modifiée ensuite par le programme.
- En Java les constantes sont définies avec le mot-clé `final`
- Dans une définition de classe un attribut constant sera déclaré « final » :

```
public class MaClasse {
    private final double VOLUME = 1,5 ;
...
}
```

Lisibilité : les noms des constantes doivent être en MAJUSCULES

# Utilisation des attributs constants

- La valeur initiale ne pourra pas être changée
  - soit elle est initialisée à la déclaration (vue précédente)
  - soit elle est initialisée dans le constructeur

```
public class MaClasse {
    private final double VOLUME;
    public MaClasse(){
         VOLUME = 1,5 ;
    } ...}
```

Maintenabilité : les constantes doivent être utilisées à la place des valeurs littérales dans un programme. Ceci rendra le programme plus facile à modifier et à maintenir.

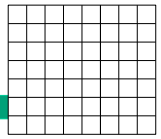


# Exemple de donnée constante

```
public class Cercle {
   private double r; //le rayon
   private final double PI = 3.1416;
   ...
    public double getRayon() {
       return this.r;
   }
   public double circonference() {
       double res;
       res = 2* PI * this.getRayon();
       return res;
   }
}
```



# Tableau multi-dimentions et constante



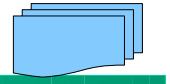
On veut faire des relevés météo journaliers des précipitations dans un premier temps sur une année.

Une année de précipitations sera représentée par un tableau à 2 dimensions à 12 lignes (mois) et 31 colonnes (jours) :

```
final int NDAYS = 31 ;
final int NMONTHS = 12 ;
double rainfall[] [] = new double[NMONTHS][NDAYS];
// initialisation du tableau
for (int month=0 ; month<rainfall.length ; month++){
     for (int day=0 ;
               day < rainfall[month].length ;
               day ++) {
                       rainfall[month][day] = 0.0 ;
     }
}
```



# Exemple à 3 dimensions



- On veut conserver les relevés sur 10 ans, un tableau à 3 dimensions est utilisé (une collection de pages) :

```
final int NYEARS = 10 ;
final int NMONTHS = 12 ;
final int NDAYS = 31 ;

double rainfall[][][]= new double[NYEARS][NMONTHS][NDAYS] ;
// initialisation du tableau
for (int year =0 ; year < rainfall.length ; year++){
     for (int month=0; month < rainfall[year].length ;month++){
           for(int day=0; day < rainfall[year][month].length;
                    day ++){
               rainfall[year][month][day] = 0.0 ;
           }
     }
}
```

# Utilisation des packages standards

- Toutes les classes de l'API de Java sont organisées en packages :
  - **`java.lang`** : classes fondamentales du langage (pas d'import).
  - **`java.util`** : classes utilitaires et collections,
  - **`java.io`** : classes gérant les entrées/sorties
  - **`java.awt`** : classes pour les interfaces graphiques
- Pour ne pas être obligé d'utiliser le nom complet des classes, il faut importer les noms de classe avec leur package dans la classe où l'on veut l'utiliser.

```
import java.util.Date ;
```

Permet ensuite de référencer le nom de la classe juste par **`Date.`**

- Les classes du package `java.lang` ne sont pas à importer, elle sont déjà présentes dans l'environnement de base.



# Les principaux packages de la bibliothèque Java

| Package | Sujet | Exemple de classes |
|---|---|---|
| java.lang | Language support | Math |
| java.util | Utilities | Scanner, ArrayList |
| java.io | Input, Output | PrintStream |
| java.awt | Abstract Windowing Toolkit | Color |
| java.applet | Applets | Applet |
| java.net | Networking | Socket |
| java.sql | Database access | ResultSet |
| javax.swing | Swing User interface | JButton |



# La classe `java.util.Scanner`

- Cette classe modélise un scanner de texte simple qui peut analyser des types primitifs et des chaînes de caractères.
- Un objet Scanner coupe les entrées en *token* en utilisant un pattern de délimiteur.
- Par défaut il utilise les espaces pour obtenir les tokens.
- Les tokens sont convertis en des valeurs des différents types en utilisant les méthode *next*
- Si on ne donne pas le bon type de donnée à lire, une erreur d'exécution se produit.



# Création d'un objet Scanner

- La classe Scanner se trouve dans le package util de Java :

```
java.util.Scanner
```

- Il faut donc importer la classe pour pouvoir l'utiliser sans son nom complet dans son application :

```
import java.util.Scanner ;
```

- Le constructeur prend en paramètre le flux d'entrée (File, InputStream) à lire ou une chaîne de caractère.
- Par exemple pour créer un scanner qui pourra lire ce que vous tapez au clavier

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

`System.in` dénote le flux d'entrée standard

# Les principales méthodes de lecture

`public int nextInt ()`

analyse le prochain token en entrée comme un `int`

`nextDouble, nextFloat, nextBoolean` (similaires)

`public String nextLine ()`

retourne le reste de la ligne courante. Cela permet de lire une ligne de texte.

`public String next ()`

cherche et retourne le prochain token complet. Cela permet de lite un mot par exemple



# Les méthodes de vérifications

- Avant de lire une donnée il peut être prudent dans certains cas de vérifier s'il reste un token à lire, en particulier quand le lecture est celle d'un fichier.
- `public boolean hasNext ()`: retourne vrai si un autre token est en entrée.
- De même on peut tester s'il reste un nombre, une chaîne ou un ligne à lire : `hasNextInt(), …, hasNextLine()`

- Fermeture du Scanner :

  `public void close ()`



# Exemple de lecture au clavier

```
import java.util.Scanner;
public class TestLecture {
 public static void main (String[] args) {
  //Création du scanner sur le flux d'entrée standard
   Scanner in = new Scanner(System.in);
   System.out.println("Entrer la quantité :" );
   int quantite = in.nextInt();  //un entier
   System.out.println("Entrer le prix :" );
   double prix= in.nextDouble();  //un réel
   System.out.println("Entrer le code postal :" );
   String cp= in.next();    // un mot
 }
}
```



# Exemple tiré de l'API Java pour Scanner

https://docs.oracle.com/javase/7/docs/api/

```
String input = "1 fish 2 fish red fish blue fish";
Scanner s = new Scanner(input).useDelimiter("\\s*fish\\s*");
System.out.println(s.nextInt());
System.out.println(s.nextInt());
System.out.println(s.next());
System.out.println(s.next());
s.close();
```

imprimera sur l'écran :

```
1
2
red
blue
```

`fish`: mot délimiteur

`\\s*` un nombre quelconque d'espace

# La classe Math (java.lang.Math)

- Cette classe contient des méthodes pour effecteur les fonctions mathématiques usuelles : exponentielle, logarithme, racine carrée, fonctions trigonométriques …
- La classe Math s'utilise sans créer d'instance.
- C'est une classe utilitaire et toutes ses méthodes sont utilisées comme des messages envoyés à la classe.



# Quelques exemples avec Math

```
double x = 10.5 ;
// calcule la valeur absolue de x
double n = Math.abs(x) ;
// calcule de la fonction sinus(x)
n = Math.sin(x) ;
// nombre aléatoire : n>=0 et n<1.0
n = Math.random() ;
```

La constante Pi est disponible avec :

```
Math.PI
```