Où en est-on?

Nous avons vu comment déclarer des classes et des objets. On peut par exemple déclarer une instance de la classe Rectangle :

```
Rectangle rect;
```

Une fois que l'on a fait cette déclaration, comment faire pour donner aux attributs de `rect` *des valeurs autres que* `0.0`

```
class Rectangle {
  private double hauteur;
  private double largeur;

  public double surface()
    { return hauteur * largeur; }
  public double getHauteur()
    { return hauteur; }
  public double getLargeur()
    { return largeur; }
  public void setHauteur(double h)
    { hauteur = h; }
  public setLargeur(double l)
    { largeur = l; }
}
```

Initialisation des attributs

Première solution : **affecter individuellement une valeur** à chaque attribut

```
Rectangle rect;

System.out.println("Quelle hauteur ? ");
lu = clavier.nextDouble();
rect.setHauteur(lu);

System.out.println("Quelle largeur ? ");
lu = clavier.nextDouble();
rect.setLargeur(lu);
```

Ceci est une *mauvaise solution* dans le cas général :

- elle implique que tous les attributs fassent partie de l'interface (`public`) ou soient assortis d'un manipulateur ☞ casse l'encapsulation
- oblige le programmeur-utilisateur de la classe à initialiser explicitement tous les attributs ☞ risque d'oubli

Initialisation des attributs (2)

Deuxième solution : définir une **méthode dédiée à l'initialisation** des attributs

```
class Rectangle {
  private double hauteur;
  private double largeur;

  public void init(double h, double l)
  {
    hauteur = h;
    largeur = l;
  }
  //...
}
```

Pour faire ces initialisations, il existe en Java des méthodes particulières appelées **constructeurs**.

Les constructeurs

Un constructeur est une méthode :

- invoquée *systématiquement* lors de la déclaration d'un objet
- chargée d'effectuer toutes les opérations requises en « début de vie » de l'objet (dont *l'initialisation des attributs*)

Syntaxe de base :

```
NomClasse(liste_paramètres)
{
 /* initialisation des attributs
    en utilisant liste_paramètres */
}
```

Exemple :

```
Rectangle(double h, double l)
{
  hauteur = h;
  largeur = l;
}
```

Les constructeurs (2)

Les constructeurs sont des méthodes presque comme les autres. Les différences sont :

- *pas* de type de retour (pas même `void`)
- *même nom* que la classe
- invoqués *systématiquement* à chaque fois qu'une instance est créée.

```
Rectangle(double h, double l)
{
  hauteur = h;
  largeur = l;
}
```

Comme les autres méthodes :

- les constructeurs peuvent être surchargés

(exemples dans la suite)

Une classe peut donc avoir **plusieurs constructeurs**, pour peu que leur liste de paramètres soit différente.

Notre programme (1/3)

```
class Rectangle {
  private double hauteur;
  private double largeur;

  public Rectangle(double h, double l)
  {
    hauteur = h;
    largeur = l;
  }
  public double surface()
    { return hauteur * largeur; }
  // accesseurs/modificateurs si nécessaire
  // ...
}
```

Initialisation par constructeur

La *déclaration avec initialisation* d'un objet se fait selon la syntaxe suivante :

Syntaxe :

NomClasse instance = new *NomClasse*(*valarg1*, ..., *valargN*);

où *valarg1*, ..., *valargN* sont les valeurs des arguments du constructeur.

Exemple :

```
Rectangle r1 = new Rectangle(18.0, 5.3); // invocation du constructeur à 2 paramètres
```

Notre programme (2/3)

```
class Rectangle {
  private double hauteur;
  private double largeur;

  public Rectangle(double h, double l)
  {
      hauteur = h;
      largeur = l;
  }
  public double surface()
    { return hauteur * largeur; }
  // ...
}

class Exemple {
  public static void main(String[] args) {
    Rectangle rect1 = new Rectangle(3.0, 4.0);
      // ...
  }
}
```