# TP de PPO en JAVA

Polytech Lille GIS4 2018-2019

## 1 Prise en main de Java

Nous allons utiliser l'environnement standard JDK (Java Development Kit) qui comprend notamment les commandes de compilation `javac` et d'exécution `java` et de nombreuses bibliothèques (packages de classes) dont la document (la "javadoc") est sur :

`file:///usr/local/jdk1.8.0_25/docs/api/index.html`
Ouvrez ce fichier dans votre navigateur (firefox) (conservez ce lien dans vos bookmarks préférés). La documentation est entièrement navigable. A gauche, vous pouvez choisir une navigation alphabétique de "All Classes" ou d'un package particulier. A droite apparaissent les informations sur la sélection, notez les onglets :
- "Tree" : hiérarchie des packages et des classes de la sélection
- "Index" : index alphabétique de tous les symboles associés (classes, variables, méthodes, constructeurs, ...)

Observer comment sont décrites quelques classes vues en cours : `Object, String, Scanner, Applet, ...`

Ecrire le programme `echo` du cours. Compiler (`javac`), exécuter (`java`).

## 2 Rectangles

Travailler dans un répertoire ``rectangles''.

### 2.1 `java.awt.geom`

Programmer une classe `Rectangle` (dans un fichier `Rectangle.java`) telle que celle vue en cours :
- un rectangle est représenté par un couple de points "origine" (point supérieur gauche) et "corner" (point inférieur droit). On utilisera la classe `Point2D.Double` du package `java.awt.geom` (voir la javadoc).
- programmer les méthodes : `largeur(), longueur(), surface(), perimetre()`.
- programmer un constructeur paramétré par les coordonnées des 2 points origine et corner (voir le poly "Java" du cours, slides 13 et suivants).
- tester la classe `Rectangle` en programmant une classe principale `TestRectangle` (fichier `TestRectangle.java` dans le même répertoire) munie d'une méthode `main` qui instancie un rectangle de coordonnées fixées en utilisant le constructeur précédent, par exemple :

```
r = new Rectangle(10.0,10.0,40.0,50.0);
```

et affiche ses caractéristiques : largeur, longueur, surface, périmètre.

### 2.2 `toString()`

Programmer une méthode `toString()` dans la classe Rectangle qui renvoie sous forme de chaine de caractères son couple de points caractéristiques : ``(<origine> , <corner>)''. `<origine>`

et `<corner>` correspondent à la représentation sous forme de chaine de caractères des points renvoyée par leur propre méthode `toString()` (voir la documentation), laissez les s'afficher comme ils veulent !

Ajouter dans la classe `TestRectangle` l'affichage du rectangle par appel automatique à `toString()` : par exemple

## 2.3 Méthodes `static` et lectures au clavier

Pour rappel, les lectures au clavier se font sur l'entrée standard `System.in` en utilisant la classe "wrappeur" `Scanner`.

Modifier la classe `TestRectangle` en ajoutant une méthode `creerRectangle()` qui demande à l'utilisateur les coordonnées de 2 points origine et corner, instancie le rectangle correspondant et le retourne en résultat. Cette méthode devant être appelée par le `main` qui est `static` doit elle-même être `static` (même niveau) de la façon suivante :

```
public class TestRectangle {
 static Rectangle creerRectangle() {
  //saisie utilisateur et creation du rectangle
  ...
 }
 public static void main(String[] args) {
  Rectangle r;
  r = creerRectangle();
  ...
 }
}
```

## 2.4 Paramètres du `main`

Dans la classe `TestRectangle`, utiliser les paramètres du main pour récupérer un quadruplet de coordonnées, instancier le rectangle correspondant et le tester.

*Indication* Comme toujours, les paramètres de `main` venant du système sont des chaînes de caractères, encodant ici des doubles (`''10.0''` par exemple) qu'il faut transformer en valeurs (`10.0`). Pour cela, en Java on utilise les classes "wrappeurs" de types de base (cf. poly Java slide 12) qui offrent des méthodes static de transformation. Par exemple la méthode static `Double.parseDouble(String s)` de la classe `Double` ("wrappeur de `double`") retourne la chaine correspondante à une valeur de double.

## 2.5 Tableau de rectangles

Modifier la classe `TestRectangle` comme suit :
- Créer un tableau de rectangles dans la méthode `main` en demandant à l'utilisateur le nombre de rectangles à créer (taille du tableau) et en le remplissant par appel itéré à la méthode `creerRectangle()`.
- Afficher le tableau de rectangles ainsi créés (par appel à leur méthode `toString()`).
- Ajouter une méthode :

`static Rectangle max(Rectangle[] t)`

qui renvoie le rectangle de plus grande surface du tableau t.
- Tester cette méthode dans le main.

# 3 Applets (question supplémentaire)

## 3.1 Applet Salut

Programmer l'applet `Salut` du cours. La tester d'abord sur l'outil standard, commande :
`appletviewer salut.html &`
puis la tester sur dans le navigateur `Konqueror` (commande de même nom).

Essayez d'autres pages web contenant des applets (il n'est pas demandé de comprendre le code...) :
`http://houplin.deule.net/~bcarre/anemometre/index.html`
`http://www.oracle.com/technetwork/java/index-135948.html`

## 3.2 Affichage graphique d'un rectangle dans une applet

Dans votre répertoire ‘‘`rectangles`’’ (où se trouve la classe `Rectangle`) programmer une applet (classe `RectApplet` et fichier `html` correspondant) dont la méthode `paint` crée une instance de `Rectangle` et l'affiche sur son support `Graphics`.

Pour cela :
- voir dans la doc la méthode `drawRect(x,y,width,height)` de `Graphics`
- ajouter à `Rectangle` une méthode `display(Graphics g)` qui l'affiche par `drawRect` sur `g`. Le support `g` est fourni par l'applet dans cet exemple mais pourrait venir de tout autre environnement d'affichage graphique, ce qui rend Rectangle réutilisable.
- quels sont les objets en jeu et leurs interactions dans ce petit programme ?