POLYTECH NICE-SOPHIA — Université Nice Sophia Antipolis
Membre de UNIVERSITÉ CÔTE D'AZUR

Découvrir la programmation orientée objet avec Java

Travaux pratiques

frederic.rallo@univ-cotedazur.fr

TP1 : Classes et Objets

Avant tout, la première chose à faire est de créer dans votre répertoire d'entrée, un répertoire nommé EIIN523B. Les différents exercices d'une feuille de TP seront regroupés dans un répertoire (un répertoire par feuille de TP) situé dans EIIN523B. Le chemin du répertoire associé à chaque feuille de TP se trouvera sous le titre (~/EIIN523B/TP1/ pour cette feuille).

L'objectif des exercices ci-dessous est de mettre en œuvre **le JDK 11.x par lignes de commandes** et de créer les premières classes Java**. Attention, il est TRES FORTEMENT recommandé d'utiliser la même version que moi** du JDK car c'est celle qui sera utilisée lors d'une prochaine évaluation.

Exercice 1

Vérifiez également que vous disposez d'un bon éditeur de texte tel que sciTE, fraise, notepadqq ou notepad++. Au besoin, installez-en un.

Exercice 2

Préparer-vous et organisez-vous !!! Sur votre espace personnel, créez une arborescence qui devra permettre de localiser de façon aisée :

- L'énoncé,
- Les annexes et tous les fichiers sources qui seront manipulés pour chaque exercice de chaque séance de TD/TP.
- Les différentes versions

Enregistrer dans cette arborescence l'énoncé de la présente feuille d'exercice. Cette action devra être reproduite ensuite pour chaque séance de TD/TP.
Attention, vous utiliserez un logiciel (à partir de l'exercice 4) qui va créer un espace de travail. Cet espace de travail (workspace) devra se placer dans votre arborescence.

Découvrir la programmation orientée objet avec Java

Travaux pratiques

Exercice 3

frederic.rallo@univ-cotedazur.fr

Le but de cet exercice est de maîtriser les outils de compilation et d'exécution de la jdk.

- Utilisez uniquement une invite de commande pour les actions suivantes pour naviguer dans votre arborescence. Toujours depuis l'invite de commande, déplacez-vous dans le répertoire ~/EIIN523B/TP1/. Créez les sous-répertoires src/ et un répertoire bin/.

- Dans un éditeur tel que Notepad++, saisissez votre première classe dans le fichier *Main.java (respectez la casse)* du répertoire dans src/. Le code est suivant :

```
class Main{
	public static void main(String[] args) {
			Personne pers = new Personne("Fred");
			System.out.println("Hello, pers = "+ pers );

	}
}
```

- Créez une seconde classe *Personne.java* dans src/exercice3 en recopiant le code suivant :

```
class Personne {
	private String Name="";

	public Personne() {
		// TODO Auto-generated constructor stub
	}

	public Personne(String aName) {
		this();
		Name = aName;
	}

	@Override
	public String toString() {
		return "Personne [" + Name + "]";
	}
}
```

- Quittez l'éditeur et continuez depuis l'invite de commande. Placez-vous dans le répertoire TP1. Jusqu'à la fin de l'exercice, vous n'avez plus le droit d'en changer !

- Vérifiez le contenu du répertoire TP1:

```
+ src
	+ Main.java
	+ Personne.java
+ bin
```

POLYTECH NICE-SOPHIA — Université Nice Sophia Antipolis
Membre de UNIVERSITÉ CÔTE D'AZUR

Découvrir la programmation orientée objet avec Java

Travaux pratiques

1. Ne changez plus de répertoire. Utilisez la commande et les options à votre disposition (voir la documentation de jdk) pour compiler les 2 fichiers src\Main.java et src\Personne.java dans bin\Main.class et bin\Personne.class et vérifiez le contenu des répertoires :

```
 + src
      + Main.java
      + Personne.java
+ bin
      + Main.class
      + Personne.class
```

2. Ne changez pas de répertoire. Modifiez le nom du fichier src\Personne.java et vérifiez les contraintes de nommage imposées par le langage en essayant de compiler à nouveau. Restaurez le nom src\Personne.java

3. Exécutez le programme en notant la commande appropriée.

4. Ne changez pas de répertoire. Détruisez src\Personne.java et bin\Main.class et vérifiez le contenu des répertoires :

```
+ src
      + Main.java
+ bin
      + Personne.class
```

5. Ne changez pas de répertoire. Compilez src\Main.java en utilisant bin\Personne.class. Le but étant de créer le fichier bin/Main.class (dans bin) qui utilise le fichier bin/Personne.class (et non les fichiers .java). Vérifiez le contenu des répertoires :

```
+ src
   + Main.java
+ bin
   + Main.class
   + Personne.class
```

6. Exécutez le programme

IL EST A NOTER :
On aurait pu s'attendre à écrire java -classpath bin bin\Main en précisant le chemin du fichier Main.class mais cela ne fonctionne pas !! Il faut seulement lui indiquer le classpath et le fichier principal

Découvrir la programmation orientée objet avec Java

Travaux pratiques

Exercice 4

Vérifiez que l'IDE Eclipse est installé sur votre poste de travail, sinon installez-le. Vous utiliserez cet IDE pour la suite de vos TP. Attention à placer l'espace de travail de Eclipse (workspace) dans votre répertoire ~/EIIN523B/. Vous créerez par exemple un projet par TP et un package par questions en les nommant de la forme exercice5

Exercice 5

Créez un nouveau projet dans Eclipse. A partir du T.A.D. point suivant (codé en c) :

```
typedef struct
{
      double x;
      double y;
}point;
```

- Ecrire la classe TestPoint.java qui contiendra le point d'entrée de votre programme : la méthode **main** (`public static void main(String[]args)`). Cette méthode permettra d'instancier des objets Point et tester les méthodes créées.
- Ecrire la classe *Point.java* définissant l'objet Point en respectant les éléments suivants :

  a. Les constructeurs adéquats (sans paramètre, valeurs des attributs en paramètre) :
  ```
  public Point()
  public Point(double x, double y)
  ```
  b. Les accesseurs de consultation afin de récupérer la valeur de chaque attribut :
  ```
  public double getAbscisse()
  public double getOrdonnee()
  ```
  c. Les méthodes appliquées à ces objets :
  ```
  // --- Calcul de la distance entre le point courant et p
  public double distanceP(Point p)
  // --- Affichage d'un point
  public String toString()
  ```

- Compiler et exécuter la classe Point et retrouvez une trace qui ressemble à celle-ci.

```
* ------------------------------------------------------------ *
* TP java                                                      *
*                                                              *
* @author frédéric rallo - frederic.rallo@univ-cotedazur.fr *
* @version TD1 - ex6                                          *
* ------------------------------------------------------------ *


point A (1.0 ; 1.0)
point B (5.0 ; 4.0)
distance du point A au point B = 5.0
```

Découvrir la programmation orientée objet avec Java

Travaux pratiques

frederic.rallo@univ-cotedazur.fr

Exercice 6 :

- Introduire dans la classe Point deux nouvelles méthodes `projX()` et `projY()` permettant respectivement de calculer le *projeté d'un point support* sur l'axe des abscisses et sur l'axe des ordonnées.

- Compiler avec succès cette nouvelle version. Modifier le programme de mise en œuvre de la classe Point (i.e. *TestPoint.java*) pour y introduire des exemples d'appel aux nouvelles méthodes et exécuter ce dernier avec succès. Retrouvez une trace qui ressemble à celle-ci.

```
* ------------------------------------------------------- *
* TP java                                                 *
*                                                         *
* @author frédéric rallo - frederic.rallo@univ-cotedazur.fr *
* @version TD1   - ex6                                    *
* ------------------------------------------------------- *


point A (1.0 ; 1.0)
point B (5.0 ; 4.0)

projection du point A sur abscisses X = (1.0 ; 0.0)
projection du point A sur ordonnées Y = (0.0 ; 1.0)
projection du point B sur abscisses X = (5.0 ; 0.0)
projection du point B sur ordonnées Y = (0.0 ; 4.0)
```

Exercice 7 :

- Écrire la classe `Vecteur.java` définissant l'objet Vecteur à partir de deux objets Point
- Dans cette classe, vous ajouterez :

    a. Une méthode `main()` afin d'instancier des objets Point et tester les méthodes
    b. Les constructeurs adéquats (vecteur nul par défaut) :
    c. Les accesseurs de consultation afin de récupérer les points qui le constituent :
    d. Les méthodes appliquées à ces objets :

```
Norme du vecteur
Addition avec un autre vecteur
Millieu du vecteur
Homothétie
Translation
Symétrie
Colinéaire
```

    e. Des exemples d'appel aux nouvelles méthodes et exécuter ce dernier avec succès.

- Ajouter à la classe *Point.java* une méthode `translation()` permettant de retourner un point issu de la translation du point courant par un vecteur V
- Retrouvez une trace qui ressemble à celle-ci.

```
* ------------------------------------------------------- *
* TP java                                                 *
*                                                         *
* @author frédéric rallo - frederic.rallo@univ-cotedazur.fr *
* @version TD1 - ex7                                      *
* ------------------------------------------------------- *
```

POLYTECH NICE-SOPHIA — Université Nice Sophia Antipolis
Membre de UNIVERSITÉ CÔTE D'AZUR

Découvrir la programmation orientée objet avec Java

Travaux pratiques

frederic.rallo@univ-cotedazur.fr

```
Vecteur null ( (0.0 ; 0.0) ; (0.0 ; 0.0) )
Vecteur AB ( (1.0 ; 0.0) ; (2.0 ; 3.0) )
Vecteur BC ( (2.0 ; 3.0) ; (0.0 ; 2.0) )
Vecteur AC ( (1.0 ; 0.0) ; (0.0 ; 2.0) )
Vecteur CD ( (0.0 ; 2.0) ; (2.0 ; 4.0) )

 -------- norme ----------
norme de ||AB|| 3.1622776601683795

 -------- addition ----------
Vecteurs AB + AC =  ( (1.0 ; 0.0) ; (1.0 ; 5.0) )
Vecteurs AB + CA =  ( (1.0 ; 0.0) ; (3.0 ; 1.0) )

 -------- milieu ----------
milieu de AB =  (1.5 ; 1.5)
milieu de AC =  (0.5 ; 1.0)

 -------- homothétie ----------
 AC*2 =  ( (1.0 ; 0.0) ; (-1.0 ; 4.0) )
 BC*2 =  ( (2.0 ; 3.0) ; (-2.0 ; 1.0) )
 AB*0.5 =  ( (1.0 ; 0.0) ; (1.5 ; 1.5) )

 -------- translation ----------
 translation de AB par AC =  ( (0.0 ; 2.0) ; (1.0 ; 5.0) )

 -------- symétrie ----------
 symétrie de AB par C =  ( (-1.0 ; 4.0) ; (-2.0 ; 1.0) )

 -------- colinéarité ----------
Vecteur colinéaire ?  AB x CD ==> false
Vecteur colinéaire ?  AB x AC ==> false
```