TD n°1 – Découvrir la programmation orientée objet avec Java

L'objectif des exercices ci-dessous est de mettre en œuvre le JDK par lignes de commandes et de créer les premières classes Java en utilisant l'IDE Eclipse.

I. Installation de votre environnement de travail

Dans le cours Moodle, les liens de téléchargement du jdk Java et de l'ide Eclipse vous sont fournis.

Vous devez les installer sur votre machine, en commençant par le jdk Java (un redémarrage de votre machine sera peut-être nécessaire pour finaliser cette étape).

N'ouvrez pas Eclipse pour l'instant et passez au point suivant.

II. Création d'une arborescence de TD&TP

Sur votre espace personnel, créez une arborescence physique de sous-répertoires dédiée à la ressource R201. Cette arborescence 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.

Enregistrer dans cette arborescence l'énoncé de la présente feuille d'exercice. Cette action devra être reproduite ensuite pour chaque séance de TD/TP.

III. Options de compilation Java

Dans une invite de commandes/terminal/console, exécuter le compilateur Java, sans aucun argument.

1) Quel est le rôle de l'option –classpath ?
2) Quel est le rôle de l'option –version ?
3) Quel est le rôle de l'option –d ?

IV. Compilation et exécution d'une première classe

Dans un éditeur tel Notepad++, créer votre première classe `HelloWorld.java` en retapant le code donné en Cours. Enregistrer cette classe dans le répertoire src de la séance de TD/TP d'aujourd'hui.

1) Dans une invite de commandes/terminal/console lancée au préalable, placez-vous dans le bon répertoire et compiler avec succès la classe `HelloWorld`.
2) Identifier le nom du fichier bytecode produit par le compilateur.
3) Modifier le nom du fichier source et vérifier les contraintes de nommage imposées par le langage.
4) Finalement, exécuter ce programme dans l'invite de commandes.

V. Utiliser l'IDE Eclipse

1) Prise en main d'Eclipse

Lire le tutoriel disponible à l'adresse : https://blog.paumard.org/tutoriaux/eclipse/

Plus précisément :

- Sur votre machine, ouvrir Eclipse et positionnez-vous dans l'environnement de développement Java (standard) :
  « Window > Perspective > Open Perspective > Other… > Java »
- Choisir comme *workspace* (espace de travail), le répertoire R201 créé sur votre espace personnel (vous pouvez cliquer sur « définir par défaut et ne plus demander »)
- Créer un nouveau projet Java (*Java Project*) intitulé TD1. Vérifier que votre jdk Java est bien trouvé et cocher l'option « create separate folders for source and class files ».
- **Create module-info.java : DON'T CREATE**
- Dans ce projet (voir le « *PackageExplorer* » sur la gauche de l'écran d'Eclipse), cliquer avec le bouton droit de la souris sur le répertoire *src* et choisir « *New > package* ». Créer 1 package par exercice de TD (voir ci-dessous). ***Le nom des packages commence par une lettre minuscule.***
- Une fois le package créé, clic droit sur le package pour ajouter une classe (New > *Class*)

2) Créer une première classe

Dans votre projet TD1, créer un package `point` dans lequel vous devez écrire la classe `Point.java` définissant un objet Point à deux dimensions.

Vous avez déjà dans le fichier créé :

```
package point;
public class Point {}
```

Dans cette classe, vous ajouterez :

- les constructeurs adéquats (sans paramètre, valeurs des attributs en paramètre) :
  ```
  public Point()
  public Point(double x, double y)
  ```

- les méthodes appliquées à ces objets :
  ```
  // --- Calcul de la distance entre le point courant et p
  public double getDistance(Point p)

  // --- Affichage d'un point : [x,y]
  public void afficher()
  ```

- Dans la classe *Point.java*, ajouter une méthode **main** (`public static void main(String[] args)`) afin d'instancier des objets `Point` et tester les méthodes.

Il faut alors exécuter ce programme, c'est-à-dire la méthode `main` de la classe `Point`. Pour cela, dans le menu de gauche, cliquez avec le bouton de gauche de la souris sur `Point`, classe qui contient la méthode `main` ; puis avec le bouton droite, dans le menu déroulant, choisissez `Run as` et dans le nouveau menu déroulant `Java Application` ; les fois suivantes, vous pourrez exécuter le programme avec une petite flèche sur fond vert qui figure dans la barre en haut d'Eclipse.

3) Encapsulation

Externaliser la méthode main : vous créerez un deuxième fichier *TestPoint.java* qui contiendra uniquement la méthode main précédemment écrite dans `Point.java`. (Pensez à commenter ou effacer la méthode main de la classe `Point` !)

- Essayer d'accéder aux attributs des objets instanciés. Est-ce possible ?
- Tester l'accès en changeant l'encapsulation des attributs : *rien, public, protected, private*.
- Créer un nouveau package « point2 » dans lequel vous créerez une classe permettant de voir les effets de l'encapsulation des attributs de `point.Point`.

Les bonnes pratiques de programmation recommandent de mettre les attributs en mode *private* afin de les protéger de modifications intempestives.

- Retourner dans la classe `Point` et ajouter les accesseurs de consultation afin de récupérer la valeur de chaque attribut :
```
public double getX()
public double getY()
```

Vous pouvez ajouter les getters/setters directement dans Eclipse à partir du menu *Source > Generate Getters/Setters*.

- Vérifier dans les programmes de mise en œuvre (*main*) créés précédemment que vous pouvez accéder à la valeur des attributs par ces accesseurs.
- 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.
- Modifier `TestPoint.java` pour y introduire des exemples d'appel aux nouvelles méthodes et exécuter ce programme avec succès.

4) Utiliser l'API Java

Consulter http://docs.oracle.com/javase/10/docs/api/.
Décrire la page de l'API :

- à quoi peut-on accéder ?
- Comment rechercher une classe ? un package ?
- Quelles informations trouvez-vous dans le descriptif d'une classe ?

Rechercher l'objet `Point` existant dans l'API,

- Dans quel package se situe la classe ?
- Quelles sont les différences avec la classe que l'on a créée ?
- Ajouter les méthodes suivantes dans votre classe `Point` suivant le descriptif de l'API :
```
public Point getLocation()
public void setLocation(Point p)
public void setLocation(double x, double y)
public void translate(int dx, int dy)
```

5) Documentation

Javadoc est un outil fourni avec le JDK pour permettre la génération d'une documentation technique à partir du code source. Dans le code source Java, les commentaires utilisent des balises *@tag* reconnues par *Javadoc* pour générer la documentation technique dans des pages html.

Afin de générer la doc dans Eclipse, suivre le tutoriel suivant : http://www.objis.com/formation-java/tutoriel-java-creation-javadoc-eclipse.html

- Commenter votre classe `Point` en utilisant les balises reconnues,
- Générer la documentation en ligne dans votre Java Project Eclipse.

Guide javadoc officiel d'Oracle : https://docs.oracle.com/en/java/javase/11/javadoc/

VI. Pour aller plus loin

- Créer un nouveau package dans le projet `TD1` que vous nommez `etudiant` ;
- Créer une nouvelle classe dans le package `etudiant` que vous nommez `Etudiant` ;

Vous allez compléter le code de cette classe en ajoutant :

- un attribut privé de type `String` nommé `nom` ;
- un constructeur publique qui a un paramètre de type `String` servant à initialiser le nom de l'étudiant ;
- une méthode publique sans paramètre et qui ne renvoie rien, nommée `travailler`, qui écrit à l'écran, si le nom de l'étudiant a pour nom toto :
  ```
  toto se met au travail !
  ```
  Pour cela, il faut utiliser la fonction : `System.out.println(this.nom + " se met au travail !");`
- une méthode publique sans paramètre et qui ne renvoie rien, nommée `seReposer`, qui écrit à l'écran, si le nom de l'étudiant a pour nom toto :
  ```
  toto se repose
  ```

Ajouter une méthode **main** (`public static void main(String[] args)`) qui :

- crée un étudiant (instance de la classe `Etudiant`) en lui donnant un nom écrit directement dans le fichier source ;
- invoque la méthode `travailler` de l'étudiant créé ;
- invoque la méthode `seReposer` de l'étudiant créé .

Exécuter ce programme. Si tout fonctionne, vous allez améliorer le programme en faisant en sorte que le nom de l'étudiant soit indiqué comme argument de la méthode `main` ; pour cela

- modifiez votre méthode `main` pour que le nom de l'étudiant soit l'argument d'indice 0 du `main` (si l'en-tête de la méthode `main` est

  `public static void` `main(String[] args)`,
  il s'agit d'`args[0]`) ;

- au moment d'exécuter, faite comme précédemment, mais choisissez dans le menu déroulant de `Run as`, l'item `Run Configuration` ; dans la fenêtre obtenue par `Run configurations`, complétez en choisissant un nom pour la configuration, en notant le nom du projet (`TD1`) et la classe contenant le programme principal (`etudiant.Etudiant`) ; puis, cliquez sur l'onglet `Arguments` et inscrire le nom choisi pour l'élève dans `Program arguments`, enfin cliquez sur `Run`.