Aix*Marseille université
Socialement engagée
Année 2022-23

Domaine Sciences et Technologies
Master informatique

Génie logiciel : *TP*0
Code UE : SINAU02L

Travaux pratiques
Remise à niveau : Programmation Objet Java

Création d'un programme Java avec Eclipse

L'environnement Eclipse permet d'éditer des classes Java à l'aide de fichier texte dont l'extension est .java. Avant d'éditer une classe, il faut définir un projet. Une fois le nom du projet spécifié (saisi), un répertoire correspondant est créé dans le répertoire *workspace* (espace de travail que vous avez défini au préalable). Ensuite, des classes sont créées au fur et à mesure par le programmeur qui définira les attributs et méthodes nécessaires. Pour faciliter la lecture des programmes, nous définissons quelques conventions :

- le nom de classe est significatif et commence toujours par une majuscule, en évitant qu'il soit trop long.
- le nom des attributs et méthodes commence par une minuscule. Une méthode est différenciée d'un attribut grâce aux parenthèses. Cas particulier : le nom du constructeur commence par une majuscule.

Prenons l'exemple de la classe `Bonjour`. Elle possède un constructeur par défaut `Bonjour()`, un attribut `chaine` de type String (chaîne de caractère en Java) et une méthode `affichage()`. Cette classe peut être codée comme suit :

```
public class Bonjour{
    private String chaine;
    public Bonjour(){
        this.chaine = "tout le monde";
    }
    public void affichage(){
        System.out.println("Bonjour " + this.chaine);
    }
}
```

Compilation

La compilation d'un projet permet de générer des fichiers `.class` qui correspondent aux différentes classes du projet. Ces fichiers générés ne correspondent pas à l'exécutable de l'application mais à un code précompilé utilisé par la machine virtuelle java (JVM) pour exécuter le projet. Eclipse remplace les lignes de commandes `javac` pour générer le code précompilé et `java` pour lancer le programme par une commande `run` (accesible à partir du menu). Les variables d'environnement peuvent être initialisées au départ, ainsi lors de l'exécution du programme, elles sont utilisées dans l'ordre défini.

NB : si les modifications ne sont pas prises en compte lors de l'exécution, pensez à rafraîchir votre projet.

Syntaxe

Définir une classe, consiste à décrire ses attributs et ses méthodes tout en précisant leur nature (type de modificateur, type d'attribut, type de retour, etc.). La syntaxe ressemble beaucoup à la syntaxe utilisée en C/C++ avec une facilité aux programmeurs on ne manipule pas des pointeurs. Les instances de classes sont manipulées seulement par leur nom. Ci-dessous une description incomplète d'une classe Classe pour montrer la syntaxe employée.

```
class Classe
{
   Type attribut1;
   Type attribut2, attribut3;
   Type[] attribut4;//attribut4 est un tableau de type Type, non limité
   Modificateur Type attributX;
   Modificateur Type methode1(){
   // code de la méthode1
   ...
   }
   Modificateur void methode2(){
   // code de la méthode2
   ...
   }
}
```

Le type d'un attribut ou le type de retour d'une méthode peut s'agir d'un type élémentaire (int, float, String, etc.) ou d'une classe déjà définie.
Les modificateurs de classes, d'attributs ou de méthodes en Java permettent de définir les accès autorisés et la visibilité envers d'autres classes. Il existe plusieurs modificateurs, par conséquent nous nous limitons à ceux listés ci-dessous :
`public` : une classe, attribut ou méthode déclaré comme public est visible par toutes les autres méthodes à l'intérieur ou à l'extérieur de la classe. Généralement, on évite de déclarer des attributs comme public et on crée des méthodes spéciales pour la mise à jour. Cela permet un meilleur contrôle et un code sûr (principe de l'encapsulation des objets).
`private` : il s'agit du niveau de restriction de visibilité le plus fort. Seules les méthodes de la classe dans laquelle l'entité private a été définie, pourront la voir (attribut ou méthode).
`protected` : ce modificateur définit l'accès comme suit : si dans une classe, on déclare une méthode ou un attribut comme protected, seules les méthodes présentes dans le même paquetage (package) ou les classes dérivées (même dans un paquetage différent) pourront y accéder.
`static` : jusqu'à maintenant, nous avons considéré que chaque attribut est propre à un objet (une classe). C'est pour cette raison que nous parlons d'instance de classe. Il se peut qu'on a besoin de définir des variables communes à plusieurs instances de même classe que nous qualifions static (pi=3.14 variable commune à toutes les instances Cercle).
`final` : un attribut ou une classe qualifié de final signifie que l'objet en question est une constante. Lorsque ce modificateur est ajouté à une classe, il sera interdit de créer une classe qui en hérite.
`abstract` : une classe définie comme abstract, est une classe qu'on ne peut pas instancier. Elle possède également des méthodes non implémentées encore. Cette classe doit être dérivée pour définir des nouvelles sous-classes et par conséquent définir des instances des sous-classes.

NB : Pour le reste des TP, nous privilégions un codage des classes en anglais. La documentation du code (commentaires) doit être faite en anglais, également.

Exercice 1 : correction d'une classe

Soit la classe `Book` codée ainsi :

```
public class Book {
  // attributes
  private String title, author;
  private int NbPages
```

```
  // Constructors
  public Book(String title, String author) {
    this.title = Title;
    author = author;
  }

  // Getters
  public String getAuthor {
    return this.author;
  }

  // Setters
  void setNbPages(int nb) {
    this.NbPages = nb;
  }
}
```

Corrigez les erreurs et oublis de ce code. Ensuite, complétez la classe `Book` en ajoutant la méthode `main()` qui permet de :

1. Créer (instancier) 2 livres dont les titres et auteurs sont à définir.
2. Afficher les auteurs de ces 2 livres.

Exercice 2 : Implémentation d'une classe

1. Définissez une classe `Pen` dont les propriétés sont `length` (longueur) et `diameter` (diamètre) avec une méthode affichage appelée `print()` qui permet d'afficher les valeurs des propriétés initialisées.
2. En utilisant la méthode principale `main()` qui sera définie dans une classe différente de la classe `Book` :
   — Créez une instance d'un stylo de type `Pen` dont la longueur = 50 et le diamètre = 0.2.
   — Modifiez votre programme pour qu'on puisse saisir les propriétés d'un stylo à partir des variables du programme (généraliser votre programme pour définir n stylos).
3. Nous voulons maintenant définir trois catégories de stylos : `Pencil` (Crayon), `Bic` et `Feather` (Plume) avec les propriétés communes `lenght` et `diameter` et une méthode qui permet d'afficher les propriétés et la catégorie de chaque strylo. Implémentez la (les) classe(s) correspondante(s). Il faudra utiliser l'argument de type tableau défini au niveau la méthode `main()`.
4. Ecrivez la méthode `print()` qui prend en entrée une instance de type Pen (Pencil, Bic ou Feather) et affiche les propriétés correspondantes en précisant la catégorie.

Exercice 3 : Développement d'une application avec interface graphique

Concevez et implémentez une application Java d'une Calculette réalisant les opérations mathématiques basiques (addition, soustraction, multiplication et division).

Pour la réalisation de l'interface de cette calculette, référez-vous à la documentation Java des classes JFrame, JTextArea, JPanel, GridLayout, ActionListener, etc.

Exercice 4 : Structuration de documents

A l'aide d'un éditeur de texte (Microsoft Word ©ou OpenOffice ©) créez le document maître (DocM) ProjetDocM, ensuite les sous-documents Cahier des charges, Spécification, Conception dans le même répertoire.

Visualisez vos sous-documents (après avoir modifié le contenu de quelques sections) à travers votre document maître, insérez automatiquement la table des matières de votre document final, et exportez-le en pdf.

Pour plus d'information accédez aux sites :

1. `http://www.pcastuces.com/pratique/bureautique/document_maitre/`
2. `https://wiki.openoffice.org/wiki/FR/Documentation/Writer_Guide/Creation_document_maitre`