TP Génie Logiciel

Création d'un programme Java avec Eclipse

L'environnement Eclipse permet d'éditer des classes Java à l'aide de fichiers textuels dont l'extension est .java. Avant d'éditer une classe, il faut définir un projet. Une fois le nom du projet spécifié, un répertoire correspondant est créé dans le répertoire workspace (espace de travail). Ensuite, des classes sont créées au fur et à mesure et le programmeur insère 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 : les constructeurs commencent par des majuscules (nom de la classe).

Exemple :

Hello: est le nom de la classe
Hello() : est le nom du constructeur
word : est le nom d'un attribut
print(word) : est le nom d'une méthode

```
public class Hello {
     public static void main(String[] args) /* the program starts
     from this line – the argument args is the array where the
     program variables are saved  */
          {
               System.out.println("Hello every one");
               // print on a console window
          }
     }
```

Rappel : Les noms des fichiers sont de la forme nom de la classe.java (Hello.java)

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* (à partir du menu). Les variables de l'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 variables (attributs) et ses méthodes tout en précisant leur nature (type de modificateur, type de variable, 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 (les mots clés sont en gras).

```
class Classe
{
      Type varibale1;
      Type varibale2, variable3;
      Type[] variable4;//variable4 is an array of Type
      Modifier Type variablex;
      Modifier Type method1()
            {
                  // code of method1
                  ...
            }
      Modifier void methode2()
            {
                  // code of method2
                  ...
            }
}
```

Le type d'une variable 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 (en anglais *modifiers*) de classes, variables ou 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, variable ou méthode déclarée comme public est visible par toutes les autres méthodes à l'intérieur ou l'extérieur de la classe. Généralement, on évite de déclarer des variables 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 pourront la voir (variable ou méthode).
protected : ce modificateur définit l'accès comme suit : si dans une classe, on déclare une méthode ou une variable comme protected, seules les méthodes présentes dans le même paquetage (package) pourront y accéder ou les classes dérivées (même dans un paquetage différent). On ne peut pas utiliser protected pour qualifier une classe.
static : jusqu'à maintenant, nous avons considéré que chaque variable est propre à un objet. 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 : une variable est qualifiée de final signifie que la variable est une constante. Lorsque ce modificateur est ajouté à une classe, 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.

Exercice 01

Voici le source de la classe Livre :

```
public class Book {
  // Variables
  private String title, Author;
  private int nbOfPages // number of pages

  // Constructor
  public Livre(String unAuthor, String aTitle) {
    Author = unAuthor;
    title = aTitle;
  }

  // Getters
  public String getAuthor() {
    return A;
  }

  // Setters
  void setNbOfPages(int n) {
    nbPages = nb;
}
```

Corrigez les erreurs et oublis de ce code, et ajoutez une méthode main() pour :

- Créer 2 livres,
- Faire afficher les auteurs de ces 2 livres.

Exercice 02

1) Définissez une classe Pen (stylo) dont les attributs sont length (longueur) et diameter (diamètre) avec une méthode affichage print() qui permet d'afficher les valeurs des attributs initialisés.

2) Dans le programme principal (définissez une classe Test) :

- Créez une instance d'un stylo avec comme longueur 50 et 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 définir trois catégories de stylos : Pencil (crayon), Bic (stylo bic) et Feather (plume) avec les propriétés communes length et diameter et une méthode qui permet d'afficher ces propriétés et la catégorie. Implémentez la (les) classe(s) correspondante(s).

4) Ecrivez une méthode affichage qui prend en entrée une instance de type stylo (Pencil, Bic ou Feather) et affiche les propriétés énoncées en précisant la catégorie.

Exercice 03

Concevez et implémentez une application java pour une Calculator (calculette) réalisant les opérations mathématiques binaires (addition, soustraction, multiplication et division).

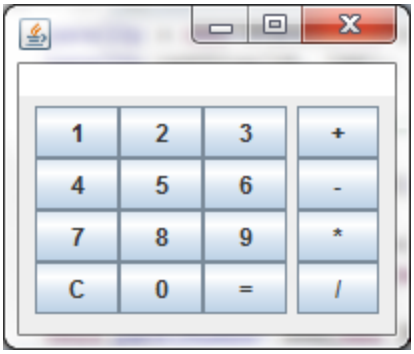

Figure 1. Interface graphique d'une calculette

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

Exercice 04

A l'aide d'un éditeur de texte (Microsoft Word © ou OpenOffice ©) créez le document maître 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