TP3 : Prise en main de l'environnement Eclipse

Florian Boudin

Module X0IC020 - 2013

Exercice 1

Dans cet exercice, vous allez devoir écrire un premier programme Java à l'aide de l'environnement de développement Eclipse. Pour vous aider dans cette tâche, la démarche à suivre a été décomposée étape par étape :

1. Commencez par lancer Eclipse. Selon les configurations, il faudra soit cliquer sur l'icône correspondante soit entrer la commande `eclipse` dans un terminal. Accepter le chemin proposé pour l'espace de travail (*workspace*, il s'agit du répertoire dans lequel les fichiers seront sauvegardés).
2. La première fois que vous démarrez Eclipse, un écran de bienvenue contenant des icônes (*Overview*, *Tutorial*, etc.) est affiché. Fermez cette fenêtre pour l'instant, vous pourrez y accéder de nouveau en cliquant *Help* → *Welcome*.
3. Créez un nouveau projet Java : *File* → *New* → *Java Project*. Donnez un nom au projet et veillez à ce que l'option *Create separate folders for sources an class files* soit activée.
4. Créez un nouveau *package* avec un clic droit (*New* → *Package*) sur le repertoire `src` du *Package Explorer* (menu de droite). La convention pour nommer un *package* consiste à mettre son url suivi du nom de package (e.g. `org.florianboudin.tutoriel`).
5. Créez une nouvelle classe avec un clic droit (*File* → *New* → *Class*) sur le nom du *package*. Donnez le nom `HelloWorld` à la classe et veillez à choisir les bonnes options (e.g. *method stubs*, *comments*, etc.)

La classe `HelloWorld` doit être composée d'une méthode `main()` permettant de faire afficher le message : *Hello World !*. Sauvegardez votre classe et exécutez le programme avec *Run*.

Exercice 2

Vous pouvez, pour vous aider, reprendre l'exercice 2 de travaux dirigés concernant la classe `Point`. Cette classe permet de créer des points dans un espace orthonormé à deux dimensions. Vous devez écrire la classe `MyPoint` et fournir les méthodes suivantes :

1. Deux constructeurs : `Point()` et `Point(double x, double y)`
2. Les méthodes `getAbscisse()`, `getOrdonnee()` retournant les valeurs en abscisse et ordonnée du point et `set(double x, double y)` permettant de modifier ses valeurs.
3. Les méthodes `distanceOrigine()` et `distance(Point unPoint)` retournant la distance du point par rapport à l'origine et par rapport à un point fourni en paramètre. Pour rappel, la distance entre deux points A et B dans un espace cartésien est calculée par $AB = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$.
4. Les méthodes `symetriqueAbscisse()`, `symetriqueOrdonnee()` et `symetriqueOrigine()` retournant le point symétrique par rapport à l'axe des abscisses, à l'axe des ordonnées et au point d'origine.
5. Ajoutez la méthode `toString()` permettant d'obtenir les coordonnées du point sous la forme `(x, y)`.
6. Créez une classe utilisatrice `DemoPoint` accueillant le programme principal et permettant de tester **toutes** les méthodes que vous avez codées.

Plusieurs structures de données et algorithmes sur celles-ci (e.g. tri d'un tableau) supposent que les objets peuvent être comparés par une relation d'ordre. Java exprime cette hypothèse en demandant qu'une classe implémente l'interface `Comparable` (vérifiez la description dans l'API `http://docs.oracle.com/javase/7/docs/api/`). Modifiez la classe `Point` pour qu'elle implémente l'interface `Comparable`, l'ordre est définit de la manière suivante : on compare les points suivant leurs coordonnées, d'abord les abscisses puis en cas d'égalité, les ordonnées.

La méthode `equals()` permet d'indiquer qu'un objet est égal à un autre. L'implémentation par défaut héritée de `Object` se contente de renvoyer `true` lorsqu'il s'agit du même objet en mémoire et ne vérifie pas les valeurs des champs de la classe (donc `x.equals(y)` équivaut à `x==y`). Dès lors que l'on a besoin de tester l'égalité des instances d'une classe, il faut que cette dernière redéfinisse la méthode `equals()`. C'est également une condition pour le bon fonctionnement de certaines méthodes des Collections de Java. Modifiez la classe `Point` pour qu'elle redéfinisse la méthode `equals()`.

Exercice 3

Écrire la classe `Carte` dont les instances représentent les différentes cartes à jouer disponibles dans une jeu de cartes. Une carte à jouer a deux propriétés distinctives : sa valeur et sa couleur. Vous utiliserez ici des énumérations pour définir le type des valeurs et couleurs, par exemple :

```
enum Couleur { COEUR, PIQUE, CARREAUX, TREFLE };
```

Écrire la classe `DemoCarte` accueillant le programme principal et permettant de créer un ensemble de 56 cartes sous la forme d'un tableau de cartes. Pour rappel :

```
// Declaration d'un tableau d'instances de la classe C contenant 5 elements
C[] tableau = new C[5];

// Parcours des elements d'une enumeration
enum Lettres { A, B, C, D};
for (Lettres uneLettre : Lettres.values()) {
    System.out.print(uneLettre); // Affiche A, puis B, etc.
}
```

Exercice 4 : les chaînes de caractères

Dans cette série d'exercices, nous allons utiliser les objets de la classe `String`. Écrire un programme qui réalise les opérations suivantes :

1. Demander à l'utilisateur la saisie d'une phrase.
2. Afficher la phrase en majuscules
3. Compter le nombre de mots de la phrase. Un mot est une séquence de lettres entourée des caractères espace (ou début/fin de chaîne).
4. Compter le nombre de caractères `a` dans la phrase puis, s'il y en a, transformer tous les `a` en `i`.
5. Tester si, entre le cinquième et le douzième caractère se trouve une séquence de caractères préalablement saisie au clavier.

Écrire un second programme qui réalise les opérations suivantes :

1. Demander la saisie de mots jusqu'à ce que l'utilisateur entre le mot `Fin`.
2. Afficher, parmi les mots saisis, le premier et le dernier dans l'ordre alphabétique.

Exercice 5 : les tableaux

Écrire un programme appelé `MoyenneDesNotes`, qui demande n notes (entier entre 0 et 20) à l'utilisateur et qui affiche la moyenne des notes saisies. Vous devez utiliser un tableau d'entiers `int[]` pour stocker les notes. Un exemple d'utilisation du programme est donné ci-dessous :

```
Entrez le nombre d'etudiants : 3
Entrez la note de l'etudiant 1: 18
Entrez la note de l'etudiant 2: 21
Note invalide, entrez une nouvelle note...
Entrez la note de l'etudiant 2: 12
Entrez la note de l'etudiant 3: 8
La moyenne est de : 12.6
```

Écrire un programme appelé `Hex2Bin` pour convertir un nombre hexadécimal (représenté par une chaîne de caractères) dans son équivalent en binaire. Un exemple d'utilisation du programme est donné ci-dessous :

```
Entrez un nombre hexadecimal : 1abc
L'equivalent en binaire de "1abc" est 0001 1010 1011 1100
```

Indice : utilisez un tableau de 16 chaînes de caractères correspondant aux nombres hexadécimaux `'0'` à `'F'` comme ci-dessous :

```
String[] hexBits = {"0000", "0001", "0010", "0011",
                    "0100", "0101", "0110", "0111",
                    "1000", "1001", "1010", "1011",
                    "1100", "1101", "1110", "1111"}
```

Exercice 6 : les méthodes

Ecrire un programme appelé `StatistiquesDesNotes`, qui demande n notes (entier entre 0 et 20) à l'utilisateur et qui affiche la moyenne, le maximum, le minimum et l'écart type des notes saisies. Votre programme doit vérifier si les notes entrées au clavier sont valides. Vous devez utiliser un tableau d'entiers `int[]` pour stocker les notes et une méthode différente pour chaque calcul. Un exemple d'utilisation du programme est donné ci-dessous :

```
Entrez le nombre d'etudiants : 4
Entrez la note de l'etudiant 1: 18
Entrez la note de l'etudiant 2: 12
Entrez la note de l'etudiant 3: 8
Entrez la note de l'etudiant 4: 10
La moyenne est de : 12.0
Le maximum est : 18
Le minimum est : 8
L'ecart type est de : 4.3
```

Pour vous aider, vous pouvez utiliser la structure de code Java donnée ci-dessous. La formule de l'écart type est $\sigma = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1} x_i^2 - \mu^2}$ où μ est la moyenne.

```
public class StatistiquesDesNotes {
  public static int[] NOTES;

  // main() method
  public static void main(String[] args) {
    lireLesNotes();
    System.out.println("La moyenne est de " + average());
    System.out.println("Le maximum est " + min());
    System.out.println("Le minimum est " + max());
    System.out.println("L'ecart type est de " + stdDev());
  }

  // Demande et memorise les notes
  public static void readGrades() { ... }

  // Retourne la moyenne
  public static double average() { ... }

  // Retourne le maximum
  public static int max() { ... }

  // Retourne le minimum
  public static int min() { ... }

  // Retourne l'ecart type
  public static double stdDev() { ... }
}
```

Exercice 7 : lecture dans un fichier

Ecrire un programme appelé `HistogrammeDesNotes`, qui lit n notes (notes anglo-saxonnes, i.e. entier entre 0 et 100) à partir d'un fichier appelé "notes.txt" et affiche un histogramme. Le format du fichier de notes, que vous pouvez télécharger sur le site web du cours, est décrit ci-dessous. La première ligne contient le nombre de notes n, la seconde contient les notes de chaque élève séparées par des espaces.

```
15
49 50 51 59 0 5 9 10 15 19 50 55 89 99 100
```

La sortie de votre programme doit comporter deux histogrammes, un horizontal et un vertical.

```
 0 -  9: ***
10 - 19: ***
20 - 29:
30 - 39:
40 - 49: *
50 - 59: *****
60 - 69:
70 - 79:
80 - 89: *
90 -100: **
                                *
                                *
 *      *                       *
 *      *                       *                       *
 *      *                   *   *                   *   *
0-9   10-19 20-29 30-39 40-49 50-59 60-69 70-79 80-89 90-100
```

Pour vous aider, vous pouvez utiliser la structure de code Java donnée ci-dessous.

```
public class HistogrammeDesNotes {
  public static int[] grades; // tableau de notes
  public static int[] bins = new int[10]; // tableau des 10 tranches (0-9, ...)

  public static void main(String[] args) {
    readGrades("notes.txt");
    computeHistogram();
    printHistogramHorizontal();
    printHistogramVertical();
  }

  // Lecture du fichier
  public static void readGrades(String filename) { ... }

  // Creation des tranches
  public static void computeHistogram() { ... }

  // Affiche l'histogramme horizontal/vertical
  public static void printHistogramHorizontal() { ... }
  public static void printHistogramVertical() { ... }
}
```

La lecture dans un fichier sera effectuée avec la classe `Scanner`.

```
Scanner in = new Scanner(new File("fichier.txt"));  // Scan input from file
int num1 = in.nextInt();                 // Read int
```

Sources

Ce document est une adaptation d'un sujet de TP similaire rédigé par Philippe Lamarre dans le cadre du cours de Programmation Orientée Objet (S31I070) et de Chua Hock-Chuan ainsi que des tutoriels de `developpez.com`

et de `oracle.com`. Rédigé par Florian Boudin, 2011–12.