TP1 : semaine 1

Classes, objets, méthodes - entiers aléatoires – tableaux – entrées/sorties console – utilitaires - String

Avant-propos : environnement et tutoriaux utiles

- L'environnement de développement supporté est NetBeans (http://netbeans.org/downloads/index.html).
- Vous trouverez beaucoup de tutoriaux sur la page campus de POO Java, entre autres le tutoriel **Mes premiers pas sur NetBeans**
- Pour ce TP et ceux qui suivront, la **documentation des API Java *(javadoc)* version 8**, que je vous conseille vivement de consulter, sera votre référence pendant que vous coderez.
- Votre code sera clairement indenté, commenté et très bien découpé et testé selon les directives qui suivront.
- Le site Openclassrooms : apprenez à programmer en Java et d'autres ressources peuvent emmener un complément d'aide aux débutants.

A lire attentivement avant les exercices

Suivez les instructions de la page internet jusqu'à la section Building and Deploying the Application http://netbeans.org/kb/docs/java/quickstart.html

Comprenez la distinction entre « Build » (pour compiler vos fichiers sources **.java** et les convertir en fichiers binaires **.class**) et « Run » (pour exécuter le fichier **.jar** créé après compilation sans erreur et distribuer l'exécutable à l'utilisateur final).

Avant de faire votre « Build » de votre projet, respectez bien les consignes du chapitre « **2 inclure les fichiers sources .java dans le jar** » du tutoriel **Mes premiers pas sur NetBeans**.

Quand vous « buildez » (compilez) votre projet avec l'icône "Marteau" ou Clean and Build Main Project (Shift-F11), un message s'affiche à la compilation : voir chapitre « **3 Compiler (build) une classe** » **Mes premiers pas sur NetBeans**.

Avec un explorateur ou un finder, allez dans le répertoire Dist (pour Distribution) de votre projet : voir chapitre « **6 Arborescence des dossiers du projet dans le workspace** » de **Mes premiers pas sur NetBeans**. Vous y trouverez votre exécutable **.jar** ainsi que ses éventuelles « librairies » (répertoire lib, nécessaire dans les derniers TP).

L'extension **.jar** est votre « .exe » java mais archivé comme un fichier zip. Néanmoins le jar java n'est pas un binaire (suite de 0 et de 1 incompréhensible pour le commun des mortels) comme en C. Essayer d'exécuter votre projet en ouvrant une fenêtre de commande (**cmd** pour Windows, Terminal pour Mac/Linux). Retapez « **java –jar** » espace « Drag&Drop » votre fichier .jar (clic-droit appuyé, déplacez le fichier .jar et lâchez) dans la fenêtre.

Pour visualiser le contenu de votre jar, consulter le chapitre « **10.1 Dezipper le jar** » dans **Mes premiers pas sur NetBeans**. Pour les utilisateurs de Win dont l'extension n'apparaît pas, suivez ce lien http://www.commentcamarche.net/faq/825-afficher-les-extensions-et-les-fichiers-caches-sous-windows. Visualisez le contenu de votre .zip. Il contient les classes de votre projet avec l'extension **.class** qui est le binaire pour la JVM (http://fr.wikipedia.org/wiki/Machine_virtuelle_Java).

Êtes-vous maintenant capable de :

- Créer un projet
- Compiler un projet
- Visualiser le code source d'un projet à partir de son jar
- Lancer un jar en dehors de l'IDE

Exercices à préparer

Exercice 1

1) Sur NetBeans, créez un nouveau projet, cliquer sur **+** devant **Source Packages** et sur le nom du package (par défaut, le même nom que le projet), supprimer le **.java** créé par défaut (clic droit, puis **Delete** et **OK**). Créez la classe nommée *Exercice1* dans le package du projet en respectant ce nom (fichier source **.java** avec le même nom que la classe *Exercice1* ci-dessous) : voir chapitre « **7 Ajouter une nouvelle classe dans un package** » dans **Mes premiers pas sur NetBeans**.

2) Copiez le code de la classe *Exercice1* ci-dessous. Compilez-la (**Build Project** dans le menu **Run** ou touche clavier **F11** ou l'icône du marteau), corrigez les erreurs et exécutez-la (clic droit sur le fichier **.java** de la classe et **Run File**). Puis consulter le chapitre « **6 Arborescence des dossiers du projet dans le workspace** » dans **Mes premiers pas sur NetBeans**.

```
public class Exercice1 {
        public static void main(String args[]) {
                /*
                Identifiez l'erreur 1 et corrigez
                */
                int i = 0;
                for (int i = 0; i < 5; i++)
                        System.out.print(i + ", ");
                System.out.print("\n");

                /*
                Identifiez l'erreur 2 et corrigez
                */
                float a = 3.0;
                double b = 4;
                float c;
                c = Math.sqrt(a * a + b * b);
                System.out.println("c = " + c);

                /*
                Identifiez l'erreur 3 et corrigez
                */
                byte b = 42;
                char c = 'a';
                short s = 1024;
                int i = 50000;
                float f = 5.67f;
                double d = .1234;
                double resultat = (f * b) + (i / c) - (d * s);
                System.out.print((f * b) + " + " + (i / c) + " - " + (d * s));
                System.out.println(" = " + resultat);
                byte b2 = 10;
                byte b3 = b2 * b;
                System.out.println("b3 = " + b3);
        }
}
```

A partir des **exercices 2** et suivants, il y aura une classe et une ou des méthodes par exercice : par exemple, la ou les méthodes de l'exercice 2 seront codées dans la classe *Exercice2*, celles de l'exercice 3 dans la classe *Exercice3* etc.

Le **main** sera dans une classe à part, par exemple *Test*, avec un menu proposant une fonctionnalité par exercice. En fonction de l'option saisie, instancier (créer) l'objet de la classe (exercice) et appeler la ou les méthode(s) de l'objet

Exercice 2

La lecture d'information au clavier durant l'exécution d'un programme en mode console n'est pas aussi simple que l'écriture. La plupart des programmeurs utilisent la classe Scanner du package java.util pour y remédier : voir le tutoriel **Exemples avec la classe Scanner**.

Dans la classe *Exercice2*, écrire une méthode qui trouve la plus grande et la plus petite valeur d'une succession de notes réelles saisies avec à blinder entre 0 et 20, ainsi que le nombre de fois où ces valeurs ont été attribuées. On supposera que le nombre de notes n'est pas connu à l'avance on s'arrête sur une valeur négative.

Utiliser le code suivant pour lire un nombre entier depuis la console :

```
import java.util.*;
...
Scanner in = null;
in = new Scanner (System.in);
int nombre = in.nextInt();
...
```

Exemples :
Saisir une note positive (<0 pour sortir) : 12
Saisir une note positive (<0 pour sortir) : 2
Saisir une note positive (<0 pour sortir) : 7
Saisir une note positive (<0 pour sortir) : 12 Saisir
une note positive (<0 pour sortir) : -1

La valeur la plus petite est 2, rencontrée 1 fois,
La valeur la plus grande est 12, rencontrée 2 fois.

Se rappeler qu'en Java, la taille d'un tableau est fixe comme dans d'autres langages, mais que sa taille est déterminé à l'exécution, et non pas à la compilation. Votre programme devrait donc créer un tableau ayant exactement la taille de la série de valeurs.

Exercice 3

Dans la classe *Exercice3*, écrire la méthode qui simule et affiche un tirage aléatoire du loto : 6 numéros entre 1 et 49, mais sans doublons. Il faut créer un objet de la classe Random dans le package java.util et utiliser sa méthode **nextInt** pour générer vos entiers aléatoires. Etudier la **documentation des API Java *(javadoc)*** de la classe pour comprendre son utilisation.

Exercice 4

Dans la classe *Exercice4*, écrire les méthodes suivantes :

- Une méthode qui instancie et initialise un tableau de 15 entiers déclaré en attribut de la classe, avec des valeurs aléatoires entre -1 et 5 (inclus).
- Une méthode qui affiche le contenu de votre tableau en affichant la série d'entiers sur la console à l'aide de la commande **System.out.println**.
- Une méthode qui dessine sur la console un histogramme en utilisant votre tableau d'entiers comme une série de valeurs occurrentes. L'affichage générée par votre programme devrait suivre le format de l'exemple ci-dessous, qui montre le résultat attendu pour la série de valeurs [-1..5] :

  5 *** 4 ** 3 * 2 *** 1 * 0 * -1 ****

Exercice 5

Remplacer la génération d'une série d'entiers aléatoires de l'exercice 4 par la saisie d'une série de valeurs depuis la console. L'utilisateur doit saisir le nombre positif (à blinder) d'entiers dans la série (tableau). Instancier le tableau en fonction de ce nombre positif puis saisir des valeurs du tableau compris entre -1 et 5 (à blinder).

Pour les exercices qui suivent vous aurez besoin des méthodes de manipulation de chaînes de caractères fournies par la classe **java.lang.**String. Lire la **documentation des API Java *(javadoc)*** de cette classe avant de continuer, surtout le grand nombre de méthodes qu'elle propose.

Exercice 6

Dans la classe *Exercice4*, écrire la méthode qui génère un mot de passe aléatoire contenant au moins 8 caractères, avec des minuscules, au moins une majuscule, un chiffre et un signe de ponctuation.

Exercice 7

Un palindrome est un mot ou une phrase sans accents que l'on peut lire dans les deux sens une fois la ponctuation et les espaces enlevés, et en transformant majuscules en minuscules. Par exemple: "Laval" et "Esope reste ici et se repose" sont des palindromes. Ecrire une méthode qui lit une phrase depuis le console, et affiche si cette phrase est un palindrome ou non. Vous trouverez d'autres exemples de palindromes sur le web.

Exercice 8

On souhaiter gérer le paramètre **args** du **main(String args[])** pour exécuter le programme par ligne de commande **cmd** (Windows) : voir le lien https://openclassrooms.com/forum/sujet/comment-passer-les-argument-de-la-methode-main-30238. L'interactivité avec l'utilisateur peut se faire via le tableau **args**. Le nombre d'éléments de ce tableau est donné par le champ **args.length**.

Reprenez votre **main** de la classe *Test* : au lieu de saisir l'option, le programme est suivi de plusieurs paramètres de type **String**, chacun associé à une option de type **int** du menu. Pour chaque option valide entre 2 (exercice 2) et 7 (exercice 7), appeler les méthodes de l'exercice associé.

Exemple en ligne de commande :

java Test 4 2 7 exécute la classe *Test* avec les 3 paramètres (options) 4, 2 et 7

Vous aurez besoin de la méthode utilitaire **Integer.parseInt** pour convertir une valeur de type String, passée en paramètre, en une valeur de type **int**. Si la valeur en paramètre n'est pas un entier, cette méthode génère une exception NumberFormatException : voir chapitre sur les exceptions ultérieurement.