SEMAINE 1 : TP1 - UTILISER DES TYPES OBJETS

1 EDITER, COMPILER, INTERPRÉTER UN PROGRAMME JAVA

1.1 RÉPERTOIRES, ÉDITION

Créer un répertoire **`TP1`** contenant deux sous-répertoires **`src`** et **`bin`**.

Les fichiers **`.java`** que vous allez éditer seront placés dans le répertoire **`src`**.

Les fichiers **`.class`** résultant de la compilation seront placés dans le répertoire **`bin`**.

Au cours de la séance, vous resterez dans le répertoire **`TP1`**.

Créer dans le répertoire **`src`** , à l'aide de l'éditeur de texte de votre choix, un fichier **`Hello.java`** contenant les lignes suivantes :

```
public class Hello{
  public static void main(String[] args){
    System.out.println("Hello, Java?");
  }
}
```

1.2 COMPILATION

La commande **`javac`** permet de compiler un programme java.

Sur la ligne de commande, hormis le nom de la classe qui vient en dernier, il faut spécifier 2 options :

- **`-sourcepath`** ***`src`*** : où *src* est le répertoire où se trouvent les fichiers sources **`.java`**
- **`-d`** ***`bin`*** : où *bin* est le répertoire où seront placés les fichiers "**`.class`**" générés (*le répertoire doit exister)*

Consultez le manuel *(`man javac`)* pour plus de précisions sur les autres options.

En étant dans le répertoire **`TP1`**, la ligne de commande à saisir pour compiler la classe est :

```
javac -sourcepath src -d bin Hello.java
```

Cette instruction crée un fichier **`Hello.class`** dans le répertoire **`bin`**.

1.3 INTERPRÉTATION

La commande **java** permet d'interpréter, et exécuter des fichiers **.class.**

Sur la ligne de commande, hormis le nom de la classe, il faut indiquer le répertoire contenant les fichiers **.class** , le répertoire *bin* dans notre cas.

- option **-classpath ou -cp**

Consultez le manuel *(man java)* pour plus de précisions sur les options.

A partir de votre répertoire **TP1**, la ligne de commande à saisir pour exécuter le programme est la suivante :

java -classpath bin Hello ou **java -cp bin Hello**

Cette commande nécessite que le répertoire **bin** existe et qu'il contienne un fichier nommé **Hello.class**.

On notera que l'extension **.class** n'est pas spécifiée.

Si ni **-cp** *ni* **-classpath** *ne sont spécifiés, les* **.class** *sont recherchés dans le répertoire courant .*

Le résultat de l'exécution du programme devrait être l'affichage à l'écran du message :

```
Hello Java?
```

1.4 ARGUMENTS SUR LA LIGNE DE COMMANDE

Lors de l'exécution il est possible de transmettre des valeurs sur la ligne de commande.

Ces valeurs sont accessibles dans le main via le tableau de **String** passé en paramètre.

Exemple :

```
public class Numerote{

  public static void main(String[] args){
    for(int i=0;i<args.length;i++){
      System.out.println(" (" + (i+1) + ") " + args[i]);
    }
  }

}
```

Compiler, exécuter en saisissant la ligne suivante : **java -cp bin Numerote Java C PHP**

Le résultat de l'exécution du programme devrait être l'affichage à l'écran du message :

```
(1) Java
(2) C
(3) PHP
```

TÉLÉCHARGER DES CLASSES

Pour les exercices suivants nous allons réutiliser des classes définies spécifiquement et déposées sur Moodle.

(Celles-ci se trouvent dans la première section du cours dans le répertoire ressources)

Voici une première manière (nous en verrons d'autres) de réutiliser des classes existantes ne faisant pas partie de l'API Java.

Pour pouvoir utiliser des classes existantes dans un programme, une classe, `Essai`, vous devez :

1. **Créer un répertoire , `lib`,** par exemple, pour accueillir les classes à utiliser :
   (y placer les fichiers `.class` téléchargés, Telephone.class et Carte.class pour ce TP)

2. **Compiler la classe** en précisant ce répertoire, `lib`, en plus du répertoire des sources `.java`
   Pour séparer les chemins le séparateur est le caractère "`:`" :

   option `-classpath` ou `-cp`

   ```
   javac -sourcepath src -cp bin:lib -d bin Essai.java
   ```

3. **Exécuter votre classe** en précisant le répertoire où se trouvent les classes utilisées :

   option `-classpath` ou `-cp` :

   ```
   java -cp bin:lib Essai
   ```

2 TÉLÉPHONE

Pour cet exercice vous allez utiliser la classe **`Telephone`** (***`Telephone.class`*** *à télécharger depuis Moodle).*

On dispose ainsi d'un type objet **`Telephone`**, dont les spécifications sont les suivantes :

`Telephone(int un,int deux,int trois,int quatre,int cinq)`	`Crée un N° de téléphone initialisé avec les numéros fournis`
`String toString()`	`Retourne le N° de téléphone sous la forme d'une chaîne : "03.20.33.44.55"`

1) Ecrire une classe `EssaiTelephone` contenant une méthode **`main()`** qui affiche, en s'appuyant sur la méthode **`toString()`,** successivement les 5 numéros de téléphone fictifs :

```
00.00.00.00.00
11.11.11.11.11
22.22.22.22.22
33.33.33.33.33
44.44.44.44.44
```

2) Compléter la méthode `main()` qui affiche 5 numéros de **`Telephone`** générés aléatoirement (i.e. dont chacun des numéros est généré aléatoirement), le 1er numéro est compris entre 1 et 9.

```
01.25.36.63.15
05.55.74.85.69
03.33.21.54.09
01.21.45.45.66
09.09.10.01.90
```

La classe **`java.util.Random`** modélise un générateur de valeurs aléatoires ***(Consultez les spécifications)***
*(à utiliser de préférence à la méthode **`Math.random()`** également définie dans l'API)*

Pour générer une valeur aléatoire comprise entre 0 et *n* (n non inclus), on utilise un objet de type **`Random`**, dont les spécifications sont les suivantes :

`Random()`	`Crée un générateur de nombres aléatoires.`
`int nextInt(int max)`	`Retourne aléatoirement un entier compris dans l'intervalle [0,max[`

3) Modifier la méthode `main()` pour que le nombre de numéros générés soit issu de la ligne de commandes *(argument passé en ligne de commande).*

3 CARTE

Pour cet exercice vous allez utiliser la classe **`Carte`** (*fichier* **`Carte.class`** *à télécharger depuis Moodle).*

On dispose d'un type objet **`Carte`** qui représente une carte à jouer :

- Les couleurs sont codées : 0:Trèfle 1:Carreau 2:Coeur 3:Pique
- Les valeurs sont codées : 7:0 8:1 ... As:7

Les spécifications de la classe **`Carte`** sont les suivantes :

`Carte(int couleur, int valeur)`	`Crée une carte avec une couleur et une valeur.`
`int getCouleur()`	`Retourne la couleur de la carte.`
`int getValeur()`	`Retourne la valeur de la carte.`
`boolean equals(Carte autreCarte)`	`Retourne vrai si deux cartes sont identiques, faux sinon`

`int compareTo(Carte autreCarte)`	`Retourne :`
	`0 si les deux cartes sont de même valeur,`
	`<0 si la valeur est plus petite que celle de autreCarte,`
	`>0 si la valeur est plus grande que celle de autreCarte.`

`boolean precede(Carte autreCarte)`	`Compare couleur et valeur pour déterminer si la carte précède celle passée en paramètre.` `Retourne vrai si oui faux sinon`
`String toString()`	`Retourne la carte sous la forme d'une chaîne :` `"As de Trèfle"`

Ecrire une classe **`EssaiCarte`** contenant une méthode **`main()`** qui :

1. crée et affiche les 32 cartes d'un jeu complet,
2. génère 2 cartes différentes aléatoirement, les affiche en indiquant laquelle précède l'autre,
3. génère 10 cartes aléatoirement, il peut y avoir plusieurs cartes identiques, affiche la carte de valeur la plus élevée.

POUR ALLER PLUS LOIN

Ecrire une classe **`Paquet`** contenant une méthode **`main()`** qui :

1. crée un tableau représentant un jeu de 32 cartes,
2. le mélange,
3. le réordonne selon l'ordre défini dans la classe par la méthode **`precede()`**.