L2 info S3
UE : Programmation Orientée Objets 1 - Java

TD1 - Eléments essentiels pour démarrer en Java

Objectifs

A la fin de ce TD, vous devez être capables de :

1. Pouvoir lire et écrire du code en conformité avec les tables syntaxiques, les conventions de Java afin de repérer et corriger facilement les erreurs syntaxiques.
2. Nommer les fichiers comportant le code source, donner les instructions pour compiler et lancer l'exécution d'un programme.
3. Compléter un programme par modification ou insertion de nouvelles instructions simples.
4. Pour les Entrées, transmettre les données soit sur la ligne de commande, soit de façon interactive. Convertir les données à partir du type `String`.
5. Pour les Sorties, afficher un résultat sur le terminal qui corresponde aux spécifications données (c-à-d respect de la mise en forme, valeur exacte).
6. Interpréter les profils (i.e. signature, en-tête) des méthodes et faire les appels adéquats (hors hiérarchie).
7. Comprendre le principe des constructeurs (hors hiérarchie) c-à-d reconnaître leur écriture spécifique et simuler leurs effets.
8. Accéder aux attributs d'instance privés et publics à la fois au sein et hors de leur classe d'appartenance.
9. Faire la différence entre un type primitif et une classe.

1 Principes de base

Soit le programme Java suivant :

```
public class Calculatrice
{
  public static void main(String[] args)
  {
     if (args.length !=3)
     {
        System.out.println("Nombre d'arguments incorrect ! ");
     }
     else
     {
        int op1 = Integer.parseInt(args[0]);
        int op2 = Integer.parseInt(args[2]);
        switch (args[1].charAt(0))
        {
          case '+' :   System.out.println(op1+op2);
                       break;
          case '-' :   System.out.println(op1-op2);
                       break;
          case '*'  :   System.out.println(op1*op2);
                       break;
          case '/' :   System.out.println(op1/op2);
                       break;
          default :    System.out.println("operateur inconnu ...");
        } // switch
     } // else (nombre d'arguments corrects)
  } // method main
} // class
```

A. Lecture du code source

1. Dire ce que fait ce programme.
2. Donner le nombre et le nom de(s) classe(s) dans ce code source.
3. Donner le nom des attributs pour chaque classe du code source.
4. Donner le nom des méthodes pour chaque classe du code source.
5. Indiquer s'il y a un constructeur dans ce code source.
6. Vérifier que la syntaxe est respectée (définition de classes, de méthodes, instruction `switch`).
7. Vérifier que les conventions de nommage des identifiants sont respectées (nom de classes, d'attributs, de méthodes).
8. Expliquer pourquoi les instructions `System.out.println(...)` sont correctes.
9. Préciser pour chaque méthode si elle est méthode de classe ou méthode d'instance.

B. Edition - compilation - exécution

Donner les différentes étapes à effectuer pour exécuter un programme donné sous forme de code source

1. Dire quel doit être le nom du fichier source.
2. Dire comment compiler le programme.
3. Dire comment exécuter le programme. Donner quelques exemples.
   Remarque :
   `*` sur la ligne de commande sera mal interprété (interprétation du SE) ;
   pour cet opérateur taper `"*"` ou `\*`.

C. Affichage du résultat

On souhaite modifier l'affichage du résultat pour avoir un texte tel que :
`Le resultat de <nom opération> est : <résultat>`.
Par exemple, si le programme a été appelé avec les arguments `45 + 5`,
l'affichage serait : `Le resultat de la somme est : 50`. Idem pour les autres opérateurs.

1. Ecriture correcte de l'instruction

   (a) Expliquer, pour chacune des instructions suivantes, le résultat obtenu

   1. ```
      System.out.println("Le resulat de la somme est : " + op1+op2) ;
      ```
      `Le resultat de la somme est : 455`
   2. ```
      System.out.println("Le resulat de la somme est : " + (op1+op2)) ;
      ```
      `Le resultat de la somme est : 50`
   3. ```
      System.out.println(op1 + args[1].charAt(0) + op2 + "=" + op1+op2);
      ```
      `93 = 455 :`
   4. ```
      System.out.println(op1 + (char) args[1].charAt(0) + op2 + "=" + (op1+op2));
      ```
      `93 = 50`
   5. ```
      System.out.println((String) op1 + (String) args[1].charAt(0) +
                (String) op2 + " = " + (op1+op2)) ;
      ```
      qui produit une erreur de compilation : `inconvertible types`.
      Aurait-on pu appeler la méthode `toString` ?
   6. ```
      System.out.print(op1);
      System.out.print(args[1].charAt(0));
      System.out.print(op2);
      System.out.println( " = " + (op1+op2)) ;
      ```
      `45+5=50`

   (b) Donner le résultat des instructions suivantes :

   1. ```
      System.out.println("" + op1 + args[1].charAt(0) + op2 + "=" +
                           (op1+op2));
      ```
   2. ```
      System.out.println(op1 + args[1] + op2 + "=" + (op1+op2));
      ```

2. Modifier le code source pour afficher le résultat de chaque opération avec le texte
   `<opérande 1> <opérateur> <opérande 2> = <résultat>`

D. Lecture, autres calculs

Ajouter les instructions

1. pour rendre les lectures interactives (lecture terminal),
2. permettant de calculer un modulo (`%`) et `x^y` (x^y : x à la puissance y).
   Aide :

   utiliser dans la classe `Math`, la méthode `pow` définie comme suit :
   `public static double pow (double a, double b)`
   Le cast `int` → `double` n'est plus obligatoire.

E. Amélioration du programme : gestion des erreurs

1. Donner les erreurs susceptibles de se produire et qui sont traitées (maladroitement) dans ce code source.
2. Donner les autres erreurs susceptibles de se produire et non traitées dans le code source.

Remarque :

La gestion des exceptions n'est pas au programme. L'accent ne sera pas mis sur ce point au cours de ce semetre.

2 Programme Java avec plusieurs (2) classes

Considérer le fichier suivant qui comporte 2 classes :

```
class Point
{
   public int x ;
   public int y ;

   public Point(int x0, int y0)
   {
      x=x0 ;
      y=y0;
   }
}

public class CreationPoint
{
   public static void main (String [] args)
   {
     System.out.println("Bonjour !") ;
     Point point1 = new Point(0,0) ;
     Point point2 = new Point(3,4) ;
     Point point3 = new Point(7,1);
   }
}
```

Questions

1. Dire ce que fait ce programme.
2. Edition - compilation - exécution
   (a) Donner le nom du fichier et les instructions pour compiler et exécuter ce programme.
   (b) Il est préférable de n'avoir qu'une seule classe par fichier. Donner les modifications à apporter et comment lancer l'exécution du programme.
3. Donner l'ordre des fonctions appelées par le programme.
4. Affichage des attributs
   Tester les différentes possibilités d'afficher les coordonnées d'un point :
   (a) Dans la méthode `main`, instruction `System.out.println` avec accès direct aux attributs du point (`point1.x` et `point1.y`).
   Que se passe-t-il si les attributs deviennent privés ?
   (b) L'affichage est exécutée dans une méthode `public void afficher()` définie dans la classe `Point` (la méthode `afficher` comporte l'instruction `System.out.println`). Que se passe-t-il si les attributs sont privés ?
   (c) La classe `Point` comporte une méthode `public String mesAttributs()` qui renvoie le texte `"Abscisse : ..., ordonnee : ...."`. Appel de la méthode dans le `main`.
   (d) La solution la plus élégante est de redéfinir la méthode `toString` : `public String toString()` dans la classe `Point`.

3 Types primitifs, classes

1. Quels sont les types primitifs existants ?
2. Quelle est la différence entre `int` et `Integer` ?
3. Écrire un programme qui calcule la moyenne des nombres entiers passés en arguments de la ligne de commande.
   Écrire les 2 versions : les données sont de type `int`, les données sont de type `Integer`.
   Remarque :
   — utiliser `args.length`
   — pour convertir un `String` en `Integer` : utiliser dans la classse `Integer`, le constructeur `Integer(String)`