2I002 - Introduction à la programmation Objet

Groupe 6 – Test n°1

Thomas Robert Remi Cadene

12 octobre 2017 - Durée : 30 minutes

Partie 1 – Questions de cours

1.1 Compilation et exécution

1. **On veut développer une classe `Truc`. Quel doit être le nom du fichier dans lequel on écrit le code de cette classe ?**
   `Truc.java`
2. **Quelle commande doit-on exécuter pour compiler cette classe ?**
   `javac Truc.java`
3. **Quelle commande doit-on exécuter pour lancer cette classe ?**
   `java Truc`
4. **Quelle méthode est exécutée lorsqu'on lance cette classe ?**
   `public static void main(String[] args)`

1.2 Conventions de nommage, modificateurs

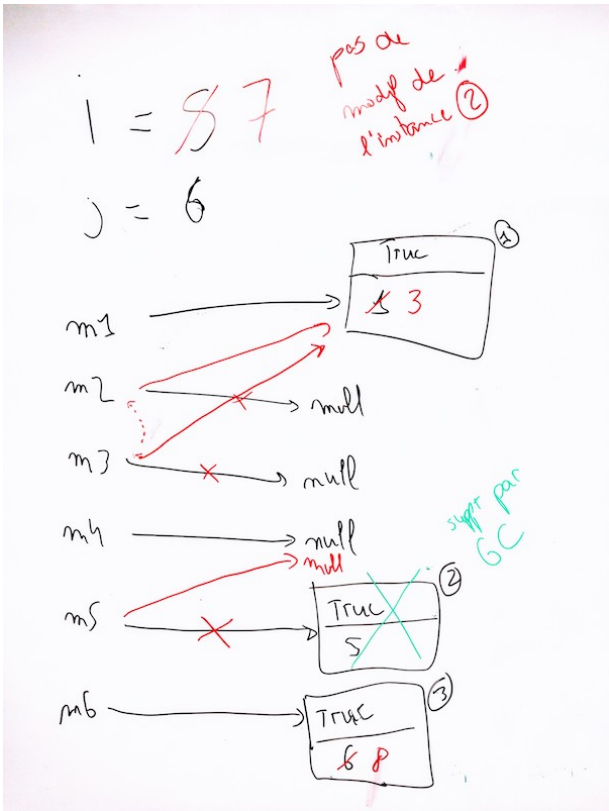
5. **Quelle sont les conventions de nommage pour les noms de classe, les variables, les méthodes et les constantes ?**
   `MaClasse`, `maVariable`, `maMethode`, `CONSTANTE`
6. **Quels sont les différents modificateurs d'accès (visibilité) possibles en Java (au moins les deux plus courants) ?**
   `public`, `private`, `protected`, "friendly" (absence de modificateur)
7. **A quoi s'appliquent-ils ?**
   Majoritairement aux variables d'instance et de classes et aux méthodes et constructeurs. Également aux classes mais avec un "pouvoir d'expression" du modificateurs assez limité (uniquement public ou private (absence de modificateur) et usage du private très marginal)
8. **Quel principe de la programmation orientée-objet permettent-ils de satisfaire ?**
   L'encapsulation

1.3 Cycle de vie

On dispose d'une classe `Truc` qui stocke l'entier fourni au constructeur et dont on peut modifier la valeur avec `setX`. On execute le code ci-dessous :

```
1  int i = 5;
2  int j = 6;
3  Truc m1 = new Truc(1);
4  Truc m2 = null;
5  Truc m3 = null;
6  Truc m4;
7  Truc m5 = new Truc(i);
8  Truc m6 = new Truc(j);
9  i = 7;
10 m2 = m1;
11 m3 = m2;
12 m5 = m4;
13 m3.setX(3);
14 m6.setX(8);
```

9. **Dessinez un schéma indiquant toutes les variables, toutes les instances crées et toutes les références à la fin de l'éxécution du programme.**



Note : Quand on passe une variable en paramètre, on passe en fait sa valeur en paramètre, valeur qui est copiée. Lorsque la **variable est d'un type de base**, la **valeur est directement la "donnée utile"** (la valeur du nombre pour un `int`). Lorsque la **variable est d'un type complexe**, la **valeur est une référence à une instance** (ou à `null`), il y a donc copie de la référence (la modification de la nouvelle variable pour référencer une nouvelle instance ne modifie pas la référence intiale) mais pas de copie de l'instance.

Dans le premier cas donc, il n'y a pas de mécanisme de "référence", et donc dans cette question, la modification de `i` n'entraine aucun changement là où `i` avait précédemment été utilisé. Pour que ça soit le cas, il faudrait que `i` soit de type complexe, auquel cas sur notre schéma il y aurait bien une flèche indiquant une référence de l'instance (1) de `Truc` vers `i`.

10. **Quelles sont la ou les classes, variables, instances, références ?**
    Classe : Truc
    Variables : i,j,m1 à m6 Instances : objets en mémoire résultats des appels à "new", boites sur le schéma
    Références : m1 à m6 sont des références, soit une des instances soit null, flèches sur le schéma
11. **Combien d'instances ont été créées ? Certaines instances seront-elles détruites au passage du garbage collector ? Si oui lesquelles ?**
    3 instances créées, 1 instance détruite, l'instance (2) barrée en vert sur le schéma

Partie 2 – Un peu de code...

Développez une classe `Personne` qui aura :

— **Des données :**
  — Un prénom, un nom et un âge
— **Des constructeurs :**
  — Un constructeur prenant en entrée un prénom et un nom ; l'âge par défaut est `0`

  - Un constructeur prenant en entrée un prénom, un nom et un âge
  - *Note : un constructeur doit utiliser l'autre*
- **Des accesseurs (getters/setters) :**
  - Permettant de récupérer le prénom, le nom et l'âge d'une personne
  - Permettant de modifier le nom d'une personne
- **D'autres méthodes :**
  - Une méthode `viellir` sans paramètre pour augmenter l'âge d'une personne d'un an
  - Une surcharge de la méthode précédente avec un entier `n` en paramètre qui augmente l'âge de `n`
  - La méthode `toString` qui retourne une chaîne avec les informations de la personne

Écrivez également une classe `TestPersonne` qui teste votre classe `Personne` :

- Création et affichage de 2 personnes `p1` et `p2` de noms différents
- Modifier `p1` pour que son nom soit le même que celui de `p2`
- Faire viellir `p2` d'un an

```java
public class Personne {
    private String prenom;
    private String nom;
    private int age;

    public Personne(String prenom, String nom, int age) {
        this.prenom = prenom;
        this.nom = nom;
        this.age = age;
    }

    public Personne(String prenom, String nom) { this(prenom, nom, 0); }

    public String getPrenom() { return prenom; }
    public String getNom() { return nom; }
    public int getAge() { return age; }

    public void setNom(String nom) { this.nom = nom; }

    public void viellir() { this.age++; }
    public void viellir(int n) { this.age += n; }

    public String toString() {
        return "Personne[" + prenom + " "+ nom + " (" + age +"ans)]";
    }
}

public class TestPersonne {
    public static void main(String[] args) {
        Personne p1 = new Personne("Thomas", "Robert", 24);
        Personne p2 = new Personne("Remi", "Cadene", 24);

        p1.setNom(p2.getNom());
        p2.viellir();
    }
}
```