TD1 : Premières classes Java

Florian Boudin

Module X0IC020 - 2013

Exercice 1

Soient les deux classes décrites ci-dessous.

```
// fichier Classe1.java
public class Classe1 {
    int a = 0 ;
    public void f() {
        a++ ;
    }
    public void affiche() {
        System.out.println(a);
    }
}

// fichier Classe2.java
public class Classe2 {
    int a = 0 ;
    public void f(int a) {
        System.out.println(a) ;
    }
}
```

1. Indiquez quel sera l'affichage lors de l'exécution des programmes ci-dessous.
2. Dans le premier programme, comment feriez vous pour faire afficher la valeur du champ `a` de `Classe2` ?

```
// fichier Programme1.java
public class Programme1 {
    public static void main(String[] args) {
        Classe2 p = new Classe2();
        p.f(12);
    }
}
```

```java
// fichier Programme2.java
public class Programme2 {
    public static void main(String [] args) {
        Classe1 p = new Classe1();
        Classe1 q = new Classe1();
        p.affiche();
        q.affiche();
        p.f();
        p.affiche();
        q.affiche();
        p = q;
        p.f();
        p.affiche();
        q.affiche();
    }
}
```

Exercice 2

Soit la classe `Point` décrite ci-dessous :

```java
// fichier Point.java
class Point {
    // deux attributs de type int
    int abscisse;
    int ordonnee;

    // constructeur
    public Point() {
        abscisse = 0;
        ordonnee = 0;
    }

    // methode permettant de changer
    // les coordonnees d'un point
    public void set(int u , int v) {
        abscisse = u;
        ordonnee = v;
    }
}
```

1. Ajouter à la classe `Point` les méthodes `getAbscisse` et `getOrdonnee` telles que `p.getAbscissse()` et `p.getOrdonnee()` retournent respectivement l'abscisse et l'ordonnée du point `p`.
2. Ajouter à la classe `Point` la méthode `translatePoint` de type de retour `void`, de telle sorte que `p.translatePoint(u, v)` effectue une translation du point `p` de déplacement `u` sur l'axe des abscisses et `v` sur celui des ordonnées.

3. Ajouter à la classe `Point` la méthode `origine`, de type de retour `boolean` qui teste si les coordonnées du point sont nulles.
4. Ajouter également une méthode `egaleCoord` telle que `p.egaleCoord(q)` renvoie `true` si et seulement si les abscisses et ordonnées des points `p` et `q` sont égaux.
5. Écrire un deuxième constructeur de la classe `Point`, tel que `Point(int u, int v)` permet d'initialiser l'abscisse et l'ordonnée avec `u` et `v`.
6. Écrire une seconde méthode `set`, prenant en argument un objet de la classe `Point`, et qui recopie les champs de ses arguments.
7. Ajouter à la classe Point une méthode `symetrie` telle que `p.symetrie()` renvoie un nouvel objet `Point` qui représente le symétrique du point `p`, dans une symétrie centrale par rapport à l'origine du repère.
8. Écrire une classe utilisatrice de la classe `Point` qui doit au minimum effectuer les opérations suivantes : Créer les points `p1`, `p2`, `p3` de coordonnées respectives (1, 1), (3, 4), et (1, 1). Le programme doit alors vérifier si `p1` et `p3` ont les mêmes coordonnées, puis effectuer une translation de (2, 2) pour `p1` et créer un point `p4` symétrique de `p1` par rapport à l'origine. Enfin, il doit afficher les coordonnées de chacun des points.

Exercice 3

Écrire une classe `Personne`, ayant pour champs un nom (chaîne de caractères) et une année de naissance. Écrire un constructeur ainsi que les méthodes d'accès et de modification. Écrire les méthodes `calculAge` et `difference` telles que :

— `p.calculAge(an)` retourne l'âge de la personne `p` à la fin de l'année `an`
— `p1.difference(p2)` retourne le nombre d'années de différence entre la personne `p1` et la personne `p2`

Écrire une méthode qui renvoie une chaîne contenant autant d'étoiles (`*`) que d'années de différence entre deux personnes. Écrire une programme créant deux personnes et affichant leur différence d'âge sous forme d'une chaine d'étoiles.

Exercice 4

Écrire une classe `NombreCache` dont le seul champ est un entier nommé `unNombre`. Cette classe gère un entier qu'elle cache. Vous devez ensuite écrire :

1. Un constructeur `NombreCache` qui permet de créer une instance de la classe ; le nombre est choisi aléatoirement, entre 0 et 100. Supposez que les méthodes suivantes existent :
   — une méthode `float random()` de la classe `Math` qui génère une valeur de type float appartenant à l'intervalle `[0,1]`
   — une méthode `int round(float a)` de la classe `Math` qui renvoie la valeur entière la plus proche du nombre passé en paramètre
2. Une méthode `compare` qui prend en paramètre un entier et renvoie -1 si l'entier en paramètre est inférieur à `unNombre`, 0 s'il est égal, et 1 si il est plus grand que `unNombre`

La deuxième partie de cet exercice consiste à programmer un jeu demandant à un utilisateur de trouver un nombre généré aléatoirement, avec un nombre maximum de tentatives permises. Écrire la classe `Jeu` qui possède un champ `maxEssais` représentant le nombre de tentatives permises pour deviner le nombre. Vous devez ensuite écrire :

1. un constructeur `Jeu` qui prend en paramètre le nombre de tentatives maximum
2. une méthode `reInit(int nb)` qui prend un entier en paramètre et affecte sa valeur au champ `maxEssais`
3. une méthode `lancer` qui utilise la classe `NombreCache` pour créer un nombre cible entre 0 et 100 puis demande à l'utilisateur une valeur tant qu'il n'a pas trouvé le nombre cible et que le nombre de tentatives n'est pas atteint. Cette méthode renvoie la valeur booléenne `true` si le joueur a trouvé, et renvoie `false` sinon. La méthode affiche trop grand, trop petit ou trouvé selon la valeur proposée par l'utilisateur lors d'une tentative. Dans le cas où la valeur cible est trouvée, le nombre de tentative doit être affiché. Ainsi, avec `maxEssais` égal à 4, et un nombre à trouver égal à 76, un exemple de dialogue avec l'utilisateur après appel de la méthode serait :

```
Donne un entier : 14
trop petit !
Donne un entier : 87
trop grand !
Donne un entier : 76
trouve ! en 3 tentatives
```

On rappelle que pour lire des entiers entrés au clavier par l'utilisateur, on peut utiliser la classe `Scanner` de la façon suivante :

```
import java.util.Scanner

// creation d'un objet de type scanner
Scanner saisie = new Scanner(System.in);
System.out.print("Donner un entier : ");
// la methode nextInt() permet de lire l'entier suivant
int entier = saisie.nextInt();
```

Sources

Ce document est une adaptation d'un sujet de TD similaire rédigé par Philippe Lamarre dans le cadre du cours de Programmation Orientée Objet (S31I070). Rédigé par Florian Boudin, 2011–12.