ULP - UFR de Mathématiques et d'Informatique

Corrigé du TD 2 : références et héritage

1 Références

Trois entiers sont crées en mémoire, tandis que seulement deux points le sont : `p3` est une référence vers `p1`, et non un objet à part entière.

Ce progamme affiche :

```
i1: 5 i2:10 i3:10
i1: 5 i2:10 i3:100
x: 5    y:5
x: 10   y:10
x: 5    y:5
x: 100  y:5
x: 10   y:10
x: 100  y:5
```

Le code de la fonction `ajoute` est le suivant :

```
void ajoute(Point p)
{
    p.x += 5;
}
```

Les paramètres des fonctions sont passés par référence et la modification du paramètre dans la fonction appelée est repercutée dans la fonction appelante.

2 Classes et objets

2.1 Classe `Lapin`

```
class Lapin
{
    String nom;
    int age;
    int position;
    int longueurPoils;

    Lapin(String _nom)
    {
        nom = _nom;
        age = 1;
        position = 0;
        longueurPoils = 1;
    }
```

```
    void afficher()
    {
        System.out.println("Lapin " + nom + ", position " + position);
    }

    void avancer()
    {
        position += Math.round(Math.random()*10) - 4;
    }
}
```

2.2 Classe Course

```
class Course
{
    public static void main(String [] args)
    {
        Lapin bugs = new Lapin("Bugs Bunny");
        Lapin roger = new Lapin("Roger Rabbit");
        Lapin max = new Lapin("Max");
        Lapin eusebe = new Lapin("Eusebe");

        Lapin[] lapins = new Lapin[4];
        lapins[0] = bugs;
        lapins[1] = roger;
        lapins[2] = max;
        lapins[3] = eusebe;

        for(int i=0; i<100; ++i)
        {
            for(int l = 0; l<lapins.length; ++l)
            {
                lapins[l].avancer();
                lapins[l].afficher();
            }
        }
    }
}
```

3 Héritage

3.1 Classe Animal

```
class Animal
{
    String nom;
    int age;
    int position;
}
```

3.2 Classe Lapin

```
class Lapin extends Animal
{
    int longueurPoils=1;

    Lapin(String _nom)
    {
        super(_nom);
    }

    void afficher()
    {
        System.out.println("Lapin : "+ nom + " position: "+position);
    }

    void avancer()
    {
        position += Math.round(Math.random()*10) - 4;
    }
}
```

3.3 Classe Tortue

```
class Tortue extends Animal
{
    int tailleCarapace = 100;

    void afficher()
    {
        System.out.println("Tortue : "+ nom + " position: "+position);
    }

    void avancer()
    {
        position += Math.round(Math.random()*2)+1;
    }

    Tortue(String _nom)
    {
        super(_nom);
    }

}
```

4 Polymorphisme

Voici la partie modifiée du `main` :

```
Lapin bugs = new Lapin("Bugs Bunny");
Lapin roger = new Lapin("Roger Rabbit");
Lapin max = new Lapin("Max");
```

```
Lapin eusebe = new Lapin("Eusebe");

Tortue tog=new Tortue("tog");
Tortue bog=new Tortue("bog");
Tortue zog=new Tortue("zog");

Animal[] animaux = new Animal[7];
animaux[0] = bugs;
animaux[1] = eusebe;
animaux[2] = max;
animaux[3] = roger;
animaux[4] = tog;
animaux[5] = bog;
animaux[6] = zog;

for(int i=0; i<100; ++i)
{
    for(int a = 0; a<animaux.length; ++a)
    {
        animaux[a].avancer();
        animaux[a].afficher();
    }
}
```

5 Bilan

L'héritage permet de regrouper les caractéristiques communes de plusieurs classes. En regroupant, on évite de dupliquer du code et on facilite donc la maintenance d'un logiciel.

Le polymorphisme permet de gérer simplement, de manière homogène, des objets différents ayant un même parent. Le comportement des objets sera distingué à l'aide de méthodes dont le comportement sera diffèrent selon la classe précise de l'objet.