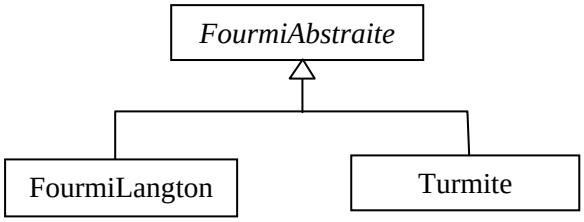
Programmation 1
Examen

Nous nous interressons au déplacements de fourmis artificielles sur un espace. Pour cela nous définissons la hiérarchie de classes ci-contre :



Une fourmi, a une position dans l'espace, une direction et une couleur. L'espace est coloré en noir, et quand une fourmi se déplace elle laisse une trace sur l'espace.

Le principe général de déplacement des fourmis abstraites est le suivant :

1. La fourmi change de direction en fonction du fait que la couleur de la case sur laquelle elle se trouve, est noire ou non.
2. Si la fourmi est sur une case de sa couleur, la case devient noire, sinon la case prend la couleur de la fourmi.
3. Puis, la fourmi avance d'une case dans sa nouvelle direction.

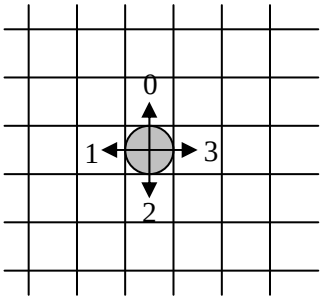
L'espace est représenté par un tableau à deux dimensions d'éléments qui sont des couleurs. On utilisera la classe *Color* pour représenter une couleur. La couleur noire est donnée par *Color.BLACK*. Lorsqu'une fourmi arrive sur un bord, elle repart sur le bord opposé.

Question 1 (2 points)

Définir le constructeur de la classe *FourmiAbstraite*.

Question 2 (3 points)

Définir la méthode *unPas* de la classe *FourmiAbstraite*. Cette méthode implémente le déplacement d'une fourmi, tel qu'il est décrit précédemment. La direction d'une fourmi est codée par un entier de la façon suivante :



Question 3 (3 points)

Définir la méthode *avancer* de la classe *FourmiAbstraite*. Cette méthode fait avancer la fourmi d'une case dans la direction *dir*. Si la fourmi est sur la première ligne du tableau *t*, et que sa direction est 0, la fourmi se retrouve sur la dernière ligne du tableau *t*. Les choses se passent de la même façon sur les autres bords du tableau *t*.

Question 4 (2 points)

Définir la méthode *toString* de la classe *FourmiAbstraite*. Cette méthode retourne une chaîne de caractères construite de la façon suivante : le nom de la classe fourmi, puis entre parenthèses la couleur, la direction et la position de la fourmi, séparés par des virgules.

Pour un objet *o*, l'appel *o.getClass().getName()* retourne une chaîne de caractères qui a pour valeur le nom de la classe de *o*.

Question 5 (4 points)

Une fourmi de Langton est une fourmi qui se déplace de la façon suivante :

- Si la fourmi est sur une case noire, elle tourne de 90° vers la droite, change la couleur de la case en *c* et avance d'une case.
- Sinon, elle tourne de 90° vers la gauche, change la couleur de la case en noir et avance d'une case.

Définir les méthodes de la classe *FourmiLangton*.

Question 6 (4 points)

Une turmite est une fourmi qui possède un état interne qui peut avoir deux valeurs 1 ou 2, et dont le comportement est défini de la façon ci-contre, en fonction de la couleur de la case, et de l'état de la turmite.

	noir	**Non noir**
Etat 1	Tourner 90° à gauche et passer dans l'état 2	Tourner 90° à droite et passer dans l'état 2
Etat 2	Passer dans l'état 1	Passer dans l'état 1

Définir les méthodes de la classe *Turmite*.

Question 7 (2 points)

Ecrire une séquence d'intructions qui définit un espace, place une fourmi de Langton et une turmite sur cet espace, et fait avancer les deux de 1000 pas.

```
public abstract class FourmiA {
   protected int iAnt, jAnt;
   protected int dir;
   protected Color c;
   protected Color [][] t;
   public FourmiA( Color c, int i, int j, int dir, Color[][] t){…}
   public void unPas(){…}
   private void avancer() {…}
   public String toString(){…}
   abstract protected  void tournerNonNoir();
   abstract protected void tournerNoir();

}
```

```
public class FourmiLangton extends FourmiA {
  public FourmiLangton( Color c, int i, int j, int dir, Color[][] t){…}
  public void tournerNonNoir() {…}
  public void tournerNoir() {…}
}
```

```
public class Turmite extends FourmiA {
   int etat;
   public Turmite( Color c, int i, int j, int dir, Color[][] t){…}
   public void tournerNonNoir() {…}
   public void tournerNoir() {…}
}
```