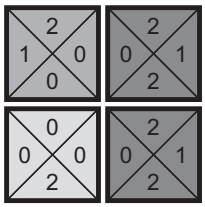
L1 informatique - Année 2022–2023

TD de Concepts Informatique n° 10

Exercice 1. *Pavages.*
Les dominos de Wang sont des carrés de même taille avec un numéro sur chaque bord qui peuvent être agencés bord à bord sur une grille carrée de sorte que deux bords adjacents devront porter le même numéro. Les rotations et symétries de dominos sont interdites.

Le but de cet exercice est d'écrire un programme qui recherche par la méthode du backtracking toutes les façons de remplir une grille carré de côté `dim` avec des copies de dominos issus d'un ensemble `dom` fixé. Un domino sera donné par un tableau de quatre entiers décrivant les numéros portés par les bords Ouest, Nord, Est et Sud dans cet ordre. L'ensemble `dom` sera vu comme un tableau de tels tableaux. Par exemple, avec `int[][] dom0 = {{0,0,0,2},{0,2,1,2},{1,2,0,0}};`, on pourra obtenir une grille de côté 2 comme celle ci-contre.



1. Développer la suite des essais que réalisera typiquement un algorithme de backtracking pour remplir une grille carré de côté 3 avec l'ensemble `dom0` de l'exemple ci-dessus.
2. Écrire une méthode `possibles` qui prend en argument une solution partielle `t`, un entier `dim` et un ensemble `dom` et qui retourne l'ensemble des dominos que l'on peut adjoindre à `t` pour obtenir une autre solution partielle.
3. Conclure en écrivant une méthode `backtrack` et une méthode `main`, le tout dans une classe `Wang`.

Exercice 2. *Révisions de traduction, « Ah... Here we go again ».*
On souhaite traduire le code suivant.

```
1  class Ex2 {
2
3    public static boolean est_pair(int n) {
4      return (n%2 == 0);
5    }
6
7    public static void binaire_inverse(int n) {
8      boolean tmp = false;
9      while (n != 0) {
10       tmp = est_pair(n);
11       if (tmp) {
12         System.out.print(0);
13       }
14       else {
15         System.out.print(1);
16       }
17       n = n/2;
18     }
19     System.out.println();
20   }
21
22   public static void main (String[] args) {
23     binaire_inverse(3);
24     binaire_inverse(6);
25   }
26 }
```

Pour cela, nous disposons des blocs suivants.

```
public class Bloc {
  private int adresseRetour;

  public Bloc(int adr) {
    this.adresseRetour = adr;
  }

  public int getAdresse() {
    return this.adresseRetour;
  }
}
```

```
public class BlocA extends Bloc{
  private boolean valeurRetour;
  private int argument;

  public BlocA(int adr, int arg) {
    super(adr);
    this.argument = arg;
  }

  public int getArg() {
    return this.argument;
  }

  public boolean getVal() {
    return this.valeurRetour;
  }

  public void setVal(boolean val) {
    this.valeurRetour = val;
  }
}
```

```
public class BlocB extends Bloc{
  private int argument;
  private boolean variable;

  public BlocB(int adr, int arg) {
    super(adr);
    this.argument = arg;
  }

  public int getArg() {
    return this.argument;
  }

  public void setArg(int val) {
    this.argument = val;
  }

  public boolean getVar() {
    return this.variable;
  }

  public void setVar(boolean var) {
    this.variable = var;
  }
}
```

1. Qu'affiche le programme lors de son exécution ?
2. Pour quelle fonction `BlocA` sera utile ? Même question pour `BlocB`.
3. Annoter le code
4. Traduire le code