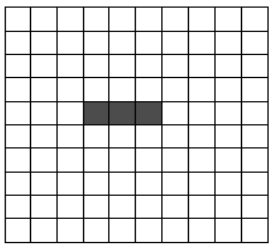
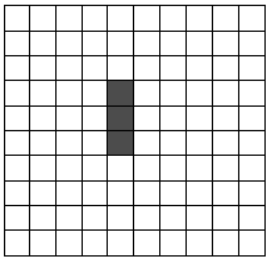
TP n°6 : le jeu de la vie de Conway

Ce « jeu » a été inventé par John Conway en 1970. Il est basé sur une grille composée de cases qui peuvent être dans deux états possibles : vivantes ou mortes. Deux règles permettant de passer d'un état du jeu au suivant, en fonction du nombre de voisins de chaque case :

- Une case morte possédant exactement trois voisines vivantes parmi les 8 devient vivante (elle naît), sinon elle reste morte.
- Une case vivante possédant deux ou trois voisines vivantes le reste, sinon elle meurt.



La grille ci-contre donne un exemple d'état du jeu. Les cases noires sont vivantes et les cases blanches sont mortes. Essayez de comprendre pourquoi l'autre grille représente l'état suivant.



Le but de ce TP est de programmer ce jeu. La grille va être représentée par un tableau à deux dimensions de booléens, de 10 lignes par 10 colonnes. Une case vivante sera représentée par la valeur vraie (`true`) ; une case morte sera représentée par la valeur faux (`false`). Le tableau aura donc des lignes allant de 0 à 9 et des colonnes allant de 0 à 9 :

```
boolean[][] grille = new boolean[10][10];
```

a) Écrire l'action `initializeGrid` permettant d'initialiser la grille avec le premier exemple précédent : initialiser toutes les cases à `false` avec deux boucles imbriquées et mettre `true` dans les 3 cases. Cette action prend en paramètre une grille. Voici son entête :

```
static void initializeGrid(boolean [][] grille) {
```

Pour faciliter le changement de taille de la grille, on va aussi utiliser une constante GSIZE définie tout au début de la classe, avant les actions et fonctions :

```
    final static int GSIZE = 10;
```

b) Pour afficher la grille, nous allons utiliser une classe existante qui permet de créer une fenêtre de pixels et de les colorer. Récupérer le fichier `PixelScreen.java` qui se trouve sur Moodle et le déposer dans votre *workspace*. Voici le programme principal :

```
public static void main(String[] args) {
    boolean[][] grille = new boolean[GSIZE][GSIZE];
    initializeGrid(grille); // initialisation de la grille
    PixelScreen viewer = new PixelScreen(GSIZE,GSIZE,50);
    while(true) {  // on boucle à l'infini
        drawGrid(grille,viewer); // affichage de la grille
        viewer.refreshAfter(200); // on attend 200 millisecondes
        grille=updateGrid(grille); // on recalcule la nouvelle grille
    }
```

N'oubliez pas d'indiquer la bibliothèque qui gère les couleurs : `import java.awt.Color;`

Écrire l'action `drawGrid` permettant d'afficher une grille. Pour chacune des cases de la grille,

il faut appeler l'action `screen.updatePixel` en lui passant le numéro de la ligne, le numéro de la colonne et la couleur : `Color.BLACK` ou `Color.WHITE` selon que la case est vivante ou morte. Vérifiez que votre programme affiche bien l'état initial de la grille.

c) Il faut maintenant calculer l'état suivant d'une grille. Écrire une fonction qui calcule le nombre de voisins vivants d'une case donnée. Cette fonction a besoin de 3 paramètres. Lesquels ? Quel est le type de la valeur renvoyée par la fonction ? Attention quand vous accédez à la case immédiatement à gauche si vous êtes sur le bord de la grille (i=0). Le mieux est de ne comptabiliser le voisin de gauche que si i>0. Même chose pour les autres bords de la grille.

d) Écrire la fonction `updateGrid` qui calcule le nouvel état de la grille : elle prend en paramètre une grille et renvoie une autre grille. Pour cela, définissez une seconde grille résultat (donc également de type `boolean [][]`) et calculez l'état de chacune de ses cases en utilisant la fonction précédente.

e) Changez l'état initial à votre guise et observez les nouveaux états. En particulier, vous pouvez essayer le planeur (figure ci-contre) qui redevient à l'identique tous les 4 états, mais décalé d'un cran. Vous pouvez aussi travailler sur une grille plus grande que 10 pixels.