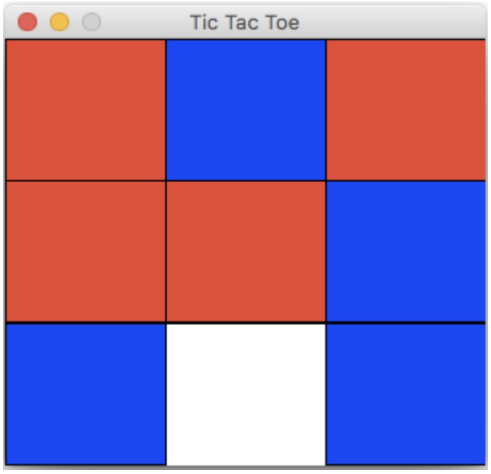
TP 2 – Conception et programmation objet

Notions mobilisées: classes anonymes, patterns Observer et MVC (ou plutôt MV)

Tic Tac Toe

Vous allez programmer une version simple du jeu TicTacToe (Morpion). L'affichage sera simpliste, initialement le jeu commence avec une grille blanche et les joueurs à tour de rôle cliquent sur une case. Les cases choisies par le premier joueur se colorient en rouge, celle du deuxième joueur se colorient en bleu, les cases qui n'ont pas encore été jouées restent blanches.



1. Téléchargez les fichiers disponibles sur Eprel, puis exécutez et lisez attentivement le code. La classe TicTacToe contient un tableau `board` à valeurs dans {0, ... ,2}, il nous permettra d'enregistrer les tour des joueurs : on marquera bord[i][j] = 1 (resp. board[i][j] = 2) si le joueur 1 (resp. le joueur 2) a clické sur la case (i,j) du jeu.
2. La méthode `colorBox(Graphics g, int i, int j)` est définie pour colorier des cases du tableau de jeu en fonction des coordonnées (i,j) d'un pixel. En utilisant colorBox, définissez une méthode `private void colorBoard(Graphics g, int[][] tab)` qui vous permettra de colorier les cases du jeu en fonction des valeurs d'un tableau, une case rouge si tab[i][j] = 1, bleu si tab[i][j]=2, la case reste blanche sinon.
3. En utilisant une classe anonyme, ajoutez un listener MouseAdapter et re-définissez la méthode mouseClicked(MouseEvent e) afin que lorsque un joueur clique sur la fenêtre de jeu, des nouveaux valeurs sont ajoutées au tableau `board` en fonction du jouer et de la position du click: board[i][j]= 1 si c'était le tour du jouer 1 et il a cliqué sur la case (i, j), board[i][j]=2 si c'était le tour du jouer 2 et il a cliqué sur la case (i,j). N'oubliez pas de mettre à jours le tableau de jeu.
4. On veut créer deux fenêtres, l'une pour le joueur 1 et la deuxième pour le joueur 2. Pour cela nous allons appliquer le pattern Observer. Créez une interface `IObserver` avec une méthode abstraite public void `update()`. Puis créez une classe `TicTacToeObservable` qui comportera une liste d'observateurs et des méthodes pour enregistrer et notifier les observateurs. La classe TicTacToeObservable doit comporter également des attributs privés int[][] board et int turn, ainsi qu'une méthode `public void makeMove(int x, int y)` qui écrit sur board[x][y] la valeur 1 si c'est le tour du joueur 1, la valeur 2 si c'est le tour du joueur 2; passe le tour au jouer suivant et notifie tous les observers du changement.
5. Maintenant vous allez définir une view. Pour cela, adaptez le code de la classe TicTacToe pour définir une classe TicTacToeView qui implémentera l'interface IObserver. L'implémentation de la

méthode update() consistera en un repaint. TicTacToeView ne doit pas manipuler directement le board ni le tour de jeux (cela est la responsabilité de la classe TicTacToeObservable), mais elle doit s'enregistrer dans la liste des observateurs de TicTacToeObservable. Modifiez également le constructeur de la classe TicTacToeView afin de pouvoir customiser le titre et la position de la fenêtre (avec frame.setTitle(String title) et frame.setLocation(int n, int m)).

6. Modifiez le main comme suit, puis lancez le jeu.

```
TicTacToeObservable TICTACTOE = new TicTacToeObservable();

new TicTacToeView(300, TICTACTOE, "Player 1", 100, 100 );
new TicTacToeView(300, TICTACTOE, "Player 2", 500, 100 );
```

Si vous avez programmé le jeu correctement, vous visualiserez deux fenêtres de jeu, chaque joueur pourra jouer sur sa fenêtre.