Interfaces graphiques avec Swing et programmation événementielle

Exercice 1 : Morpion et UML

L'objectif de cet exercice est d'utiliser UML pour modéliser le jeu du Morpion.

Le jeu de Morpion est un jeu à deux qui se joue sur une grille de taille 3 cases par 3. Les cases de la grille sont initialement vides. Chaque joueur joue en alternance en inscrivant un symbole dans une case vide (l'un des joueurs est représenté par un anneau, l'autre joueur par une croix). Ce sont les croix qui commencent. Le jeu s'arrête dès que l'un des joueurs aligne trois symboles identiques (horizontalement, verticalement ou en diagonale) ou lorsque toutes les cases sont occupées.

1.1 Proposer un diagramme des cas d'utilisation.

1.2 Dessiner un diagramme de séquence contextuel qui montre le déroulement d'une partie. Il devra montrer que les règles du jeu doivent être respectées.

1.3 Lister les événements extérieurs, ceux déclenchés par l'utilisateur de l'application.

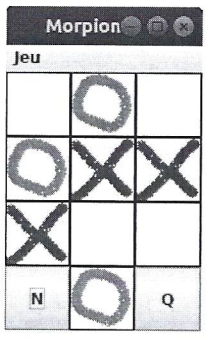
1.4 Dessiner un diagramme d'état du jeu de Morpion.

Exercice 2 : IHM graphiques pour le jeu du Morpion

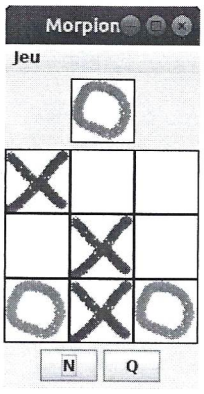
On envisage deux IHM (Interface Homme Machine) pour le Jeu du Morpion présentées figures 1(a) et 1(b).

2.1 Indiquer les composants graphiques à utiliser.

2.2 Expliquer comment construire la partie présentation des deux IHM envisagées.



(a) version 1



(b) version 2

Exercice 3 : Différentes manières de réaliser les réactions du Morpion

Plusieurs solutions sont possibles pour programmer les réactions dans le jeu du Morpion. L'objectif de cet exercice est d'identifier ces solutions et d'indiquer leurs avantages et inconvénients.

3.1 Considérons d'abord les deux boutons Quitter et Recommencer. Expliquer comment il serait possible d'implanter les réactions correspondantes.

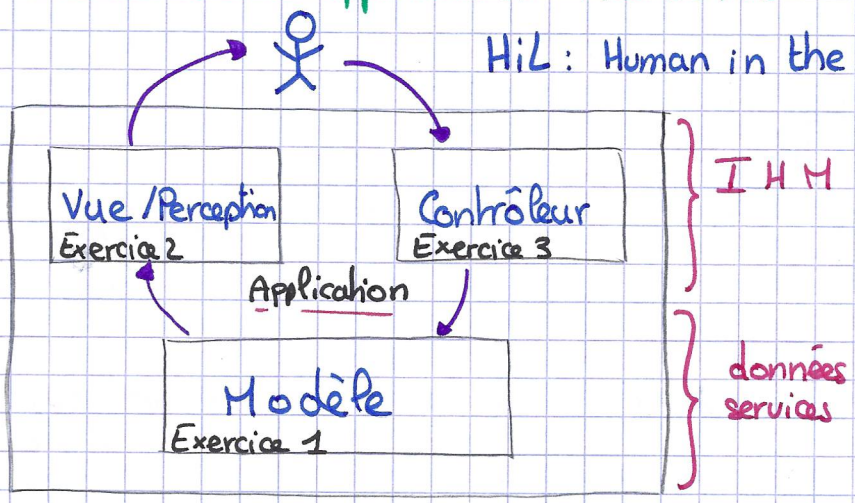
3.2 Les cases de l'aire de jeu étant des JLabel, la réaction doit être définie comme un MouseListener. Discuter les avantages et inconvénients à utiliser MouseListener ou MouseAdapter.

3.3 Lorsque le joueur clique dans une case, il faut que dans la réaction on puisse retrouver les coordonnées de la case cliquée (ceci peut être nécessaire pour déterminer de manière efficace si la partie est terminée). Expliquer comment connaître cette information.

TD 1
TOB

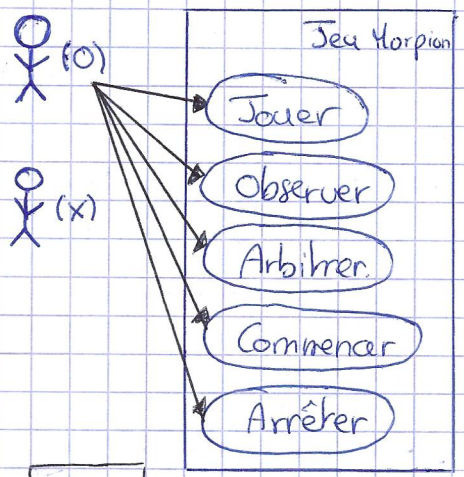
Interfaces graphiques avec Swing et programmation évènementielle

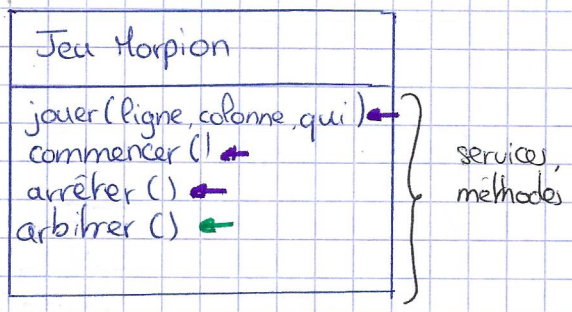
Eléments d'une application interactive



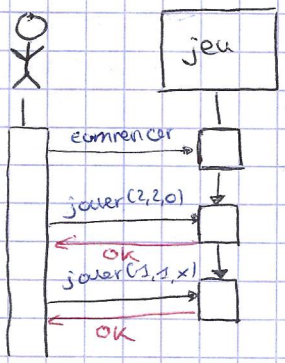
Exercice 1 : Morpion et UML

1.1 /

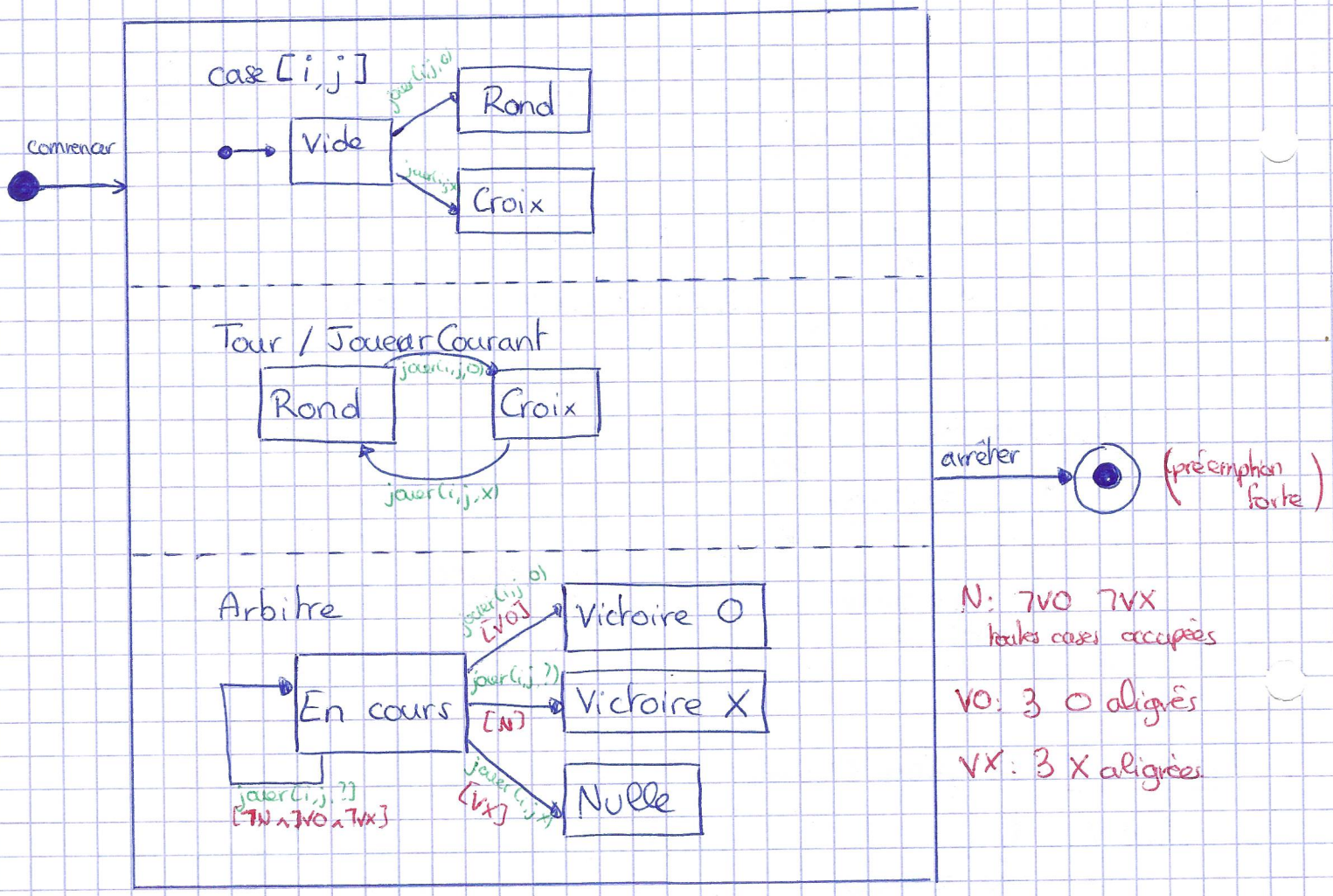




évènements extérieurs
évènements internes

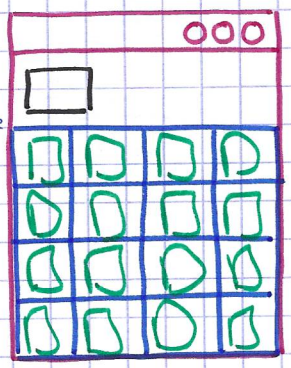


1.4 / Machine à états décrivant le comportement du jeu de Morpion

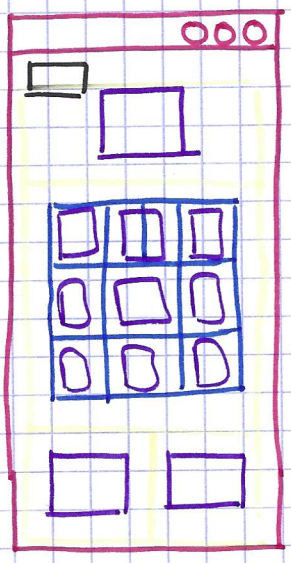


Exercice 2: IHM graphiques pour le jeu du Morpion

1.1/ Des fenêtres: JFrame
Des grilles: GridLayout
Des boutons: JButton, JLabel (dessin spécifique)
Un menu: Jmenu
Un layout:



Grid Layout associé à la JFrame



Exercice 3: Différentes manières de réaliser les réactions du Morpion.

```
public class Morpion extends JFrame {
    JButton bQuitter, bNouveau;
    public Morpion () {
        super();
        bQuitter = new JButton();
```

TD 9.2
TOB

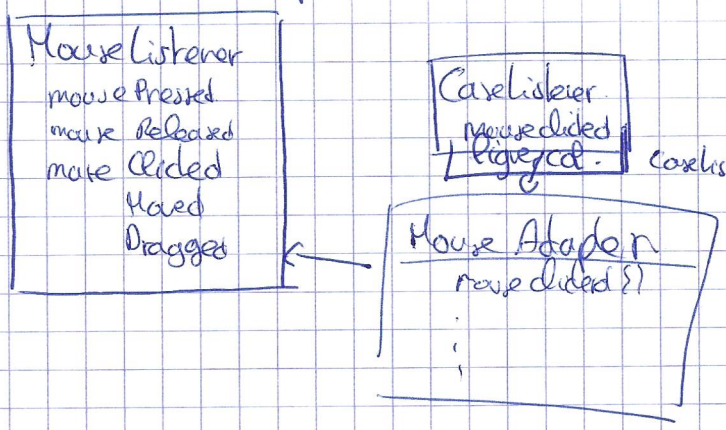
```
bNouveau = new JButton ();
bQuitter.add (new ActionListener () {
                public void action () {
                    System.exit(0);
                }
});
```

enregistre observateur du bouton Quitter

Classe anonyme implantant ActionListener

① new QuitterListener()

```
public class QuitterListener implements ActionListener {
    public void action() { System.exit(0); } }.
```




```
void mouseClicked (MouseEvent evt) {
    // Comment identifier les coord de JLabel ?
```

```
for(i=0; i<=2; i++) {
    for (j=0; j<=2; j++) {
        JLabel case = new JLabel();
        case.add (new CaseListener(i,j))
```