Programmation Événementielle : TM 1

Exercice 1 : Terminer un programme (5mins)

En partant du code de Morpion du CM 1, ajoutez le code nécessaire à ce que le programme se termine lorsque toutes les cases sont pleines. Pour cela, utilisez simplement un compteur qui vérifie que 9 coups ont été joués et appelez `System.exit(0)`.
NB1 : Faites attention à gérer la non-finalité comme il faut.
NB2 : NE FAITES PAS les cas de victoire par alignement ! C'est facile mais long, donc faites le plutôt chez vous.

Exercice 2 : Une application à deux fenêtres (20mins)

Modifiez le code de Morpion obtenu pour avoir 2 fenêtres (un `JFrame` par joueur), et pour que le click d'un joueur ne soit pris en compte que lorsque c'est son tour.

Exercice 3 : Puissance 4 (20mins)

Programmez le jeu Puissance 4 (voir Wikipédia) en partant d'un fichier vide, en limitant les copier-coller et recherche internet (sans la gestion de la victoire comme précédemment).
NB : Il y a deux différences avec le Morpion : la taille de la grille (6 lignes et 7 colonnes), et le fait que lorsque l'on clique sur un bouton, cela n'indique que la colonne où l'on souhaite jouer, donc il faut déterminer la bonne case à modifier en respectant la chute d'une pièce.

Exercice 4 : Gérer soi-même l'affichage (30mins)

Dans le fichier `ClickDraw.java` associé au TM, vous trouverez un programme Java/Swing simple qui réagit aux cliques pour déplacer une cible. Tenter de comprendre ce qu'il fait et globalement comment il le fait. En vous en inspirant pour refaire le jeu du Morpion avec un graphisme plus agréable, sans aucune utilisation de `JButton`.

Exercice 5 : Compréhension de code (5mins)

Entièrement sur papier et sans le moindre ordinateur, dites ce le code suivant s'affiche.

```
interface A { public void a(); }
class B {
  A x; int c = 0;
  B(A x) { this.x = x; }
  void go() { if (c<2) { x.a(); c++; } }
}
void main() {
  var y = new B(() -> { IO.println("hey !"); });
  for(int i=0; i<5; i++) y.go();
}
```

(NB: Pour exécuter ce code, placez le dans un fichier `Mano.java` et exécutez le dans un terminal avec `java Mano.java` dans un terminal (fonctionne avec Java 25)).

Exercice 6 : Non-finalité de variable dans les lambdas (7mins)

Dans le CM1, deux astuces ont été présentées pour gérer l'utilisation interdite d'une variable de contexte non-finale dans une lambda.

La première était de faire une copie (c'est là qu'on a créé les variables `lFinal` et `cFinal`), et le second est l'utilisation d'une référence (c'est là qu'on est passé de `int j = 1;` à `var j = new int[] { 1 };`). Il est important de réaliser quand est-ce que l'on veut faire l'eux ou l'autre.

Afin de vérifier votre compréhension, essayer de prédire ce qui se passerait de bizarre si l'on prenait le code du cours et que l'on transformait plutôt les variables `l` et `c` en tableau d'une case (donc en référence vers un entier comme on a fait pour `j`) au lieu d'un faire des copies `lFinal` et `cFinal`.