UNIVERSITÉ DES MASCAREIGNES

SAVOIR, C'EST POUVOIR

Université de Limoges

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGIES

DEPARTMENT APPLIES INFORMATICS – YEAR 2

2021 - 2024

JavaFx

TP – Snake

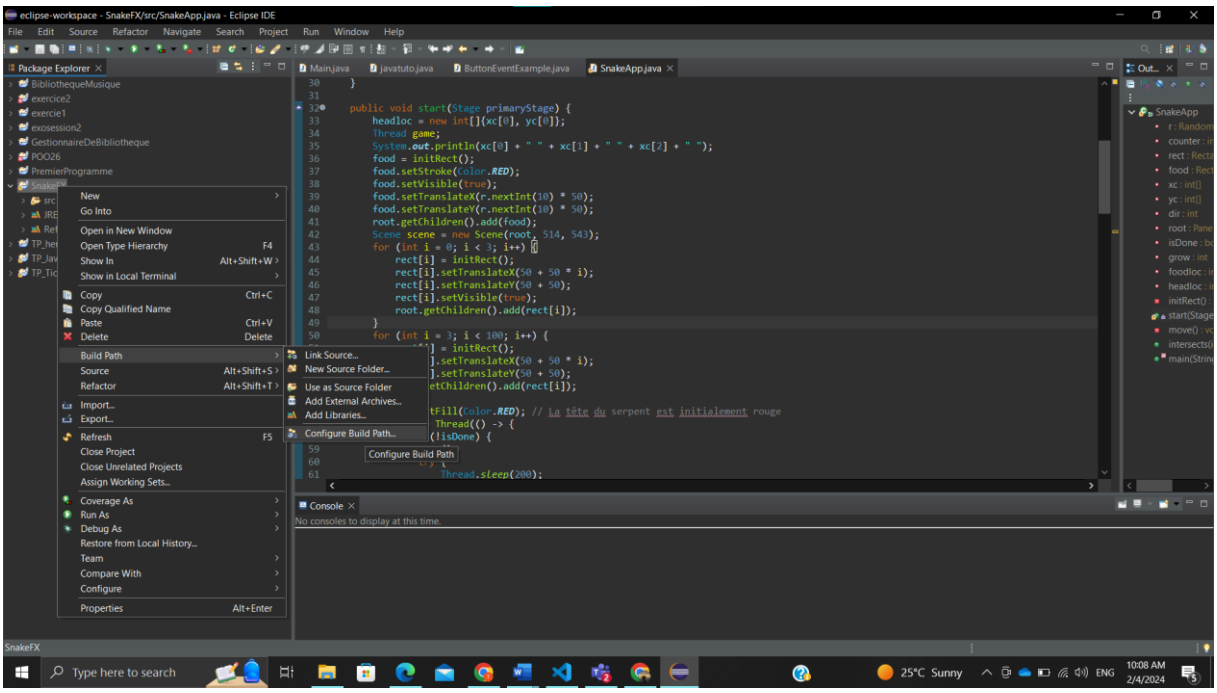
Présentée par : Nasandratsoa Patricia HANI RAKOTOASIMBOLA

Encadrée par : Mr Khadimoullah Ramoth

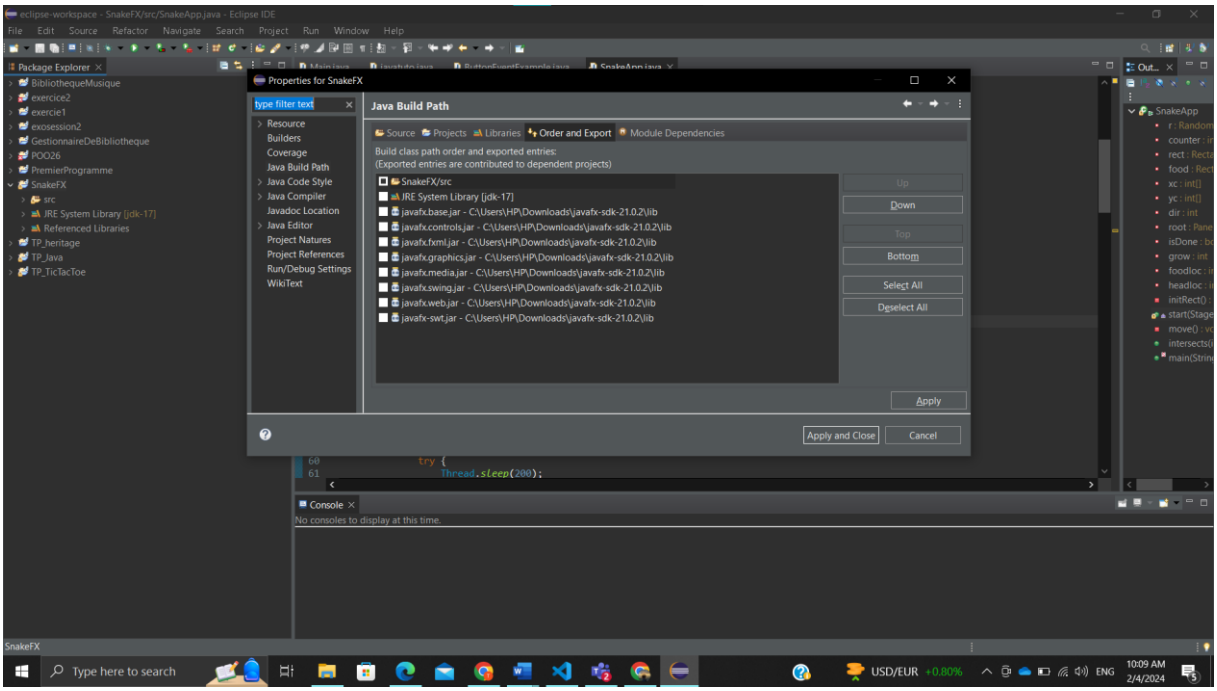
1. Configuration JavaFX :

Après avoir télécharger le javaFX-sdk et et d'avoir extrait le dossier, on passe à la configuration :

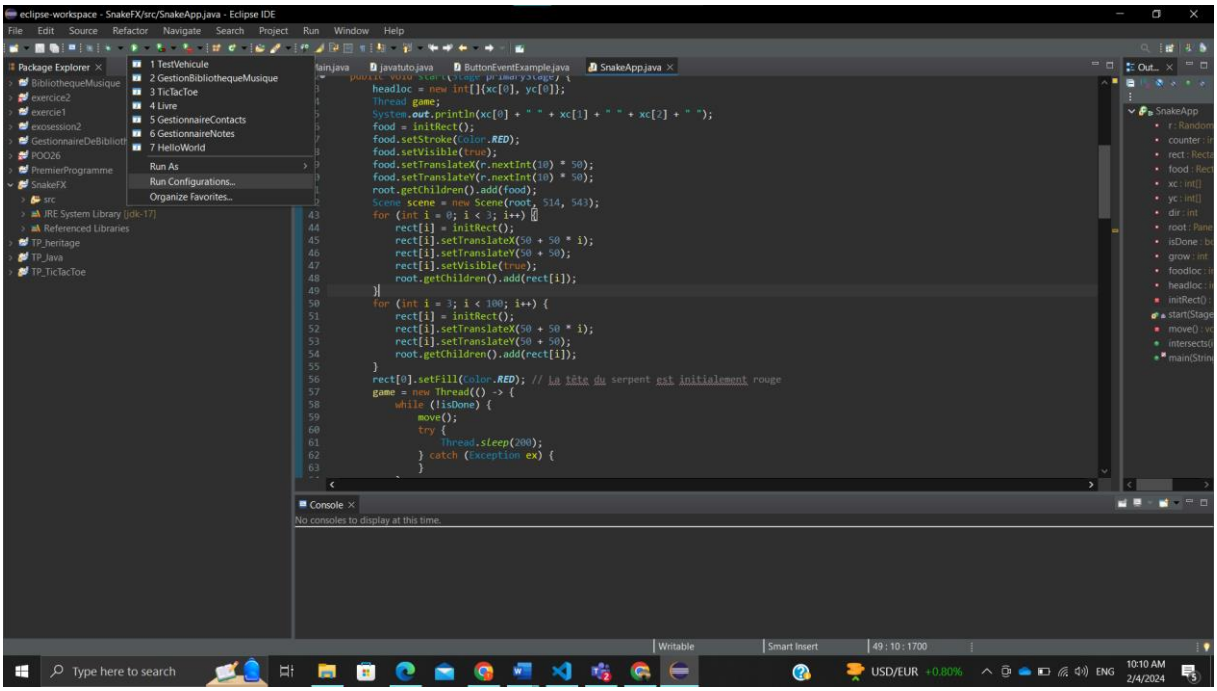
- D'abord on crée le projet SnakeFX
- On fait une clique droite -> Build path -> configure build path



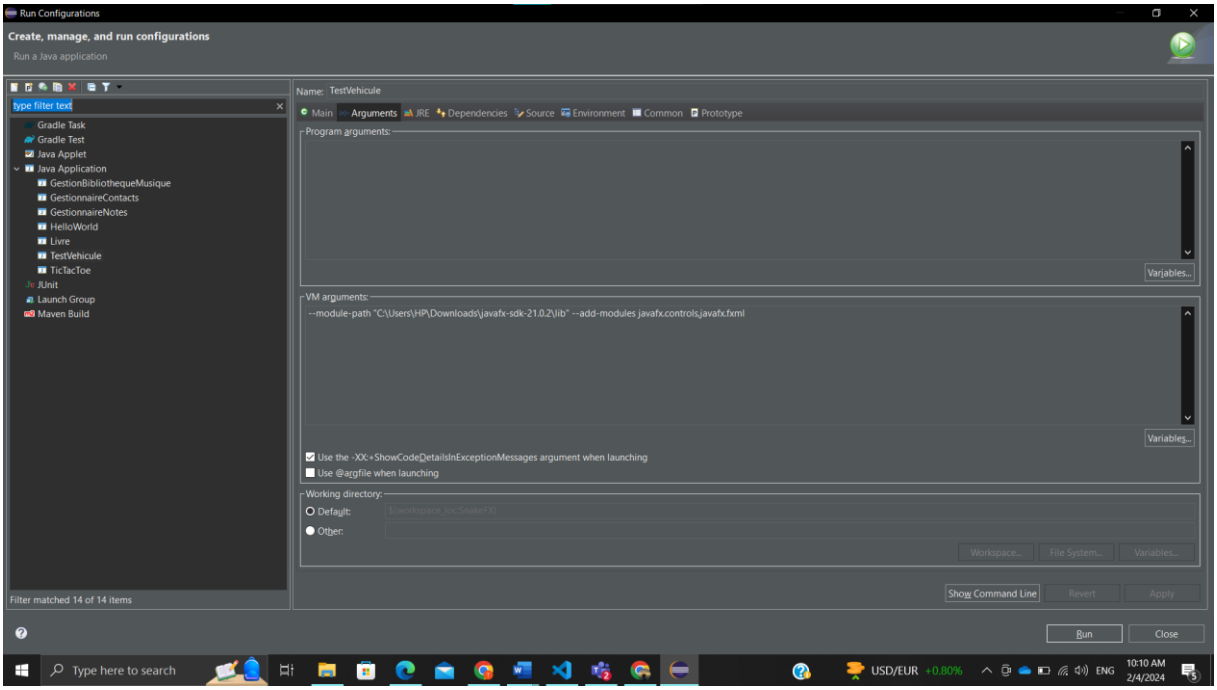
Après avoir fait ça, il y a un onglet qui s'ouvre, On colle les fichier venant du dossier extrait(que l'on vient de télecharger) ici dans le add external Jars et ensuite click sur Apply :



Ensuite, on va faire la configuration du VM argument, donc on va run le projet SnakeFX comme ce qui suit :



On colle ensuite le chemin vers le dossier dans le VM arguments :



Et ce n'est qu'après avoir fait cela qu'on run et continuer à coder.

2. Explication et fonctionnement du code :

```
private Rectangle initRect() {
    Rectangle res = new Rectangle(45, 45);
    res.setFill(Color.BLACK);
    res.setStroke(Color.BLACK);
    res.setVisible(false);
    return res;
}
```

La classe Rectangle est utilisée pour représenter chaque segment du serpent

```
private Pane root = new Pane();
```

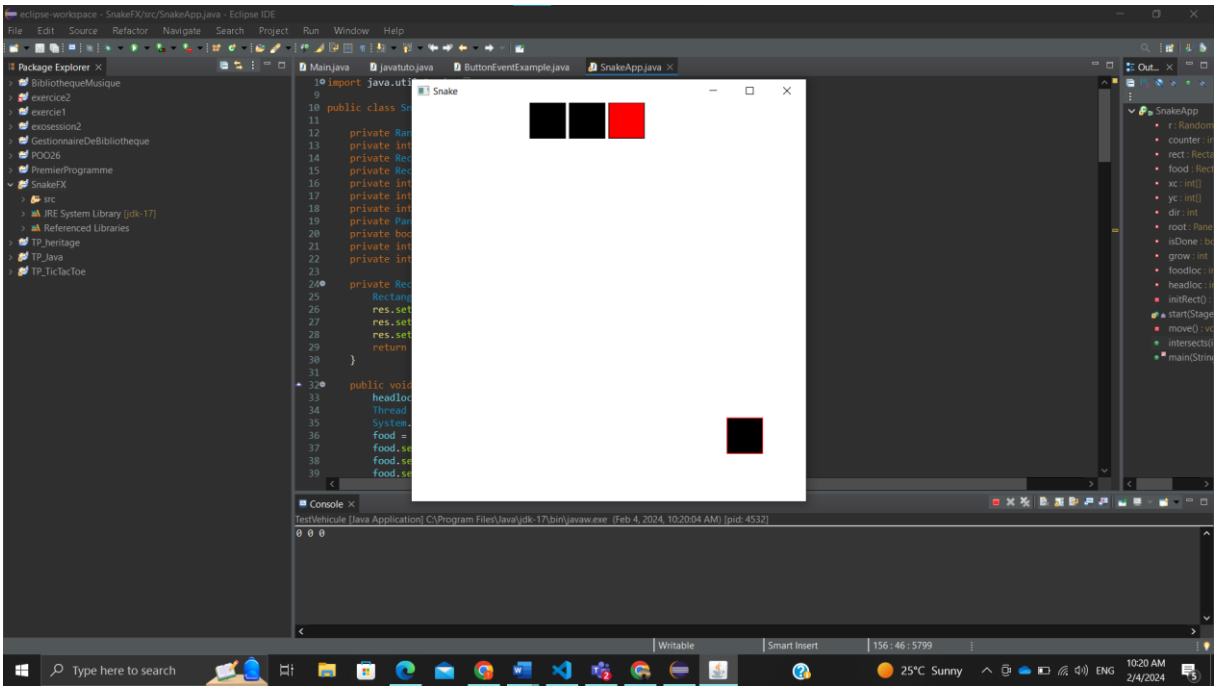
Pane est utilisé comme conteneur racine pour organiser les éléments

```
Scene scene = new Scene(root, 514, 543);
for (int i = 0; i < 3; i++) {
    rect[i] = initRect();
    rect[i].setTranslateX(50 + 50 * i);
    rect[i].setTranslateY(50 + 50);
    rect[i].setVisible(true);
    root.getChildren().add(rect[i]);
}
for (int i = 3; i < 100; i++) {
    rect[i] = initRect();
    rect[i].setTranslateX(50 + 50 * i);
    rect[i].setTranslateY(50 + 50);
    root.getChildren().add(rect[i]);
}
rect[0].setFill(Color.RED); // La tête du serpent est initialement rouge
game = new Thread(() -> {
    while (!isDone) {
        move();
        try {
            Thread.sleep(200);
        } catch (Exception ex) {
        }
    }
});
```

C’est l’initialisation du Serpent à l’output, la tête du serpent est en rouge et le reste en noir,

```
food.setStroke(Color.RED);
```

le bouton qui ne se relie pas avec le serpent est le « food » qui fait grandir le serpent à chaque foit qu’il mange :

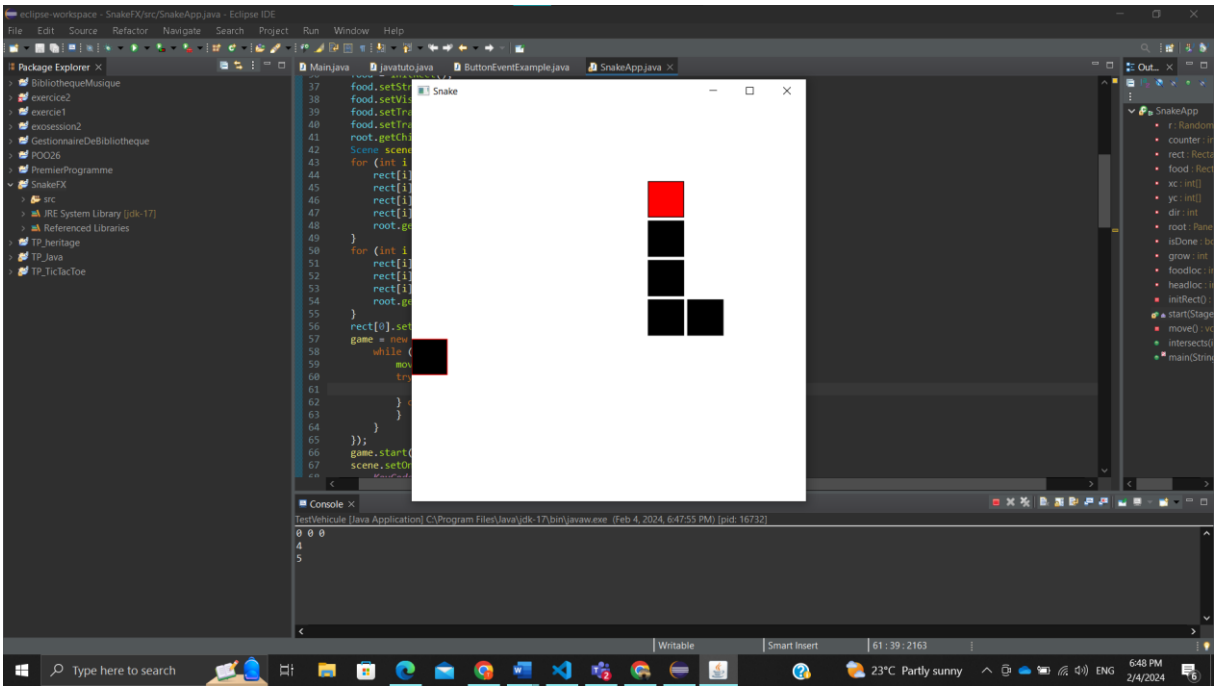



```
game.start();
scene.setOnKeyPressed(event -> {
    KeyCode k = event.getCode();
    switch (k) {
        case D:
            if (dir != 1 && ((dir == 2 || dir == 0) && headloc[1] != yc[0])) dir = 3;
            break;
        case A:
            if (dir != 3 && ((dir == 2 || dir == 0) && headloc[1] != yc[0])) dir = 1;
            break;
        case S:
            if (dir != 2 && ((dir == 3 || dir == 1) && headloc[0] != xc[0])) dir = 0;
            break;
        case W:
            if (dir != 0 && ((dir == 3 || dir == 1) && headloc[0] != xc[0])) dir = 2;
            break;
    }
    headloc = new int[]{xc[0], yc[0]};
});
```

Cette partie du code illustre que les touches qui montrent les movements du joueur sont ASDZ :

- ✓ A : à gauche
- ✓ S : en bas
- ✓ D : à droite
- ✓ Z : en haut

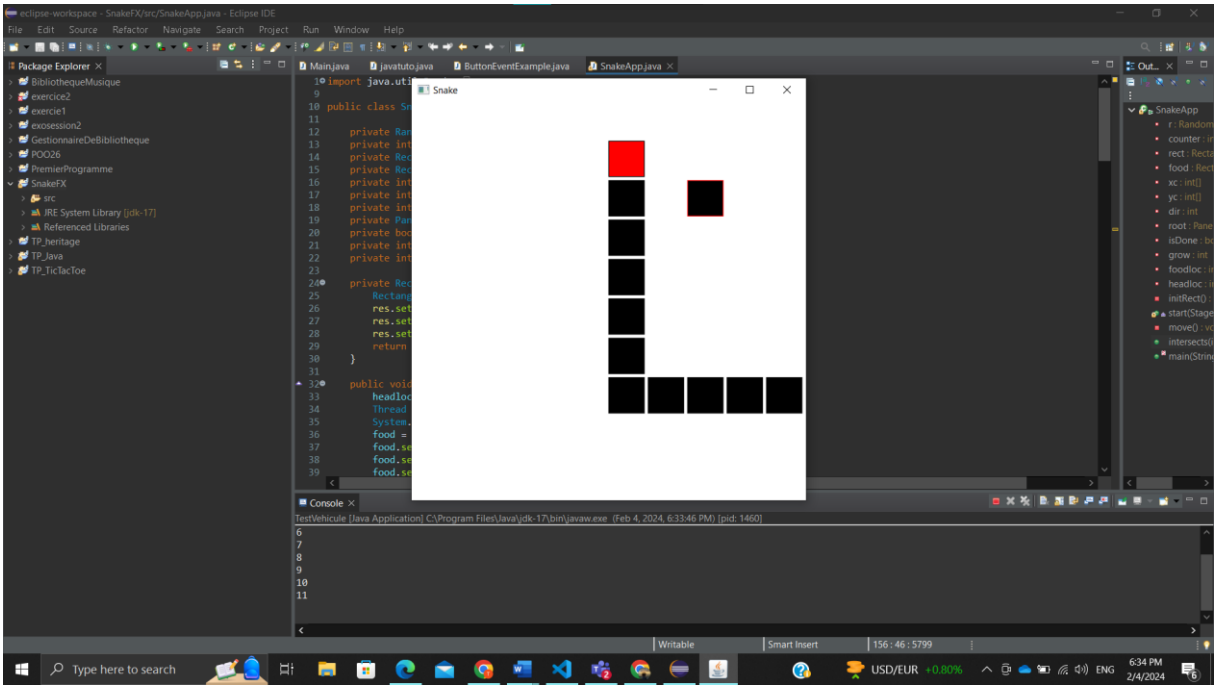
Et le seprent change en place par ces touches :




```java
private void move() {
    int[][] tmp = {{xc[1], yc[1]}, {}};
    xc[1] = xc[0];
    yc[1] = yc[0];
    for (int i = 2; i < counter; i++) {
        tmp[1] = new int[]{xc[i], yc[i]};
        xc[i] = tmp[0][0];
        yc[i] = tmp[0][1];
        tmp[0] = tmp[1];
        if (grow > 0 && xc[counter - 1] == foodloc[0] && yc[counter - 1] == foodloc[1]) {
            rect[counter - 1].setVisible(true);
            --grow;
        }
    }
    switch (dir) {
        case 0:
            if (yc[0] + 50 <= 450) yc[0] += 50;
            else yc[0] = 0;
            break;
        case 1:
            if (xc[0] - 50 >= 0) xc[0] -= 50;
            else xc[0] = 450;
            break;
        case 2:
            if (yc[0] - 50 >= 0) yc[0] -= 50;
            else yc[0] = 450;
            break;
        case 3:
            if (xc[0] + 50 <= 450) xc[0] += 50;
            else xc[0] = 0;
            break;
    }
```

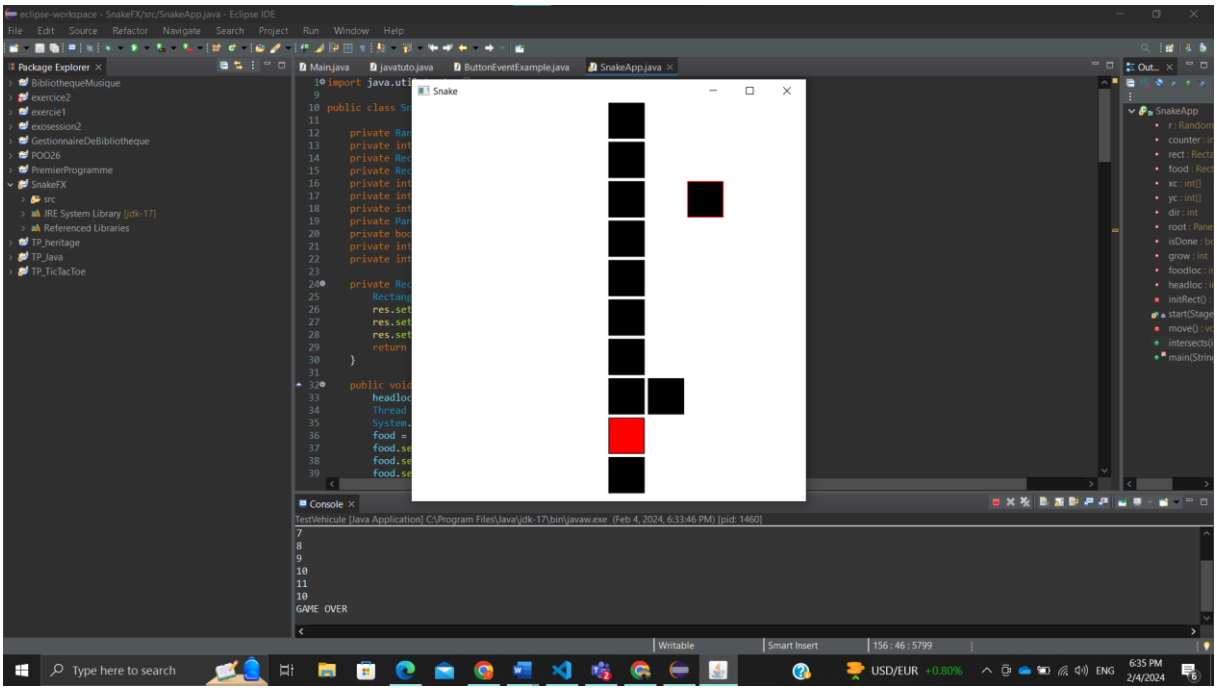
```
        break;
}
if (intersects(xc[0], yc[0])) {
    System.out.print("GAME OVER");
    isDone = true;
} else {
    for (int i = 0; i < counter; i++) {
        rect[i].setTranslateX(xc[i]);
        rect[i].setTranslateY(yc[i]);
    }
    if (rect[0].getBoundsInParent().intersects(food.getBoundsInParent())) {
        boolean isChanged = false;
        foodloc = new int[]{(int) food.getTranslateX(), (int) food.getTranslateY()};
        while (!isChanged) {
            food.setTranslateX(r.nextInt(10) * 50);
            food.setTranslateY(r.nextInt(10) * 50);
            if (intersects(foodloc[0], foodloc[1]) && (int) food.getTranslateX() != foodloc[0] || (int) food.getTranslateY() != foodlo
                isChanged = true;
        }
        xc[counter] = (int) food.getTranslateX();
        yc[counter] = (int) food.getTranslateY();
        rect[counter].setTranslateX(rect[counter - 1].getX());
        rect[counter].setTranslateY(rect[counter - 1].getY());
        ++counter;
        ++grow;
        System.out.println(counter + "");
    }
}
```

Ces codes sont ce qui font que le serpent grandit à chaque fois qu'il mange le point noir, et on peut voir ici que le serpent depuis de plus en plus long :




```
            break;
        }
        if (intersects(xc[0], yc[0])) {
            System.out.print("GAME OVER");
            isDone = true;
        } else {
```

Et quand le serpent se colle entre son propre corps, la partie est terminée : GAME OVER


Snake
Console
7
8
9
10
11
10
GAME OVER