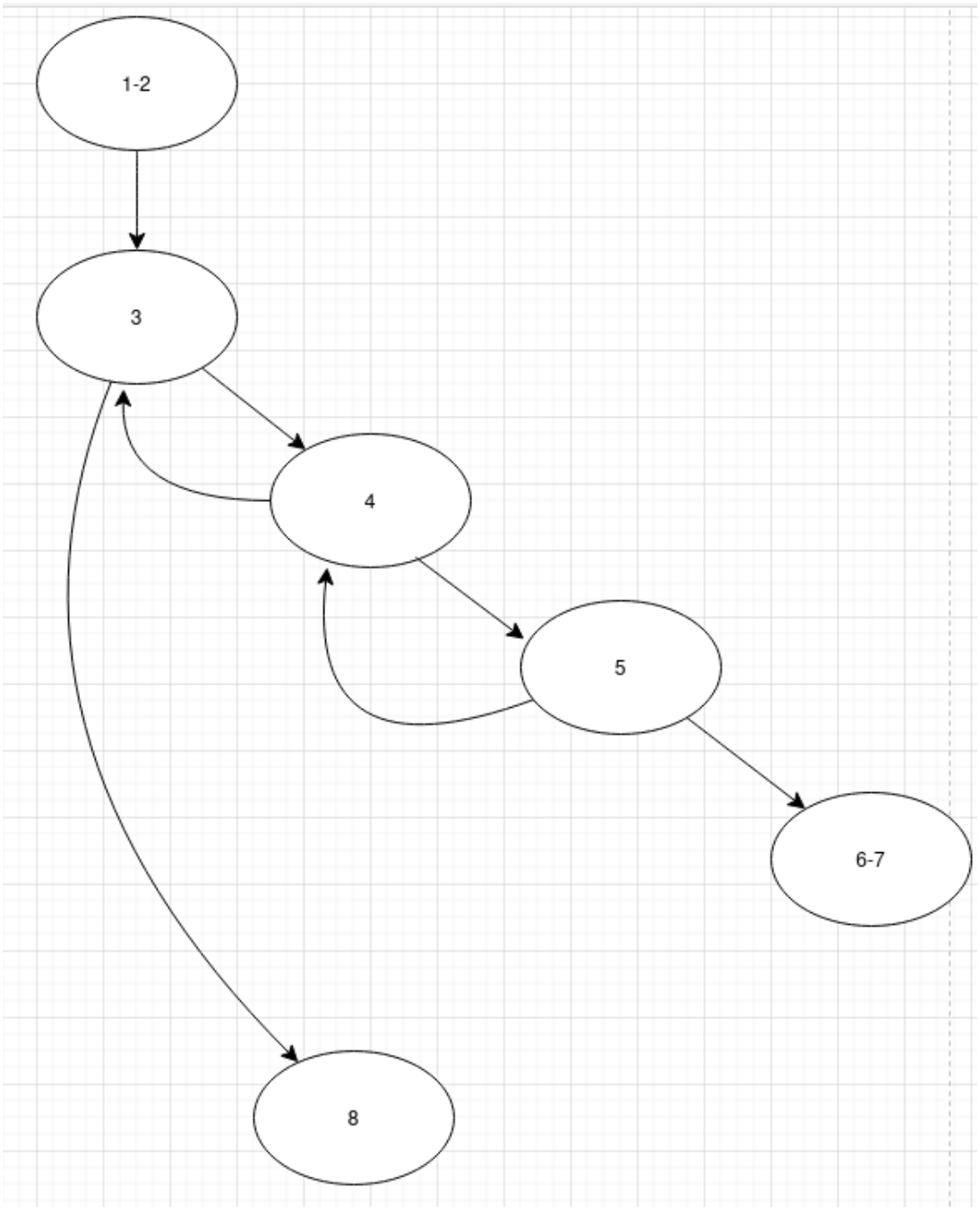
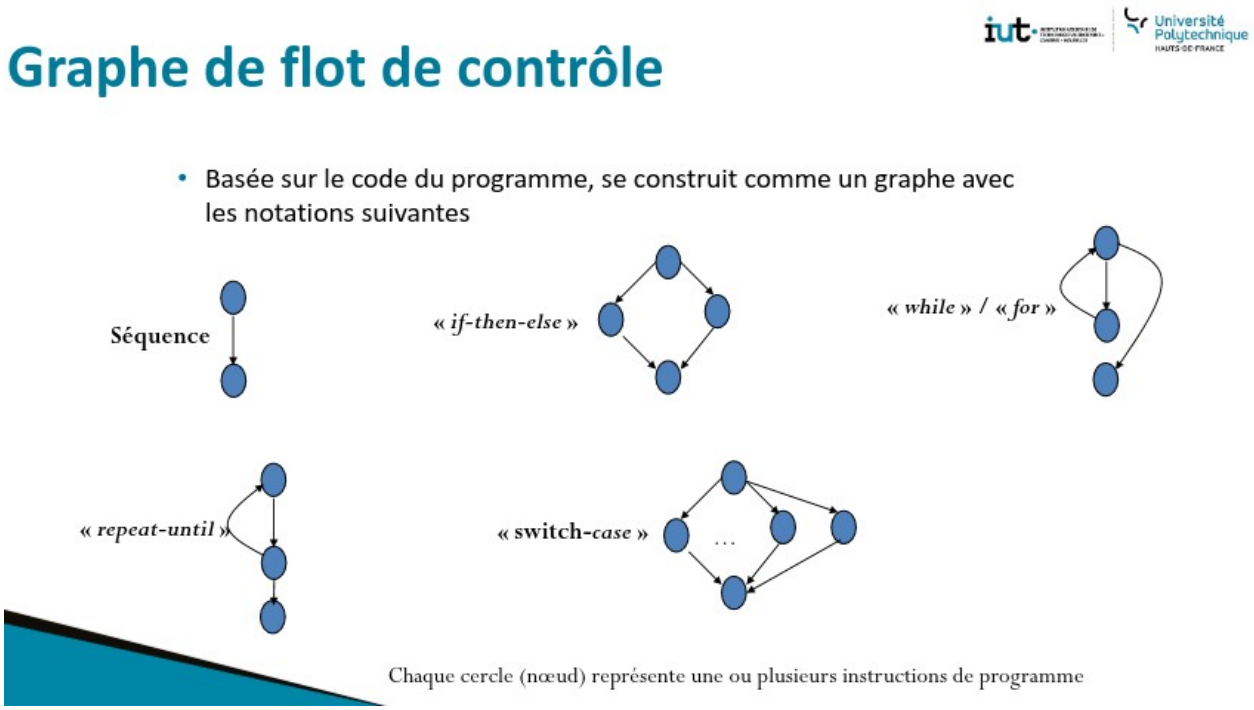
TEST BOITE BLANCHE MÉTHODE ESTCONNEXE :

Dans ce document, vous trouverez le test de boîte blanche de la méthode «estConnexe », qui permet de vérifier la connexité d'un labyrinthe.
Pour cela, nous avons réalisé un graphe de flot de contrôle représentant le déroulement du code, à l'aide de cercles illustrant chaque étape de l'exécution.
Je vous présente ci-dessous ce graphe de flot de contrôle.



Pour réaliser ce graphe de flot de contrôle, je me suis aidé du cours vu avec Mme Oliveira, et plus particulièrement du cours de Techniques de Tests, où j'ai pu revoir comment construire une boucle «`for`». Voici un extrait de cette partie du cours :



Maintenant que nous avons le graphe de flot de contrôle, nous pouvons choisir l'un des quatre critères de test : l'instruction, la branche, le chemin ou la condition.
Dans notre cas, nous avons décidé de choisir le critère de branche, ce qui signifie que nous devons trouver le minimum de chemins permettant de parcourir au moins une fois chaque branche du code.
Le tableau suivant présente ces chemins :

Critère Branche :

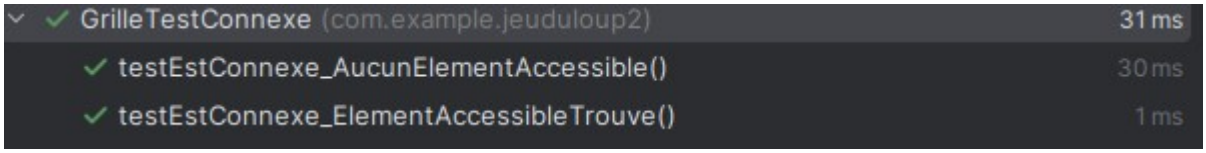
Critères	y	nbLignes	X	NbColonnes	elements[x][y]	Elements[x][y].isAccessible()	Valeur attendu
[1-2,3,8]	1	5	1	5	Rocher(1,1)	False	false
[1-2,3,4,5,6-7]	1	5	1	5	Herbe(1,1)	True	True

Après cela, j'ai pu effectuer mes tests sur IntelliJ, ce qui m'a permis d'obtenir ce code :

```
package com.example.jeuduloup2;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

import java.lang.reflect.Field;

class GrilleTestConnexe {

    private Grille grille;
    private Elements[][] elements;
    private int nbLignes = 5;
    private int nbColonnes = 5;

    @BeforeEach
    void setUp() throws Exception {
        grille = new Grille(nbColonnes, nbLignes);

        Field elementsField = Grille.class.getDeclaredField("elements");
        elementsField.setAccessible(true);
        elements = (Elements[][]) elementsField.get(grille);

        Field nbLignesField = Grille.class.getDeclaredField("nbLignes");
        nbLignesField.setAccessible(true);
        nbLignesField.set(grille, nbLignes);

        Field nbColonnesField = Grille.class.getDeclaredField("nbColonnes");
        nbColonnesField.setAccessible(true);
        nbColonnesField.set(grille, nbColonnes);
    }

    @Test
    void testEstConnexe_AucunElementAccessible() {
        elements[1][1] = new Rocher(1, 1);

        for (int y = 0; y < nbLignes; y++) {
            for (int x = 0; x < nbColonnes; x++) {
                if (!(x == 1 && y == 1)) {
                    elements[x][y] = null;
                }
            }
        }

        boolean resultat = grille.estConnexe();
        assertFalse(resultat);
    }

    @Test
    void testEstConnexe_ElementAccessibleTrouve() {
        elements[1][1] = new Herbe(1, 1);

        for (int y = 0; y < nbLignes; y++) {
```

```
            for (int x = 0; x < nbColonnes; x++) {
                if (!(x == 1 && y == 1)) {
                    elements[x][y] = new Rocher(x, y);
                }
            }
        }

        boolean resultat = grille.estConnexe();
        assertTrue(resultat);
    }
}
```

Une fois le code lancé, j'ai obtenu le résultat suivant :



Cela montre que notre jeu de tests est bien terminé, car nos deux tests sont fonctionnels.

Conclusion :

En conclusion, les résultats obtenus valident notre jeu de tests basé sur le critère de branche, en garantissant que chaque branche du code a bien été couverte par au moins un scénario de test.