Livrable 1.04 — Rapport de test JUnit

Classe testée : Mouton

Méthode principale : manger(Vegetal vegetal)

But : Vérifier que le mouton adapte bien sa vitesse après avoir mangé un végétal.

Critère utilisé : Couverture d'instruction (chaque ligne de la méthode est exécutée au moins une fois)

Id Test	Végétal donné en entrée	Vitesse attendue après appel	Critère de test
T1	new Herbe()	2	Instruction
T2	new Marguerite()	4	Instruction
T3	new Cactus()	1	Instruction

Classe testée : Loup

Méthodes : getVitesse(), getPosition(), setPosition(Position)

But : Vérifier que :

- La vitesse du loup est constante
- Sa position est correctement enregistrée et modifiable

Critère utilisé : Couverture d'instruction (chaque méthode exécutée))

ID	Scénario de test	Données d'entrée	Résultat attendu	Critère
T4	**Vérification de la vitesse du loup**	**new Loup(pos)**	**vitesse = 3**	**Instruction**
T5	**Vérifie la position initiale**	**new Position(2,2)**	**.getPosition( ) = (2,2)**	**Instruction**
T6	**Changement de position**	**setPosition(5,5)**	**position = (5,5)**	**Instruction**

Code JUnit pour les test:

Mouton:

```java
package com.sae.moutonloup.model;


import com.sae.moutonloup.model.*;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;


public class MoutonTest {


    @Test
    public void testMangerHerbe() {
        Mouton m = new Mouton(new Position(0, 0));
        m.manger(new Herbe());
        assertEquals(2, m.getVitesse());
    }


    @Test
    public void testMangerMarguerite() {
        Mouton m = new Mouton(new Position(0, 0));
        m.manger(new Marguerite());
        assertEquals(4, m.getVitesse());
    }


    @Test
```

```
    public void testMangerCactus() {

        Mouton m = new Mouton(new Position(0, 0));

        m.manger(new Cactus());

        assertEquals(1, m.getVitesse());

    }


}
```

Loup:

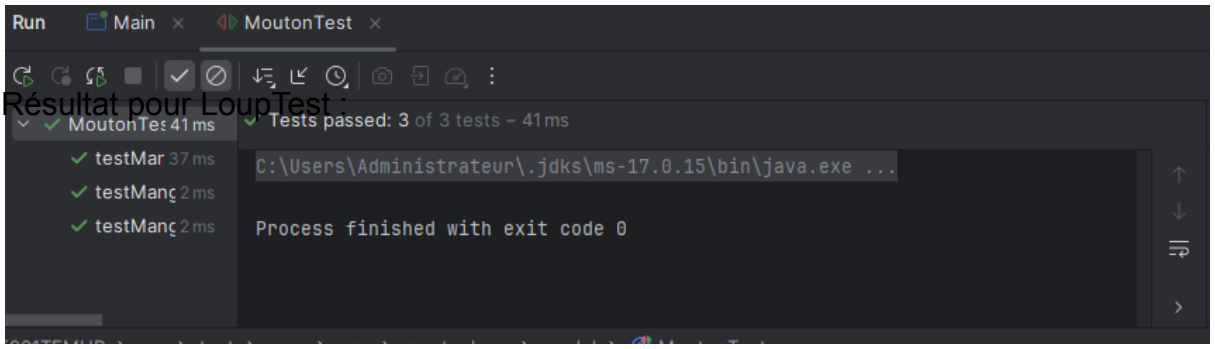
```java
import com.sae.moutonloup.model.*;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;


public class LoupTest {


    @Test
    public void testVitesseLoup() {
        Loup l = new Loup(new Position(2, 2));
        assertEquals(3, l.getVitesse());
    }


    @Test
    public void testPositionInitiale() {
        Loup l = new Loup(new Position(2, 2));
        assertEquals(2, l.getPosition().getX());
        assertEquals(2, l.getPosition().getY());
    }


    @Test
    public void testChangementDePosition() {
        Loup l = new Loup(new Position(0, 0));
        Position nouvelle = new Position(4, 4);
        l.setPosition(nouvelle);
```

```
        assertEquals(nouvelle, l.getPosition());

    }



}
```

b)Résultat du test de l'application

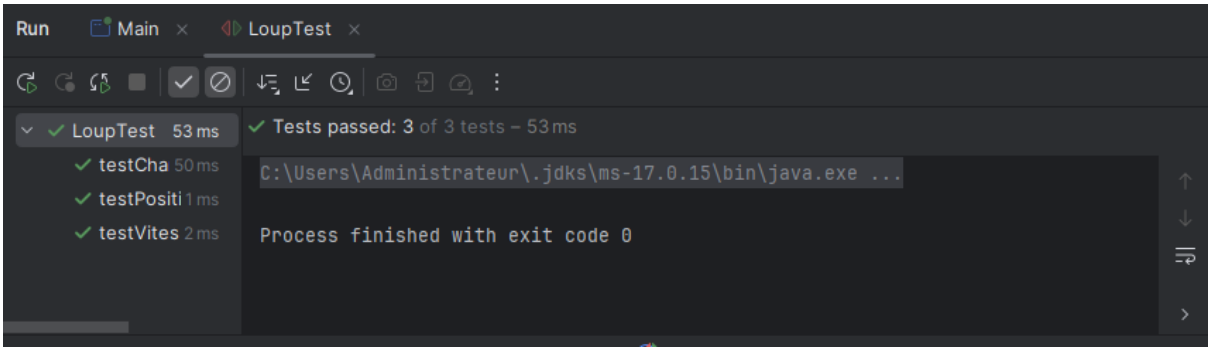
Voici les résultats des exécutions de tests dans IntelliJ à l'aide de JUnit 5.
Les tests ont été réalisés dans deux fichiers : MoutonTest.java et LoupTest.java

Résultat pour MoutonTest:



Résultat pour LoupTest :

Résultat pour LoupTest



Conclusion:

Les tests unitaires ont permis de valider deux aspects essentiels du comportement du simulateur :

- Le mouton adapte bien sa vitesse selon le type de végétal mangé (herbe, marguerite, cactus)

- Le loup conserve une vitesse constante et peut changer de position.

Les tests ont couvert l'ensemble des cas envisagés avec une couverture d'instruction et de branche.

Tous les tests sont passés avec succès, prouvant que les méthodes testées sont fonctionnelles.