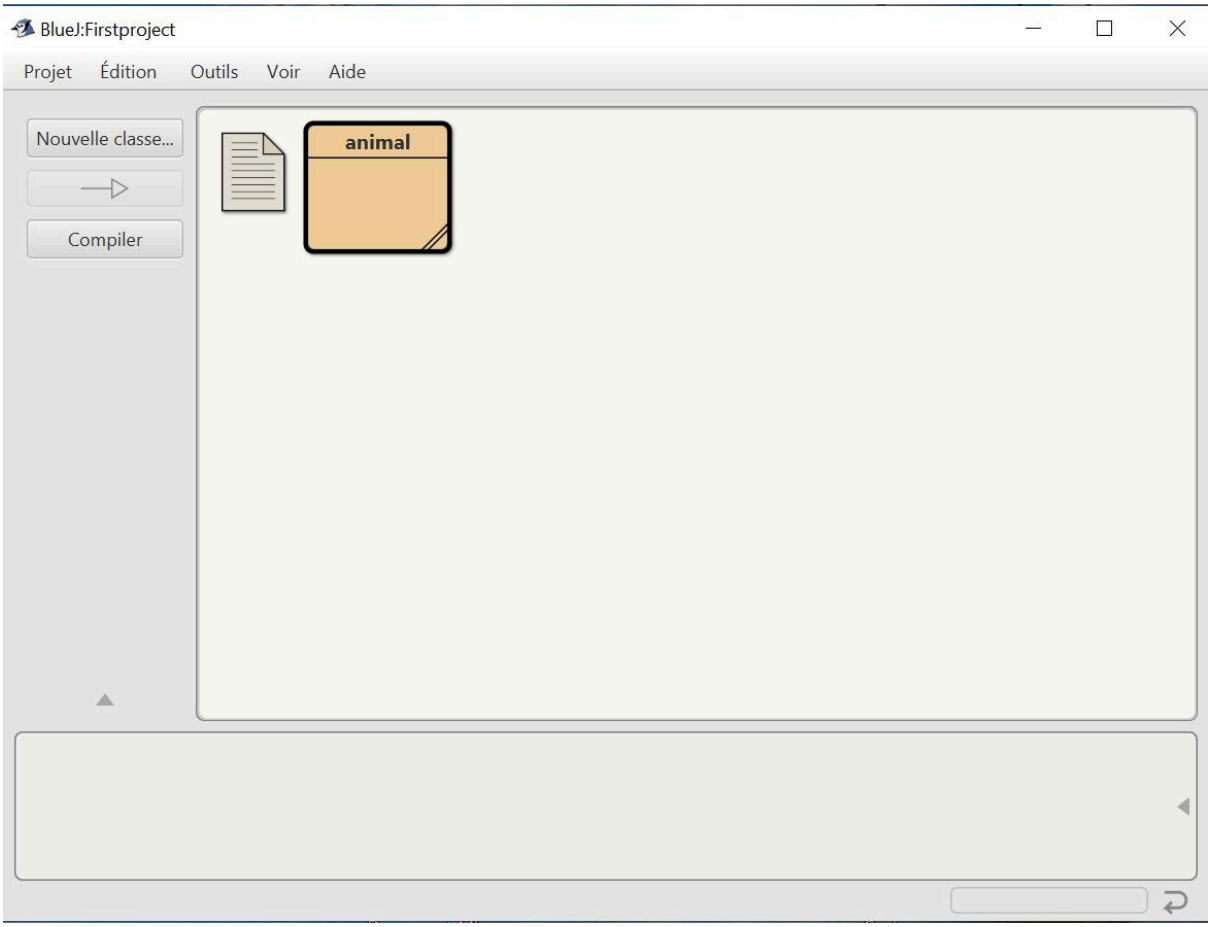
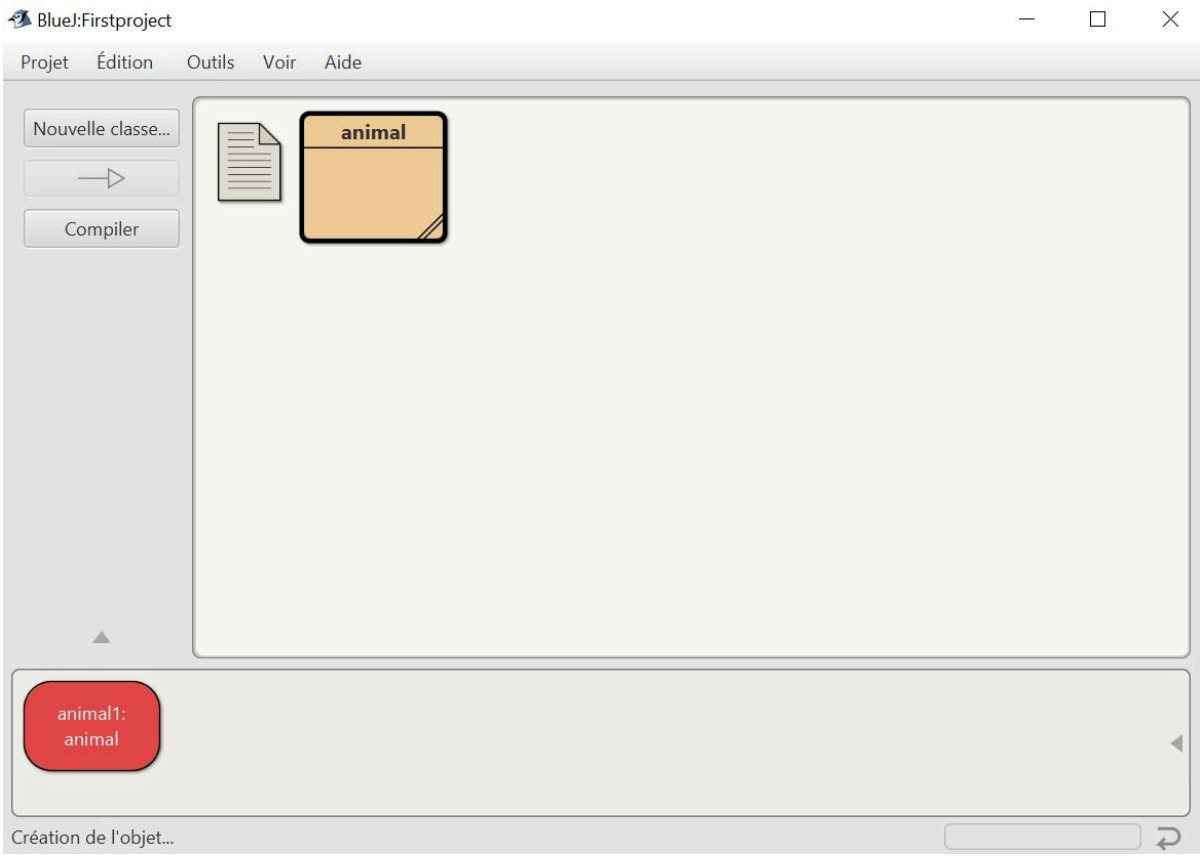
Commentaires sur les captures d'écran et description du fonctionnement

Création d'un nouveau projet et d'une classe "animal"

La capture montre la création et la compilation d'une nouvelle classe nommée "animal". Cette classe sera utilisée pour représenter des animaux avec des attributs et des méthodes spécifiques.



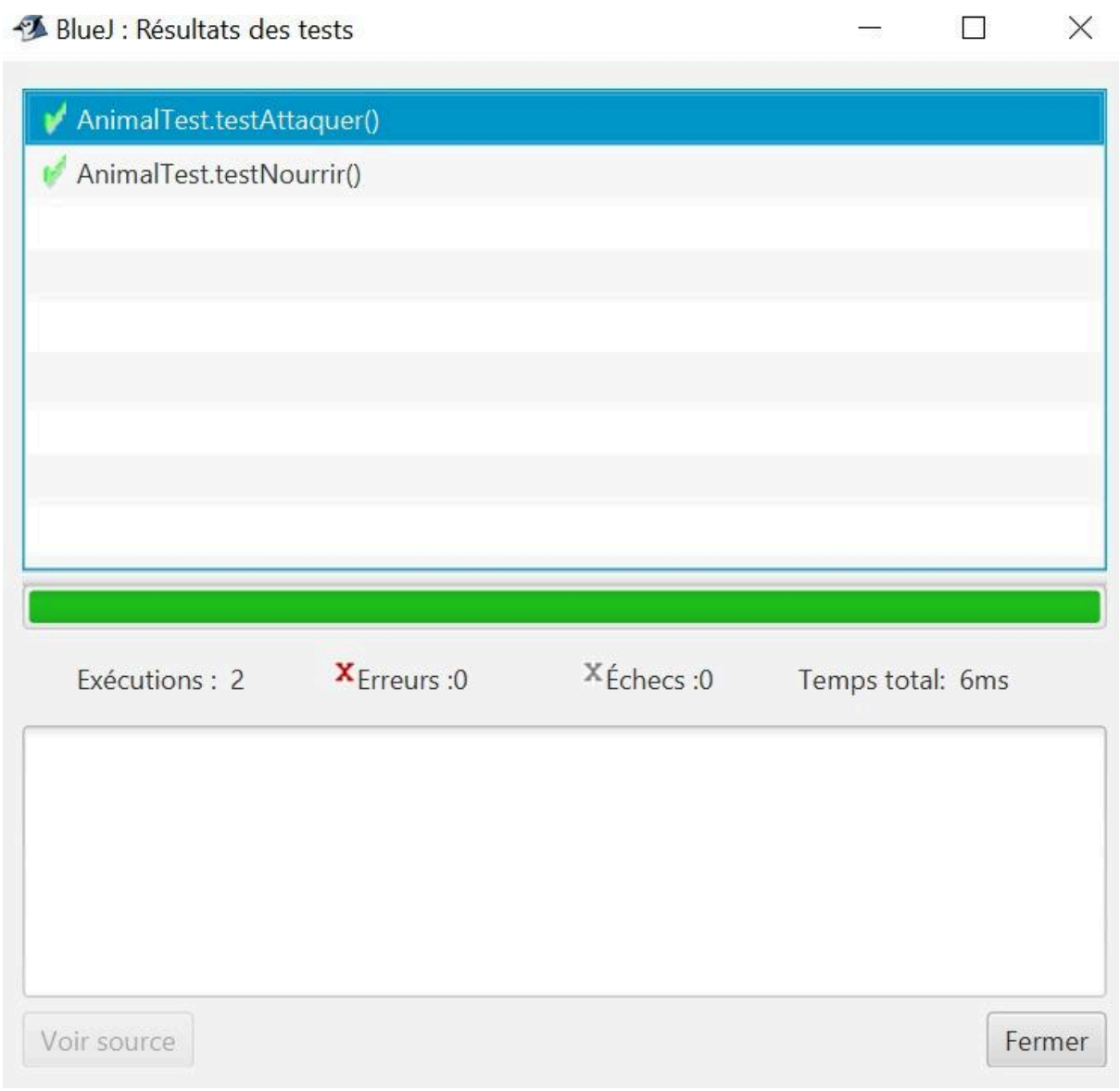


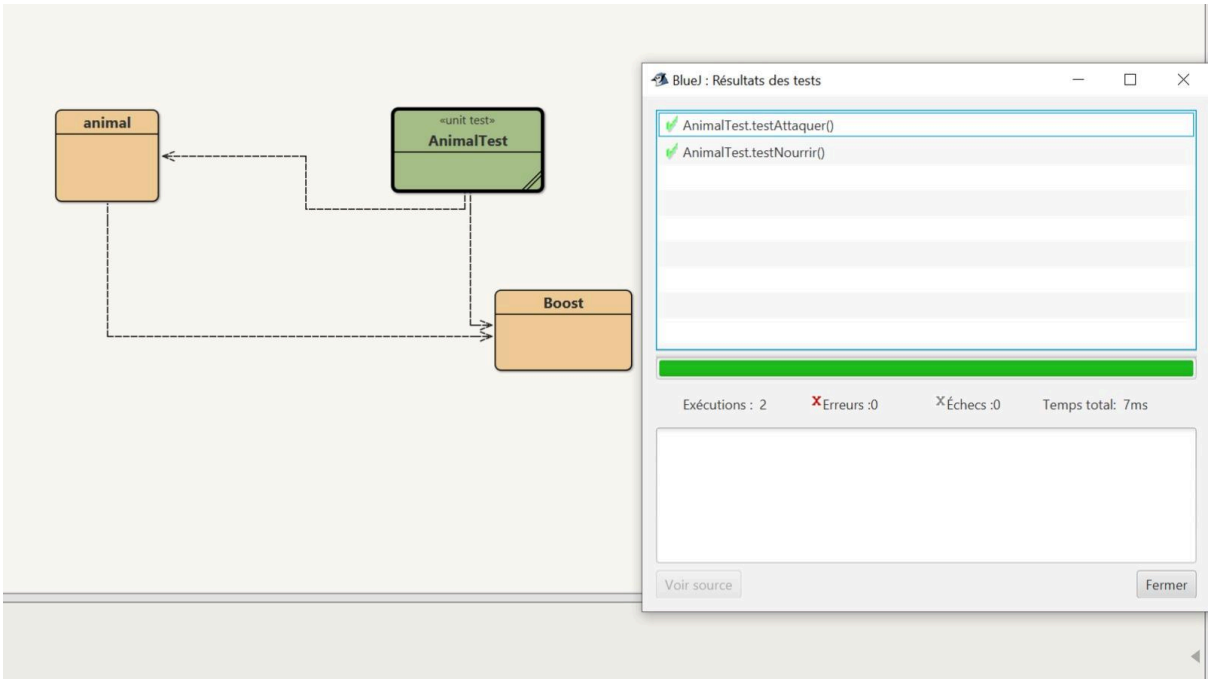
Cette capture montre l'instanciation de la classe animale.

Code de la classe "animal"

Ces captures montrent le code de la classe "animal". On voit que la classe a des attributs tels que "nom", "age" et "energie", ainsi que des méthodes pour les manipuler. La méthode "attaquer" permet à un animal d'attaquer un autre animal, ce qui réduit l'énergie de l'animal attaqué.

```java
public void afficherEnergie() {
  System.out.println("Energie =" + this.energie);
}
public void nourrir(Boost boost){
    energie += boost.Boost_energie();
}
public void attaquer(animal an){
    an.energie-=50;
}
public int getEnergie(){
    return this.energie;
}
```





Test unitaire de la classe "animal"

Cette capture montre les résultats des tests unitaires pour la classe "animal". Les tests sont réussis, ce qui indique que les méthodes de la classe fonctionnent comme attendu.

Instantiation de la classe "animal"

Ces captures montrent l'instantiation de la classe "animal". Un objet "animal1" est créé avec des paramètres spécifiques, tels que le nom et l'âge.

```java
// variables d'instance - remplacez l'exemple qui suit par le vôtre
private String nom;
private int age ;
private int energie = 100;

/**
 * Constructeur d'objets de classe animal
 */
public animal(String n,int a)
{
    nom = n;
    age = a;
}
```

```java
private int type;
private int energie;
```

```java
public Boost(int ty)
{
    type = ty;
}


public int Boost_energie()
{
    if (type == 1){
        return 60;
    }
    else if (type == 2){
        return 80;
    }
    else {
        return 120;
    }

}
```

Code de la classe "animal" avec la méthode "nourrir"

Ces captures montrent le code de la classe "animal" avec la méthode "nourrir". Cette méthode permet à un animal de se nourrir en utilisant un objet "Boost", qui est une autre classe.

```java
@BeforeEach
public void setUp() {
    animal1 = new animal("papicha", 3);
    boost1 = new Boost(2);
    animal2 = new animal("son", 2);
}

@AfterEach
public void tearDown() {
    animal1 = null;
    boost1 = null;
    animal2 = null;
}

@Test
public void testNourrir() {
    animal1.nourrir(boost1);
    assertEquals(180, animal1.getEnergie());
}

@Test
public void testAttaquer() {
    animal1.attaquer(animal2);
    assertEquals(50, animal2.getEnergie());
}
```

Relation entre les classes "animal" et "Boost"

Le diagramme de classes montre la relation entre les classes "animal" et "Boost". La classe "animal" a une association avec la classe "Boost", ce qui signifie qu'un animal peut utiliser un boost pour se nourrir.

Fonctionnement de la classe "Boost"

La classe "Boost" a des attributs tels que "type" et "energie", et une méthode pour calculer l'énergie du boost en fonction de son type. La classe "animal" utilise cette classe pour se nourrir.

En résumé, les deux classes "animal" et "Boost" sont liées par une association. La classe "animal" représente des animaux avec des attributs et des méthodes spécifiques, tandis que la classe "Boost" représente des boosts qui peuvent être utilisés par les animaux pour se nourrir. La méthode "nourrir" de la classe "animal" utilise un objet "Boost" pour augmenter l'énergie de l'animal.