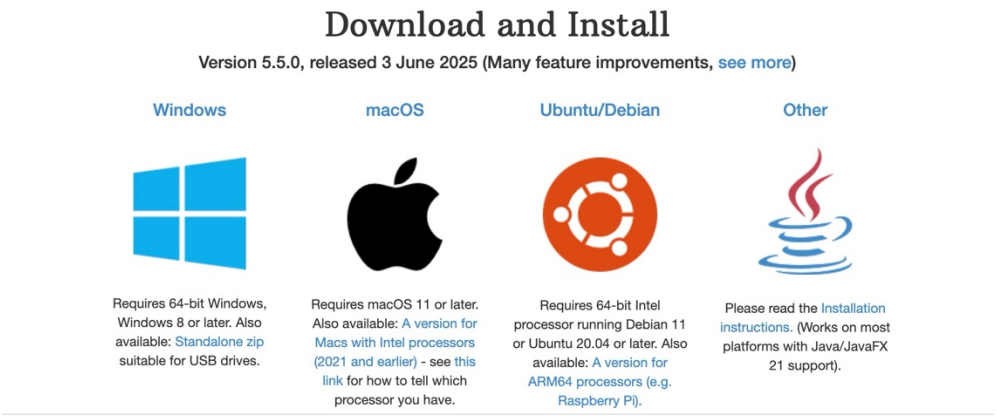
TUTORIEL : LES TESTS UNITAIRES ET LE REFACTORING

CHEN Yuxuan
LIN Hongxiang
TP1 Agilité

Dauphine | PSL
UNIVERSITÉ PARIS

1. Téléchargement et installation du logiciel

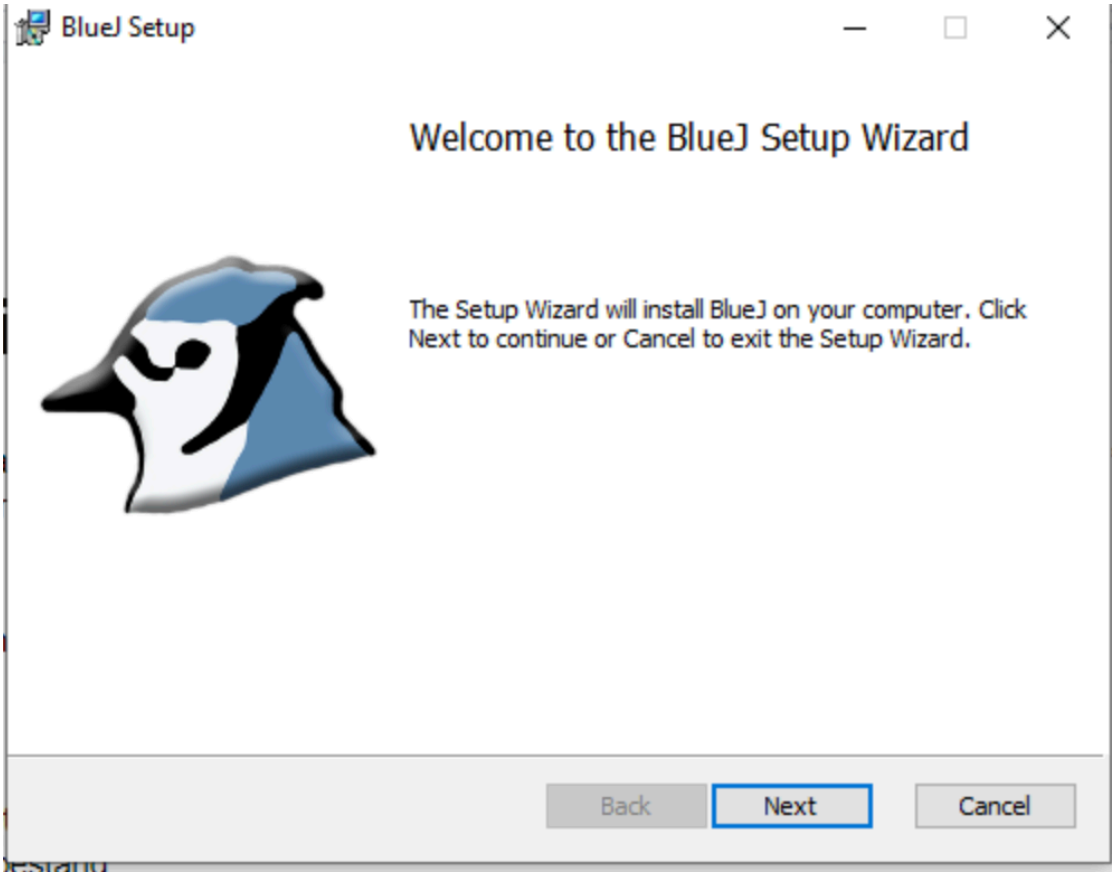


Téléchargez BlueJ via le lien suivant : http://www.bluej.org/
Pensez à bien sélectionner la version adaptée à votre système d'exploitation (Window,mac OS,Linux…)

CHEN Yuxuan
LIN Hongxiang
TP1 Agilité

Dauphine | PSL
UNIVERSITÉ PARIS

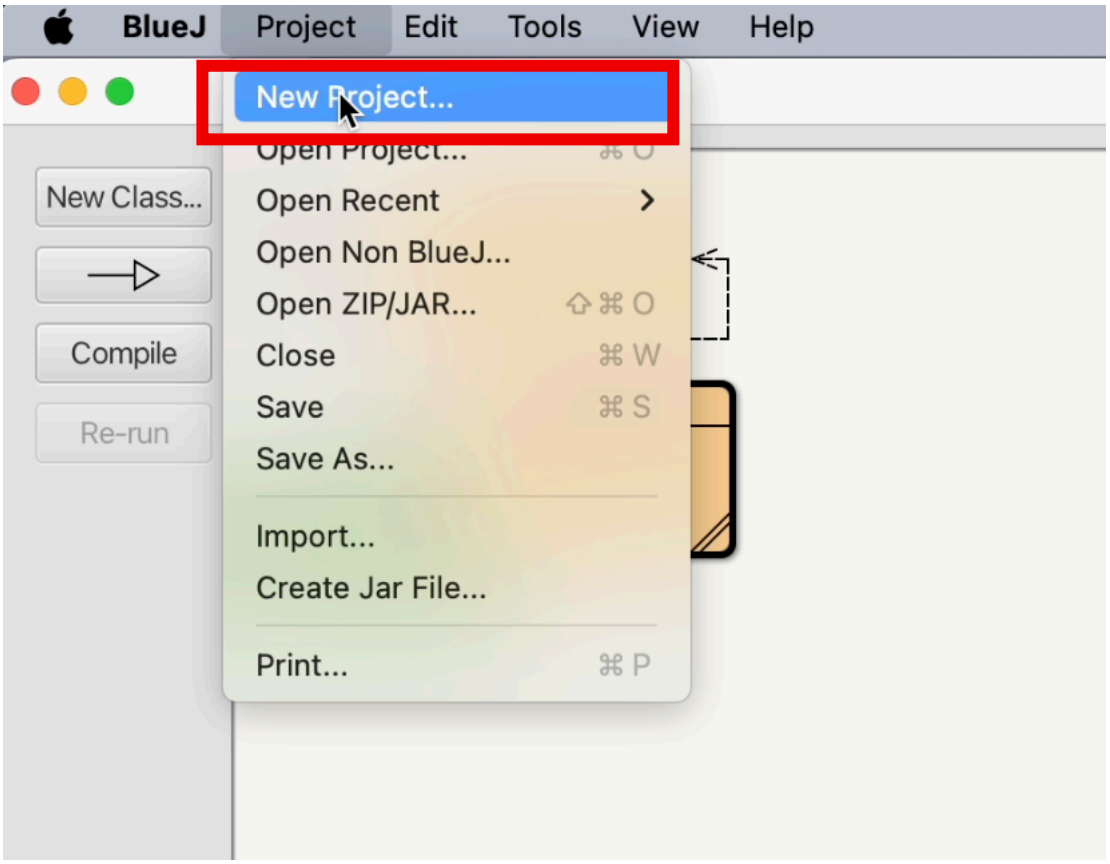
2. Installation du logiciel



Exécutez le fichier téléchargé et suivez les étapes d'installation par défaut

3. Création d'un nouveau projet

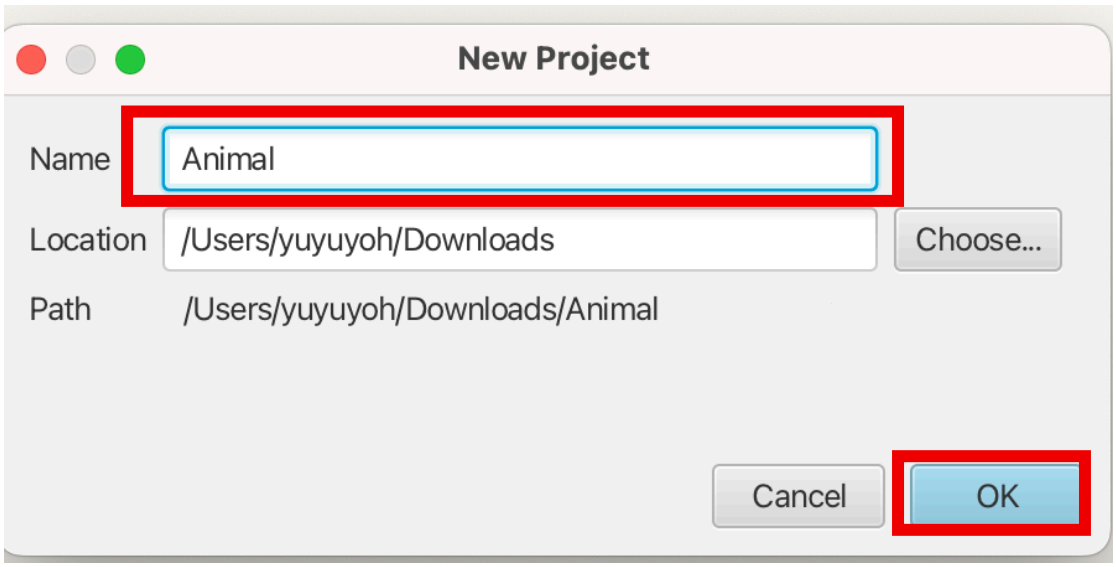
Lancez BlueJ et accédez au menu principal.



Sélectionnez **Project**, puis optez pour **New Project**.

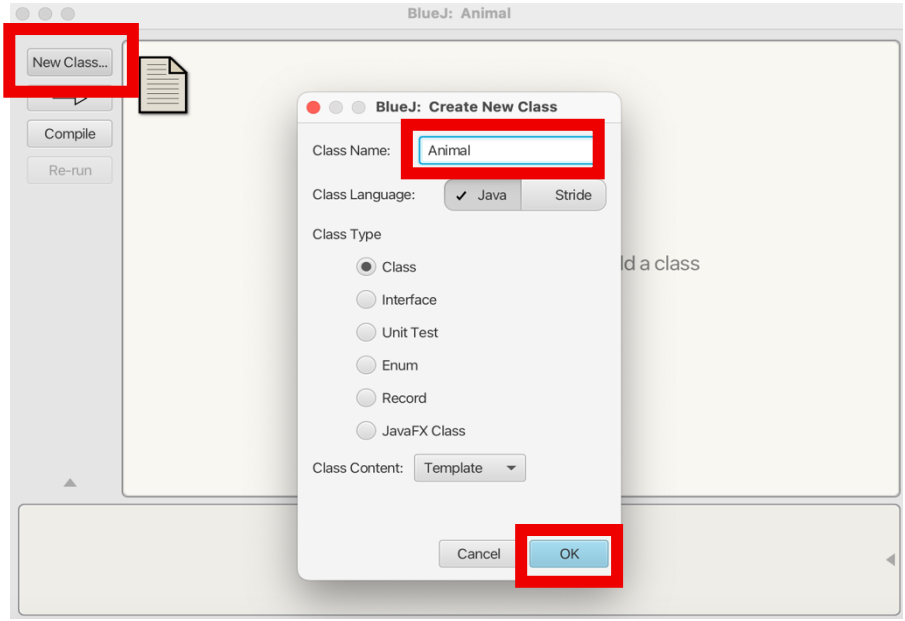
Dans le cadre de cette démonstration, nous allons modéliser un système écologique centré sur la faune. Libre à vous d'imaginer un autre domaine d'application si vous le souhaitez : gestion d'une bibliothèque, suivi sportif, ou tout autre univers qui vous inspire.

Un conseil technique important : privilégiez les caractères ASCII standards (lettres non accentuées, chiffres) pour nommer vos projets, classes et variables. Évitez les accents, cédilles ou signes diacritiques qui pourraient entraîner des erreurs d'encodage inattendues.



Saisissez ensuite un nom pour votre projet — par exemple **"Animal"** — et validez en cliquant sur **OK**.

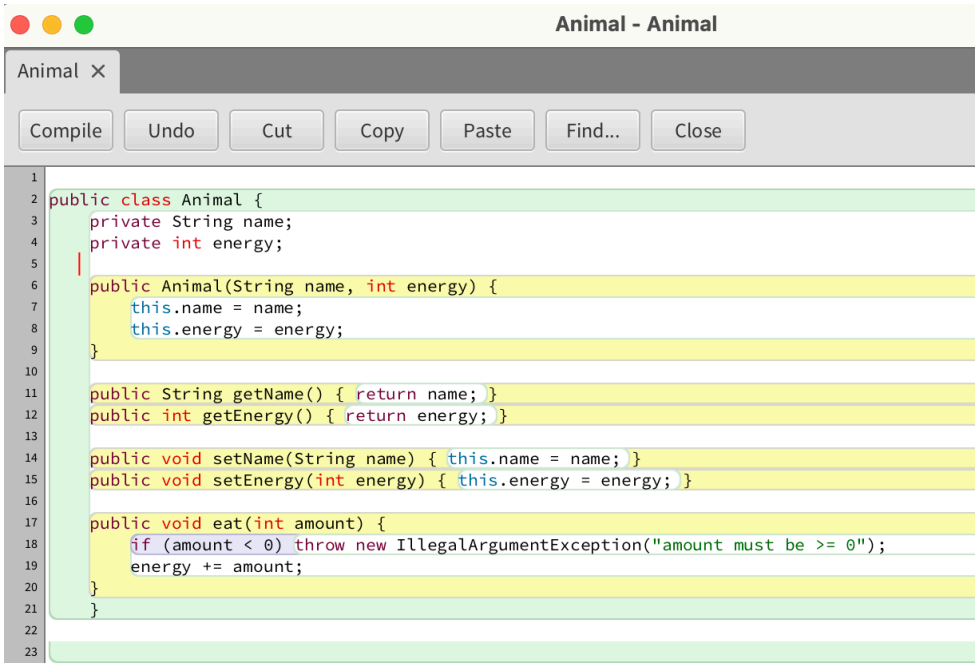
4. Création d'une classe



Créer un modèle d'animal. En programmation, on appelle ça une **classe**.

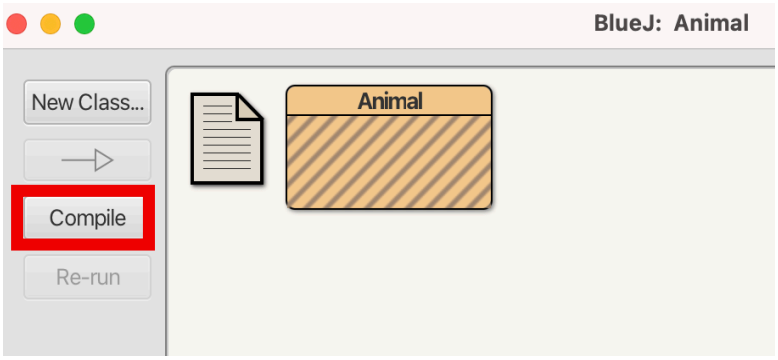
Cliquez sur le bouton « new class », saisissez ensuite un nom pour votre classe— par exemple **"Animal"** — et validez en cliquant sur **OK**.

Maintenant, écrivons du code :



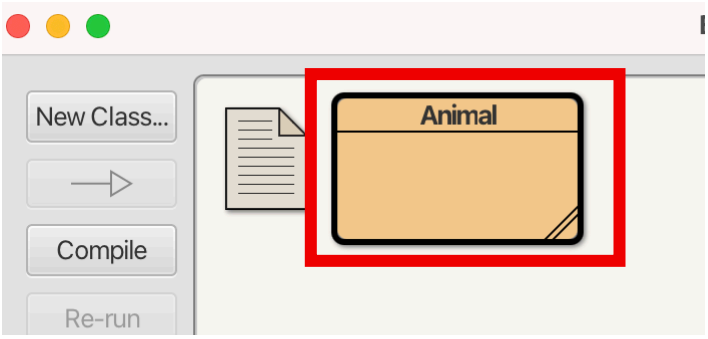
- nom et energy sont des **caractéristiques** de l'animal (comme son nom et son niveau d'énergie)
- getNom() et getEnergy() sont des **questions** qu'on peut poser à l'animal ("Comment tu t'appelles ?", "Tu as combien d'énergie ?")

5. Compilation d'une classe



Cliquez droit sur la classe Animal Sélectionnez « Compile » Vérifiez que le fond de la classe devient à damiers (signe de compilation réussie)

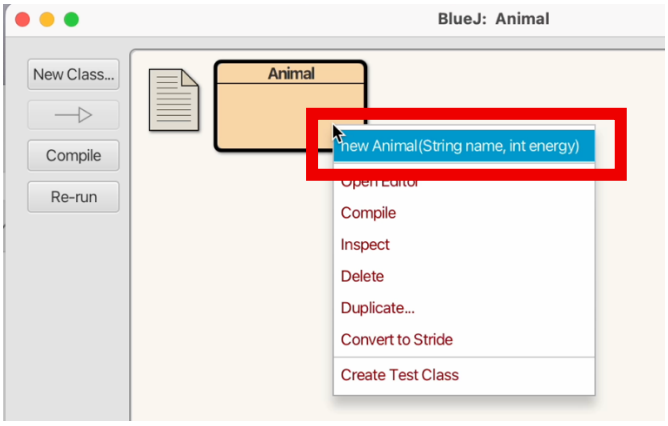
CHEN Yuxuan
LIN Hongxiang
TP1 Agilité

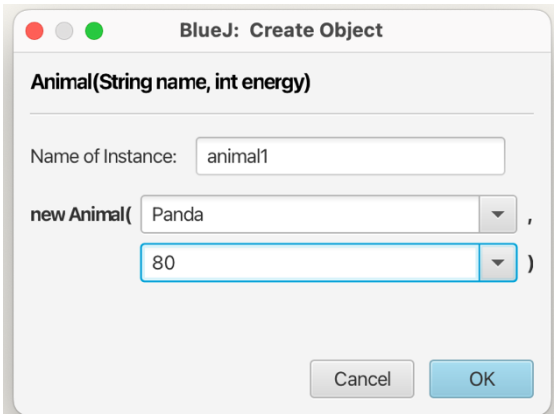


6. Instanciation

Créer un animal concret à partir de votre modèle.

Cliquez droit sur la classe Animal, Sélectionnez new Animal (String name, int Energy)

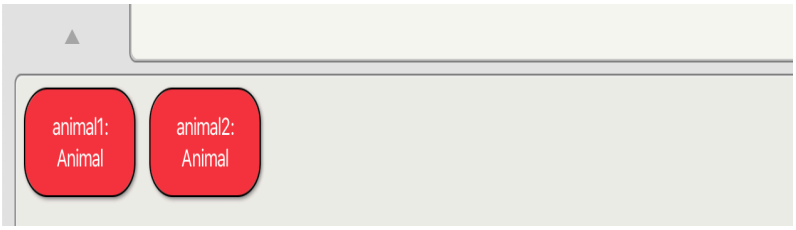




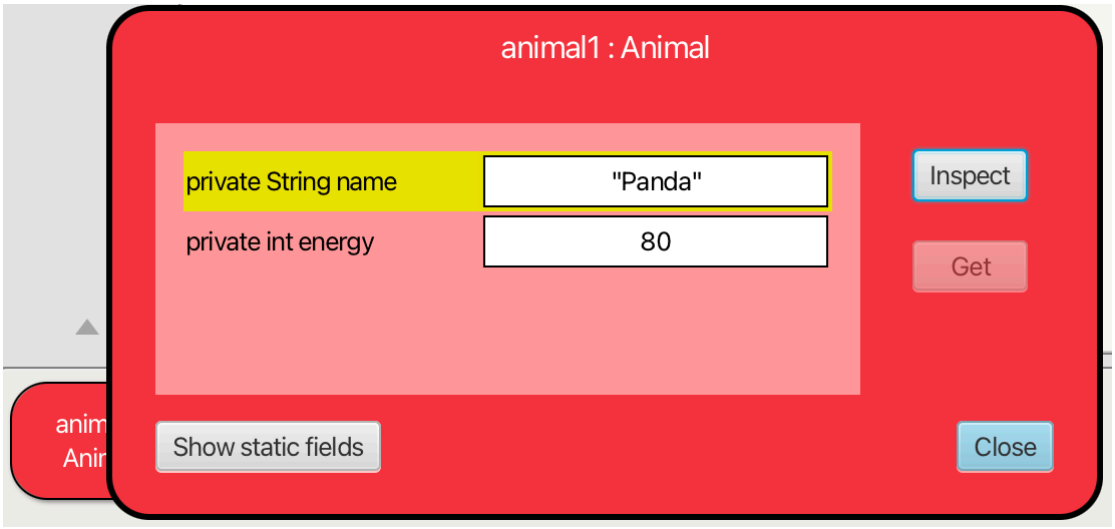
Dans la fenêtre qui s'ouvre :

- Name of instance : laissez animal1
- Paramètres : « Panda » et 80 (!! **IMPORTANT** : "Panda" doit avoir **des guillemets)**

Cliquez sur OK et observez l'objet panda apparaître dans l'Object Bench (en bas)

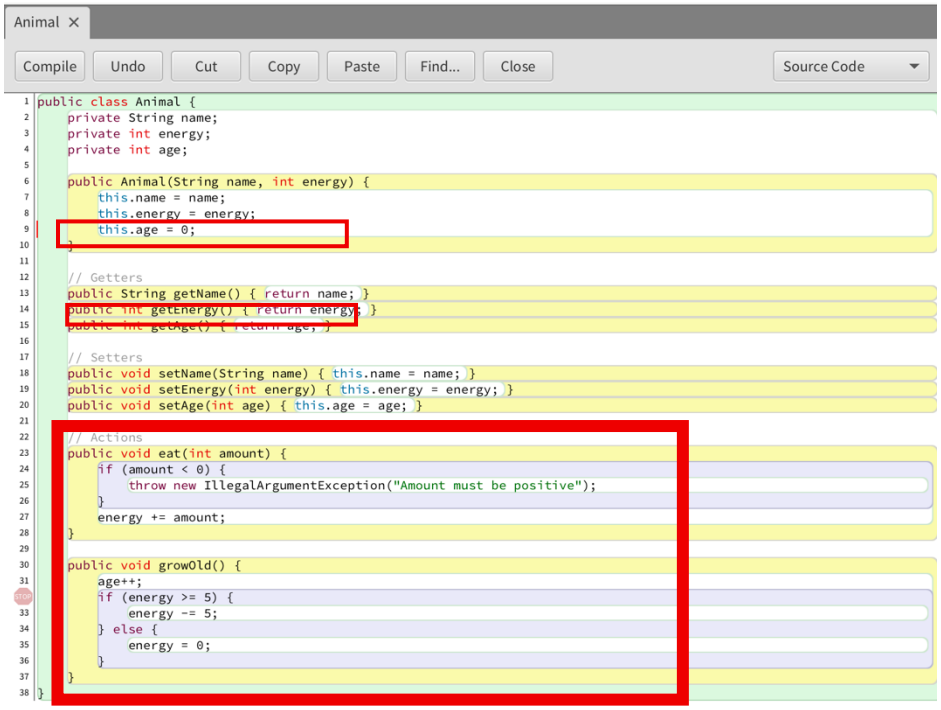


CHEN Yuxuan
LIN Hongxiang
TP1 Agilité



7. Ajout des attributs dans la classe

Nous allons maintenant ajouter l'âge et des actions (manger, vieillir). Double-cliquez à nouveau sur « Animal » et Fermez et compilez en cliquant sur « compile »



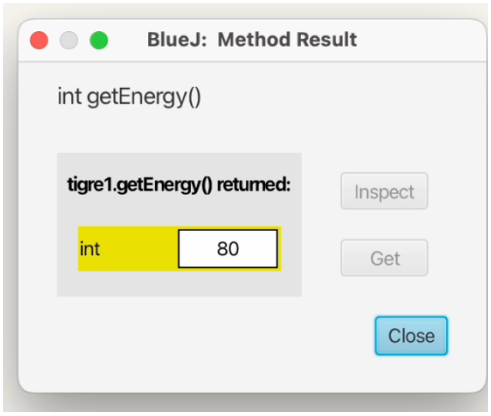
- age: nouvelle caractéristique
- eat() : action qui augmente l'énergie
- growOld() : action qui augmente l'âge et baisse l'énergie

CHEN Yuxuan
LIN Hongxiang

8. Test interactif

Nous allons jouer avec votre animal pour voir comment il réagit.

1. Supprimez l'ancien panda (clic droit sur animal1 → Remove)
2. Créez un nouveau : new Animal ("Tigre", 80) → tigre1



Test 1 - Vérifier l'état initial :

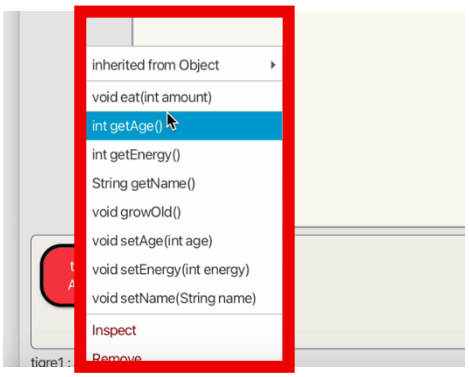
- Clic droit sur tigre1
- getNom() → "Tigre"
- getEnergy() → 80
- getAge() → 0

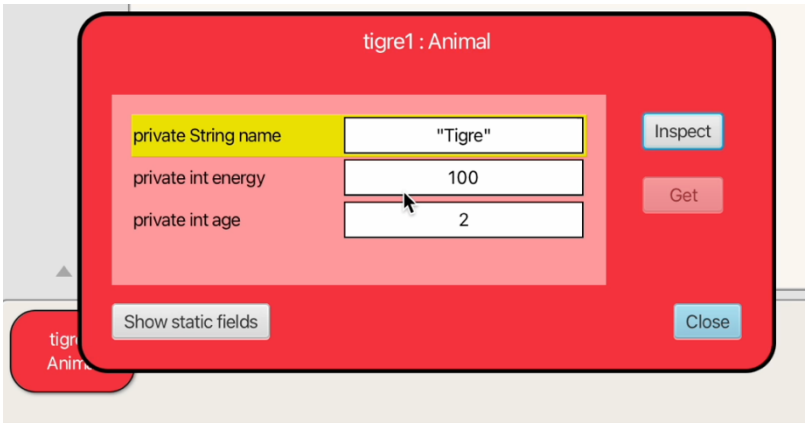
Test 2 - Le faire manger:

- Clic droit sur tigre1
- eat(int quantite)
- Écrivez 30 (sans guillemets, c'est un nombre)

Test 3 - Le faire vieillir :

- growOld()
- Résultat : "Tigre a maintenant 2 an(s)"
- getEnergy() → 100 (110 - 10)



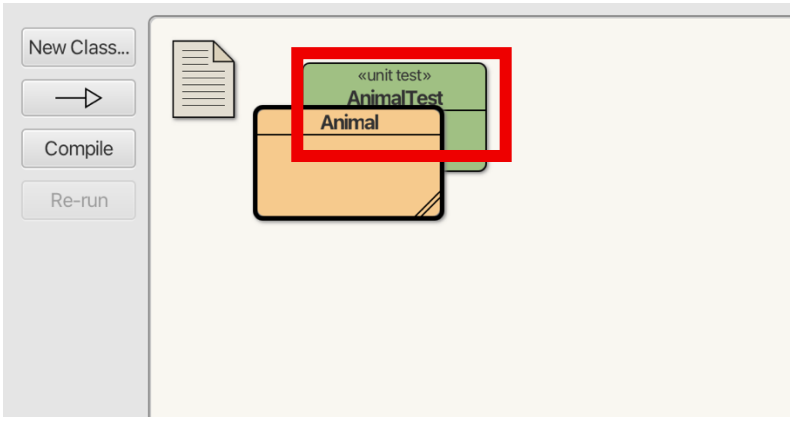


9. Test Unit

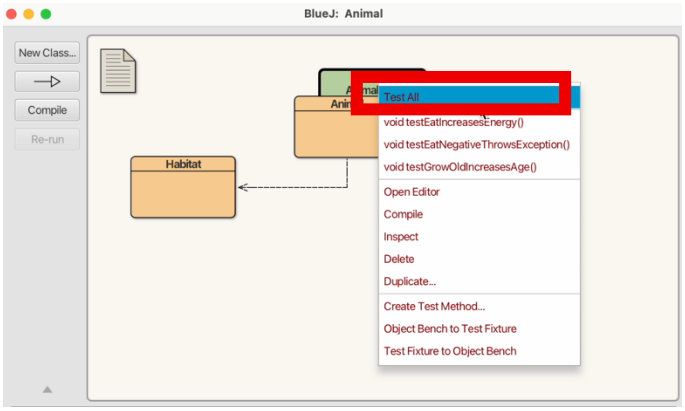
Nous allons créer des tests unitaires qui testent automatiquement votre code.

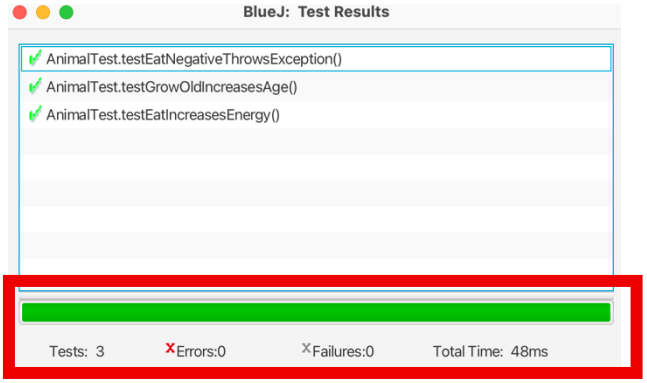
- Cliquez droit sur Animal et cliquez sur « create Test Class »
- Nommez-la AnimalTest

CHEN Yuxuan
LIN Hongxiang
TP1 Agilité



- Compilez AnimalTest
- Cliquez droit sur AnimalTest → Test All

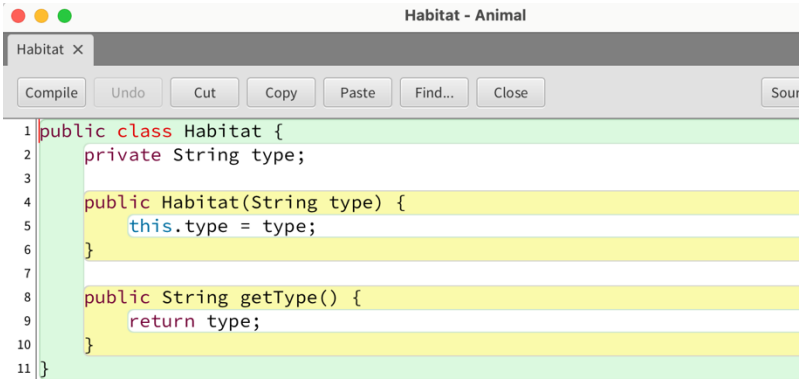




Une barre verte apparaît ! Cela signifie que tous vos tests passent. Bravo !

10. Création d'une seconde classe Habitat et association avec la classe Animal

Nous allons dans cette section créer un endroit où l'animal peut vivre et les relier.
Tout à bord, nous allons créer la classe Habitat



CHEN Yuxuan
LIN Hongxiang
TP1 Agilité

Ensuite , nous allons modifier la classe Animal pour qu'il connaisse son habitat :

- Ouvrez la classe Animal
- Ajoutez le nouveau attribut habitat (après les autres attributs) et les nouvelles méthodes liées à la classe Habitat à la fin.

Animal ×

Compile | Undo | Cut | Copy | Paste | Find... | Close | Source Code

```
1  public class Animal {
2      private String name;
3      private int energy;
4      private int age;
5      private Habitat habitat;
6
7      public Animal(String name, int energy) {
8          this.name = name;
9          this.energy = energy;
10         this.age = 0;
11         this.habitat = null;
12     }
13
```

```
41     // Habitat methods
42     public Habitat getHabitat() {
43         return habitat;
44     }
45
46     public void moveTo(Habitat newHabitat) {
47         if (energy >= 10) {
48             this.habitat = newHabitat;
49             energy -= 10;
50         }
51     }
52
53     public void leaveHabitat() {
54         this.habitat = null;
55     }
56
57     public boolean hasHabitat() {
58         return habitat != null;
59     }
60
61     public String describe() {
62         if (hasHabitat()) {
63             return name + " (" + energy + " energy) lives in " + habitat.getType();
64         } else {
65             return name + " (" + energy + " energy) is homeless";
66         }
67     }
68
```

CHEN Yuxuan
LIN Hongxiang

11. Collaboration entre les 2 classes

Maintenant que vous avez réussi à connecter un Animal à son Habitat dans votre programme, une nouvelle dimension s'ouvre à vous ! Cette connexion signifie que vos objets peuvent désormais "se parler" et partager des informations. C'est un peu comme si l'animal pouvait regarder autour de lui et comprendre dans quel type d'environnement il se trouve.

Imaginez cette situation pratique : grâce à ce lien, votre programme peut maintenant calculer combien de nourriture un animal a besoin chaque jour, en tenant compte de l'endroit où il vit. Pour y arriver, vous devrez prévoir trois cas différents :

- **Sans habitat** : L'animal doit survivre avec des ressources de base
- **Habitat ordinaire** : Une forêt ou une prairie offre suffisamment de ressources
- **Habitat difficile** : Un désert ou une montagne demande plus d'efforts pour survivre

Pour ajouter cette intelligence à votre Animal, vous allez créer une nouvelle "instruction" (ce qu'on appelle une méthode en programmation). Voici comment procéder :

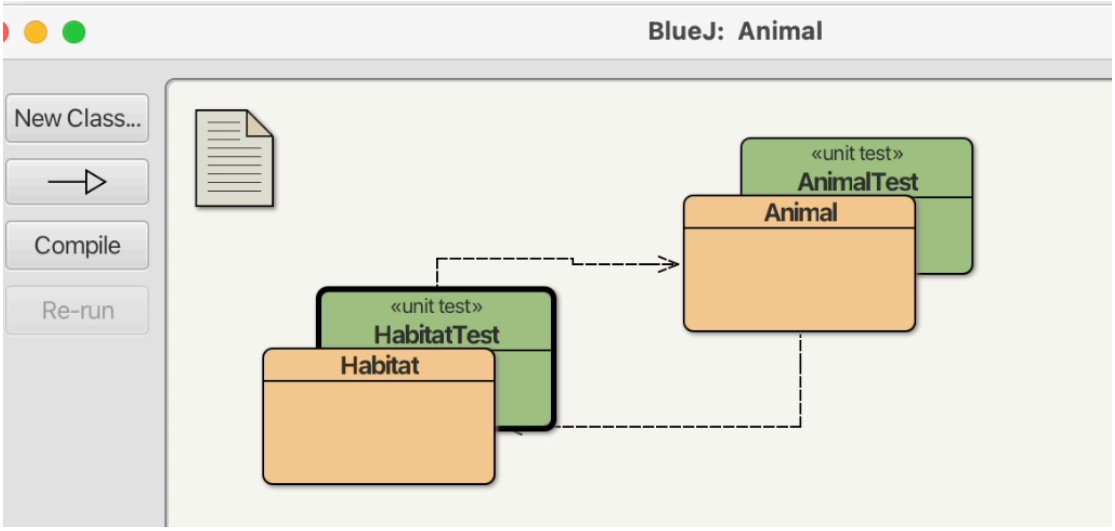
Dans la fenêtre de code de la classe Animal, ajoutez cette nouvelle fonction :

```
63              return name + " (" + energy + " energy) is homeless";
64          }
65      }
66      public int calculateDailyNeeds() {
67          if (habitat == null) {
68              return 20; // Basic needs
69          }
70
71          String type = habitat.getType();
72          if (type.equals("Desert")) {
73              return 40;
74          } else if (type.equals("Forest")) {
75              return 25;
76          } else if (type.equals("Savanna")) {
77              return 30;
78          } else {
79              return 20;
80          }
81      }
82
83      public boolean canSurvive() {
84          return energy >= calculateDailyNeeds();
85      }
```

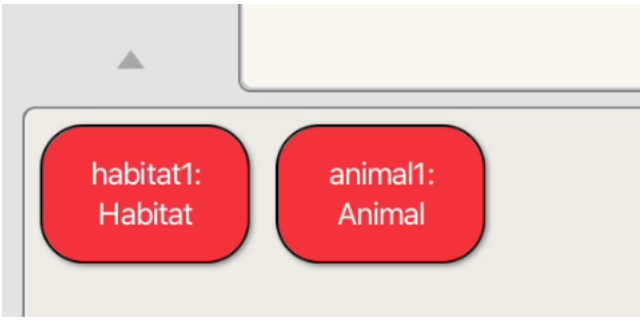
CHEN Yuxuan
LIN Hongxiang
TP1 Agilité

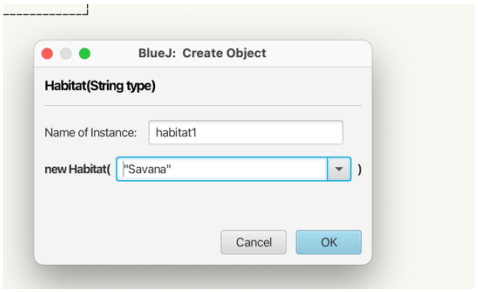
12. Fixturation

Maintenant que notre code permet d'associer un Animal à son Habitat, nous allons préparer un environnement de test reproductible. Plutôt que de créer de nouveaux objets pour chaque test, nous allons configurer une fois pour toutes un écosystème de base qui servira de point de départ à tous nos tests. C'est ce qu'on appelle une **fixture**.



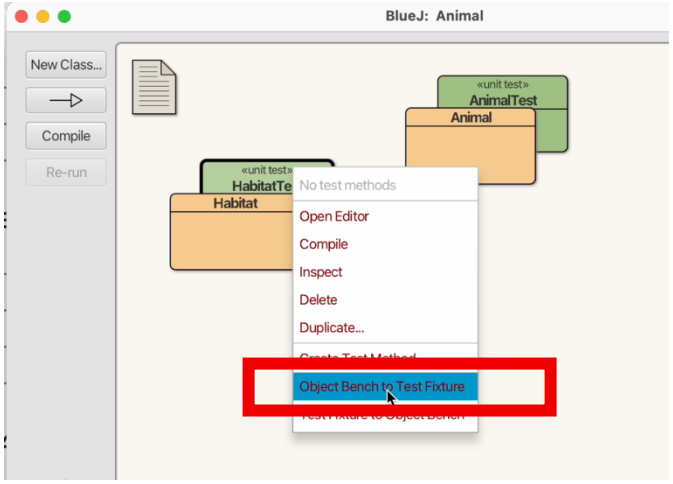
Dans BlueJ, cette préparation se fait de manière très visuelle. Commencez par créer dans l'Object Bench un Habitat "Savanna" et un Animal "Lion" avec 100 points d'énergie, puis associez-les à l'aide de la méthode moveToHabitat().



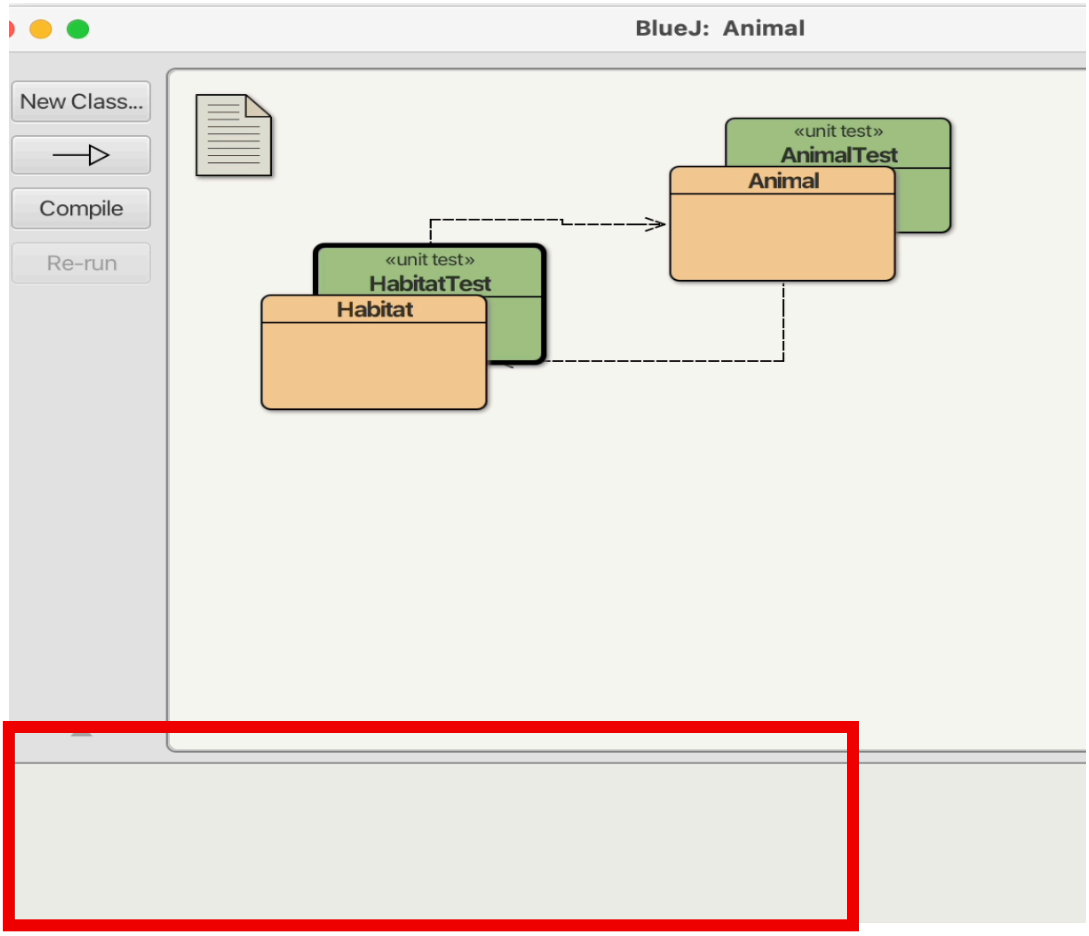


CHEN Yuxuan
LIN Hongxiang
TP1 Agilité

Une fois ces objets créés, faites un clic droit sur votre classe de test AnimalTest (ou HabitatTest) et sélectionnez **"Object Bench → Fixtures"**.



BlueJ va alors capturer l'état actuel de vos objets et les intégrer automatiquement dans un environnement de test. Vous remarquerez que les objets disparaissent de l'Object Bench : ils sont maintenant encapsulés dans la fixture et seront recréés automatiquement avant chaque exécution de test



Cette approche garantit que tous vos tests partent du même point de départ, rendant les résultats plus fiables et les tests plus faciles à maintenir.

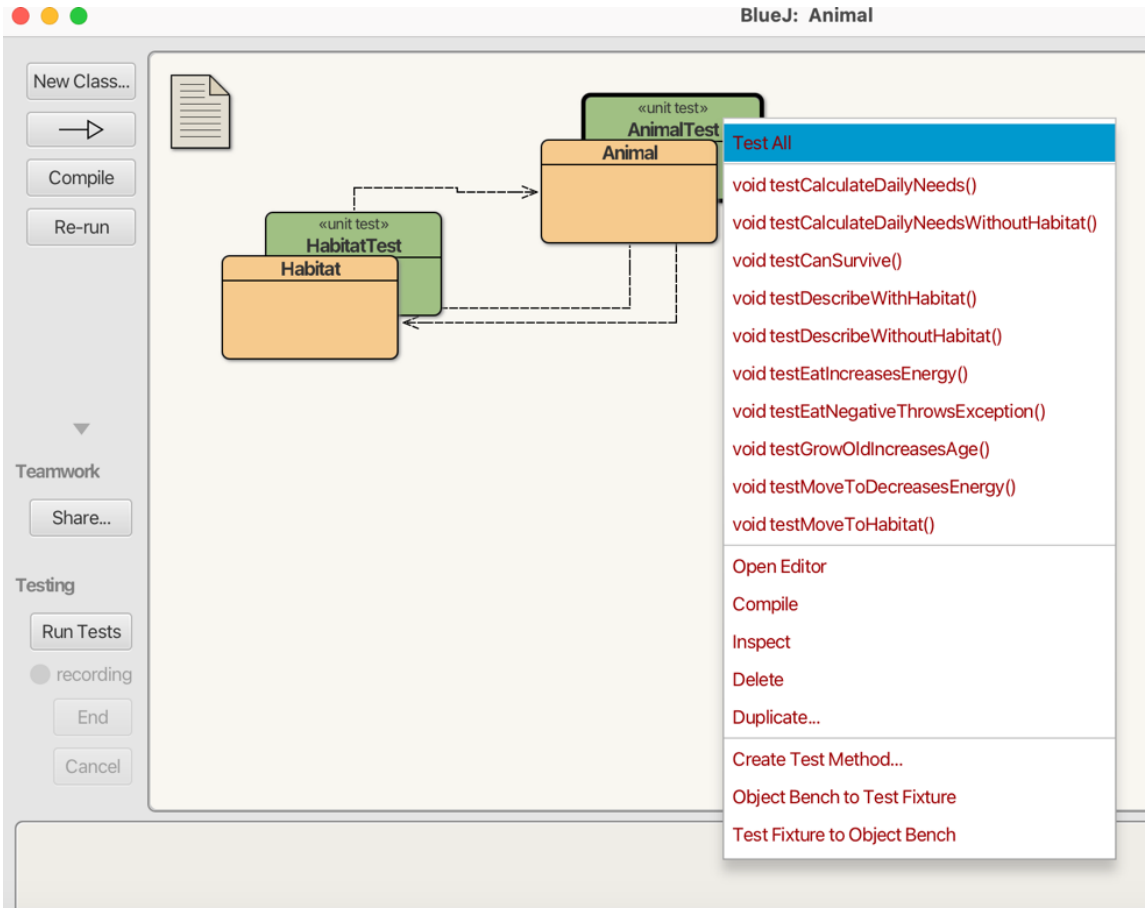
CHEN Yuxuan
LIN Hongxiang
TP1 Agilité

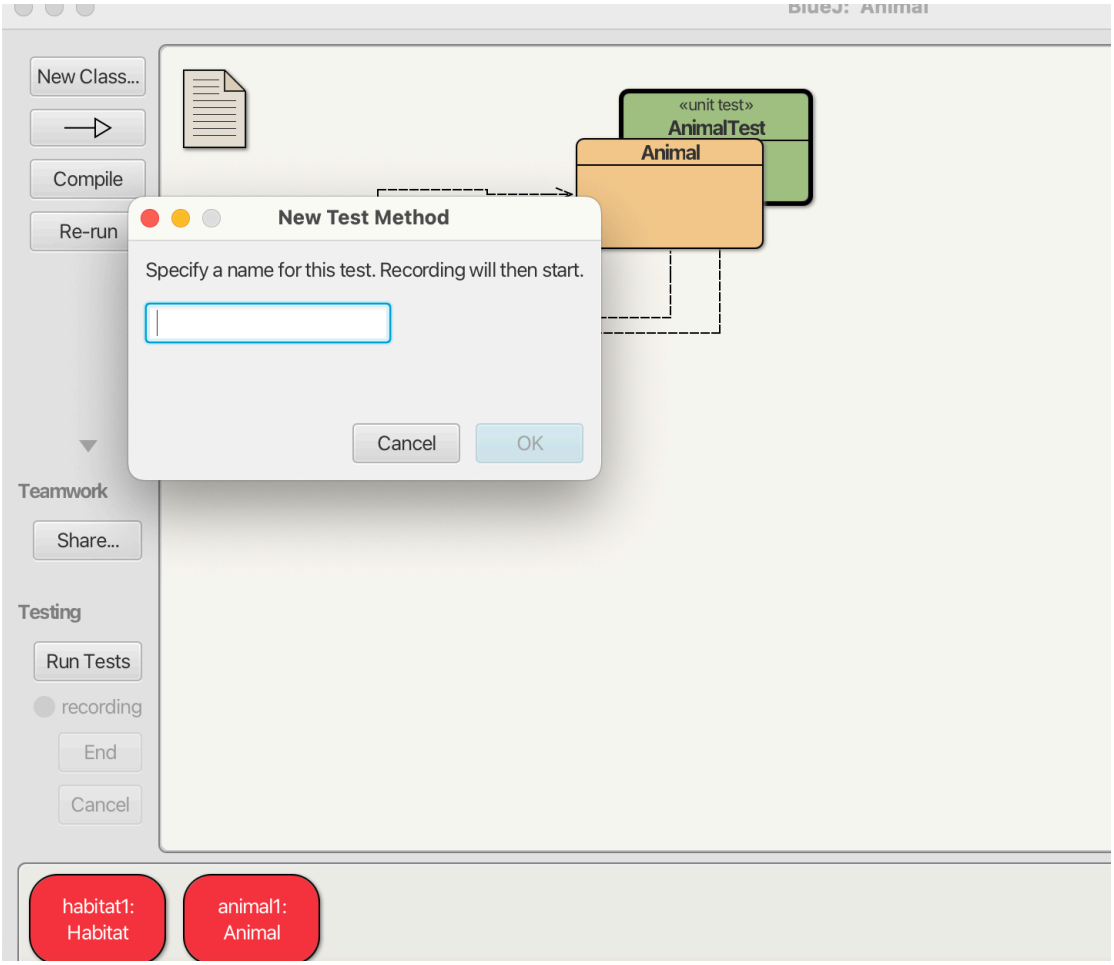
Dauphine | PSL
UNIVERSITÉ PARIS

Grâce à cette fixture, vous pouvez maintenant écrire des tests qui vérifient par exemple si le lion a bien son habitat attribué, ou calculer ses besoins énergétiques en fonction de la savane où il réside, sans avoir à répéter la configuration initiale à chaque fois.

13. Test complet

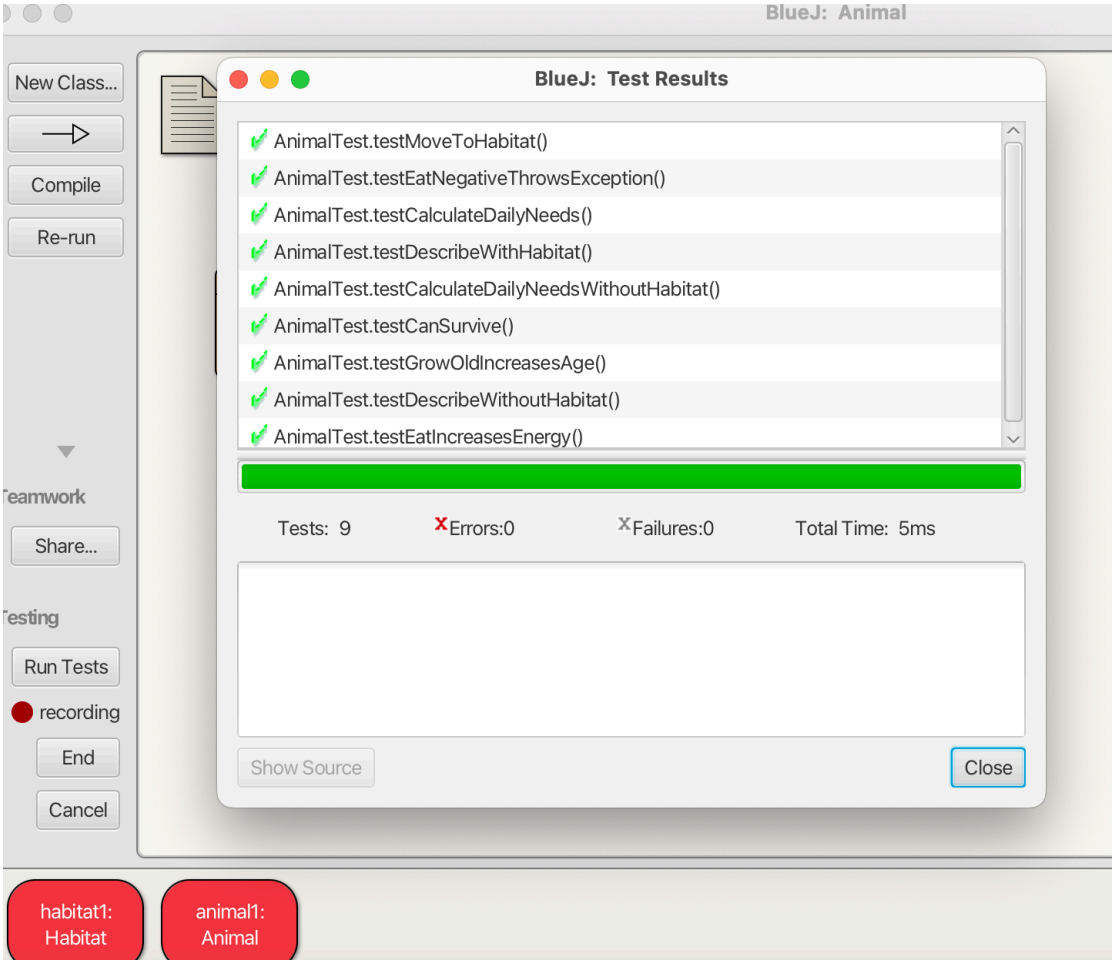
Maintenant, on va créer un test dans l'environnement de test que nous avons préparé. Pour cela, comme dans la première partie, on va faire un clic droit sur la classe AnimalTest et sélectionner **"Create Test Method"** (Enregistrer une méthode de test).
Une fois ce bouton cliqué, comme nous sommes dans l'environnement de test, tu remarqueras que les objets que nous avions instanciés et qui avaient précédemment disparu sont de nouveau présents dans la barre d'objets en bas (habitat1 et animal1).





CHEN Yuxuan
LIN Hongxiang
TP1 Agilité

Dauphine | PSL
UNIVERSITÉ PARIS



Attention à bien respecter la casse, les espaces et le contenu exact de la chaîne. Ensuite, on peut terminer l'enregistrement du test en cliquant sur **"End"**, puis exécuter les tests pour vérifier que tout fonctionne bien.

Si cela a bien fonctionné, tu devrais avoir tous les tests au vert (✓) dans la fenêtre des résultats !

CHEN Yuxuan
LIN Hongxiang
TP1 Agilité