



Tutoriel BlueJ : découvrir les pratiques agiles de tests unitaires et refactoring avec Pokémons

Baptiste Matrat
Marie Probert

Bienvenue dans le monde des Pokémons !

"Bonjour ! Je suis ravi de faire ta connaissance. Bienvenue dans le monde merveilleux des Pokémons ! Mon nom est le Professeur Java, mais tout le monde ici m'appelle simplement le "Professeur BlueJ".

Ce monde est peuplé de créatures appelées Pokémons. Pour certains, les Pokémons sont des objets d'étude ; d'autres les utilisent pour combattre. De mon côté, ma profession est d'étudier la structure de leur code et de veiller à ce qu'ils soient bien... "compilés".

Mais dis-moi, es-tu prêt à devenir un Maître Développeur ? Ta quête ne consiste pas seulement à les capturer, mais à apprendre à les instancier, à tester leurs attributs et à refactorer ton code pour qu'il soit aussi robuste qu'un Mackogneur !

Ton voyage commence ici, dans le laboratoire de BlueJ. Tu vas apprendre à construire tes propres Pokémons, ligne après ligne, en utilisant les méthodes agiles des meilleurs dresseurs de tests unitaires.

Mais attention, un projet mal structuré est plus dangereux qu'un Ultralaser ! Prépare tes Pokéballs (et ton clavier), car nous allons explorer ensemble les rouages de la programmation orientée objet.

Prêt ? C'est parti ! En route pour l'aventure Java !"





Sommaire :

Bienvenue dans le monde des Pokémons !.....	2
Etape 1 : Téléchargement.....	4
Etape 2 : Installation.....	5
Etape 3 : Créer un nouveau projet.....	6
Etape 4 : Créer une classe.....	8
Etape 5 : Compiler la classe.....	9
Etape 6 : Instancier la classe.....	10
Etape 7 : Ajouter des attributs.....	11
Etape 8 : Instancier la classe avec des attributs.....	12
Etape 9 : Tests unitaires.....	14
Exercice : Création des autres tests unitaires.....	21
Etape 10 : Création d'une seconde classe Pokeball et association avec la classe Pokemon	22
Etape 11 : Méthode collaborative.....	25
Etape 12 : Fixturisation :.....	26
Etape 13 : Test en utilisant la fixture :.....	28



Etape 1 : Téléchargement

The screenshot shows the official website for BlueJ (bluej.org). The page features a large blue penguin icon on the left. The main content area has a light blue background with a white sidebar on the right. The sidebar contains a Java code snippet:

```
/** * Add a student to this LabClass. */
public void enrollStudent(Student newStudent)
{
    if(students.size() == capacity) {
        System.out.println("The class is full.");
    }
    else {
        students.add(newStudent);
    }
}
```

Below the code, there's a quote from James Gosling: "One of my favourite IDEs out there is BlueJ" — James Gosling, creator of Java.

The sidebar also includes logos for "Created by KINGS COLLEGE LONDON" and "Supported by ORACLE".

At the bottom, there's a section titled "Download and Install" for Version 5.5.0, released on 3 June 2025. It provides links for Windows, macOS, Ubuntu/Debian, and Other operating systems, each accompanied by its respective logo and system requirements.

Télécharge BlueJ via le lien suivant : <http://www.bluej.org/>

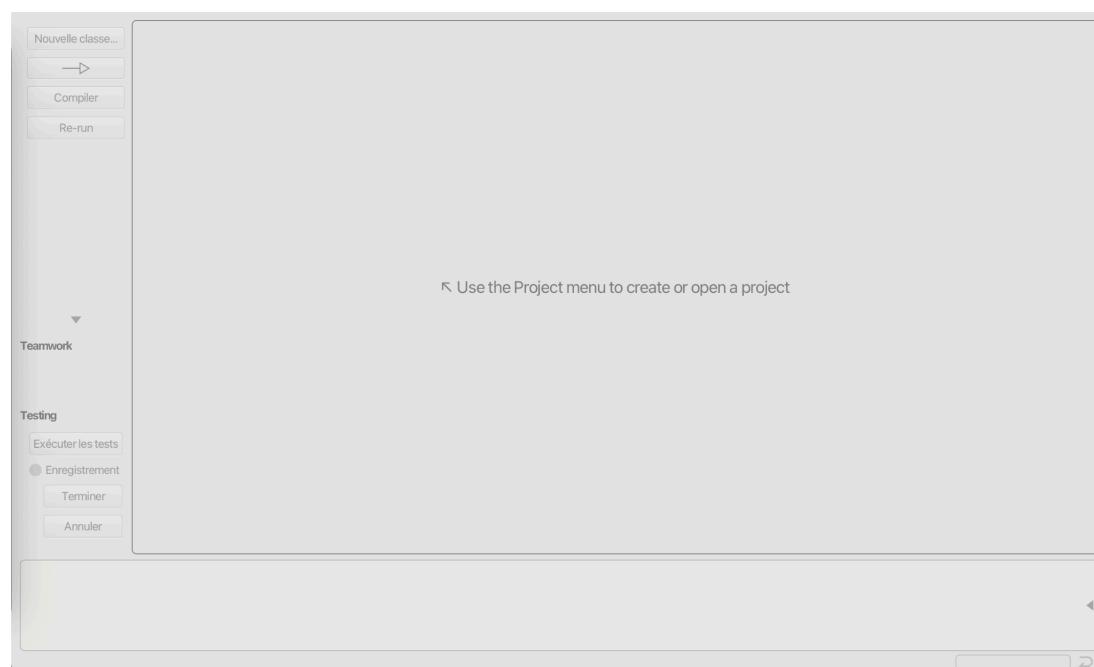
Pense à bien sélectionner la version adaptée à ton OS (le système d'exploitation de ton ordinateur)



Etape 2 : Installation

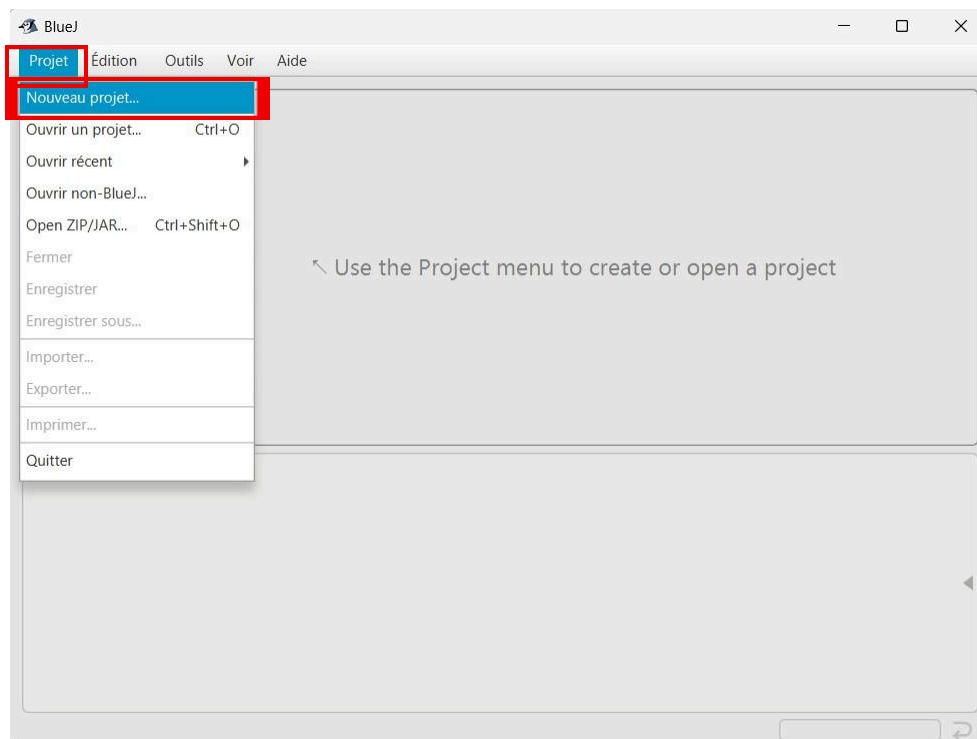


Suis les étapes d'installation de BlueJ, une fois bien installé, en ouvrant l'application tu devrais aboutir à cet écran :





Etape 3 : Créer un nouveau projet

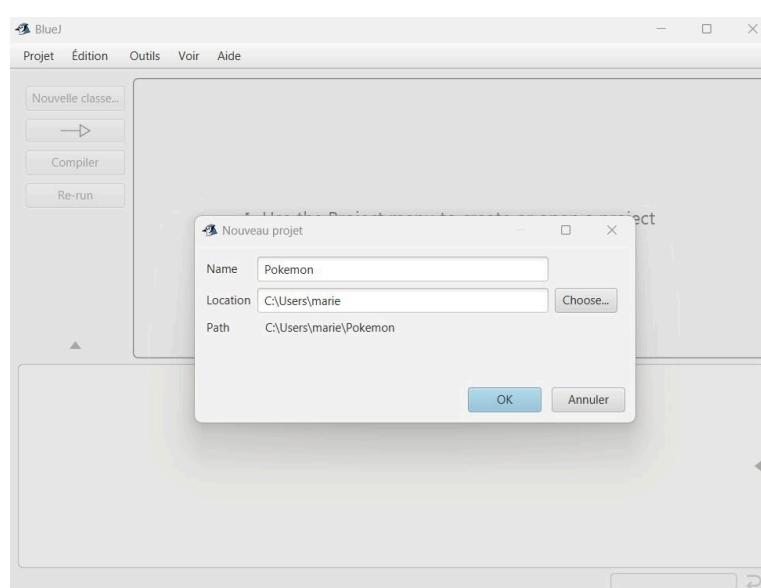


Clique sur le menu Projet, puis sur Nouveau projet...

Pour ce tutoriel, on prendra l'exemple de Pokémons, mais tu peux choisir autre chose comme exemple : des Personnes, des livres, des chansons etc.



N'utilise pas de caractères tels que des é è ê ï ou autres, il faut utiliser uniquement les caractères "normaux", tu remarqueras que nous utilisons toujours "Pokemon" au lieu de "Pokémon" qui pourrait créer de potentiels bugs





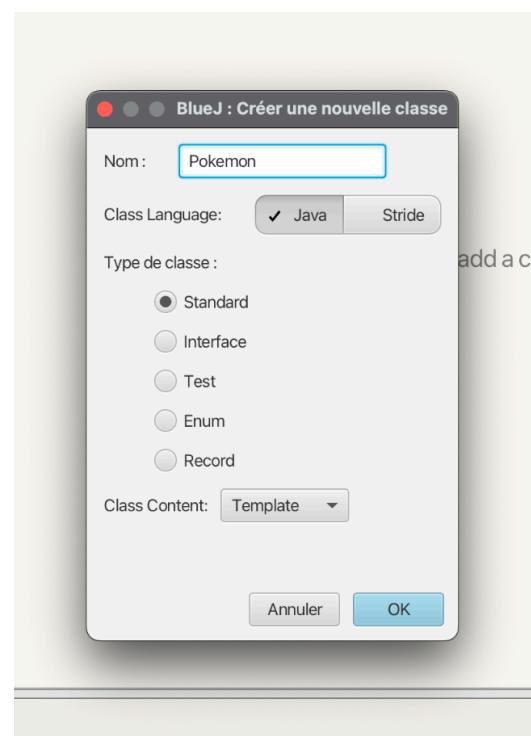
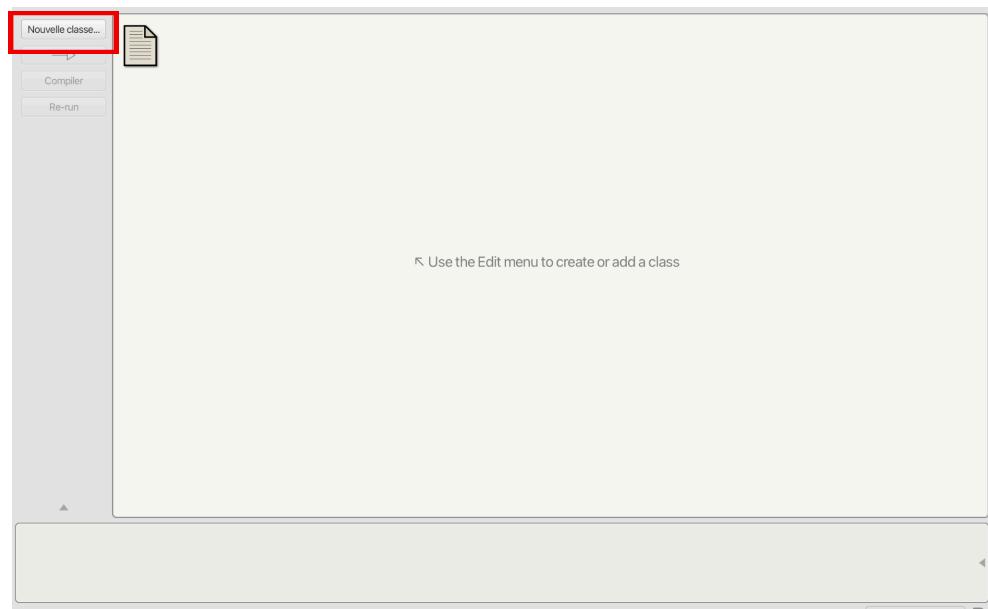
Une fois le nom rentré tu peux cliquer sur OK et Félicitations ! Tu viens de créer ton premier projet Java !

Normalement, les boutons qui étaient grisés ne le sont plus et tu vas pouvoir passer à la prochaine étape de ta découverte du Java, la création de classes !



Etape 4 : Créer une classe

Pour créer notre classe Pokémon, on va cliquer sur le bouton "Nouvelle classe" à gauche, plusieurs paramètres s'affichent : rentre le nom de la classe. Tu peux laisser les paramètres par défaut pour cette classe principale.



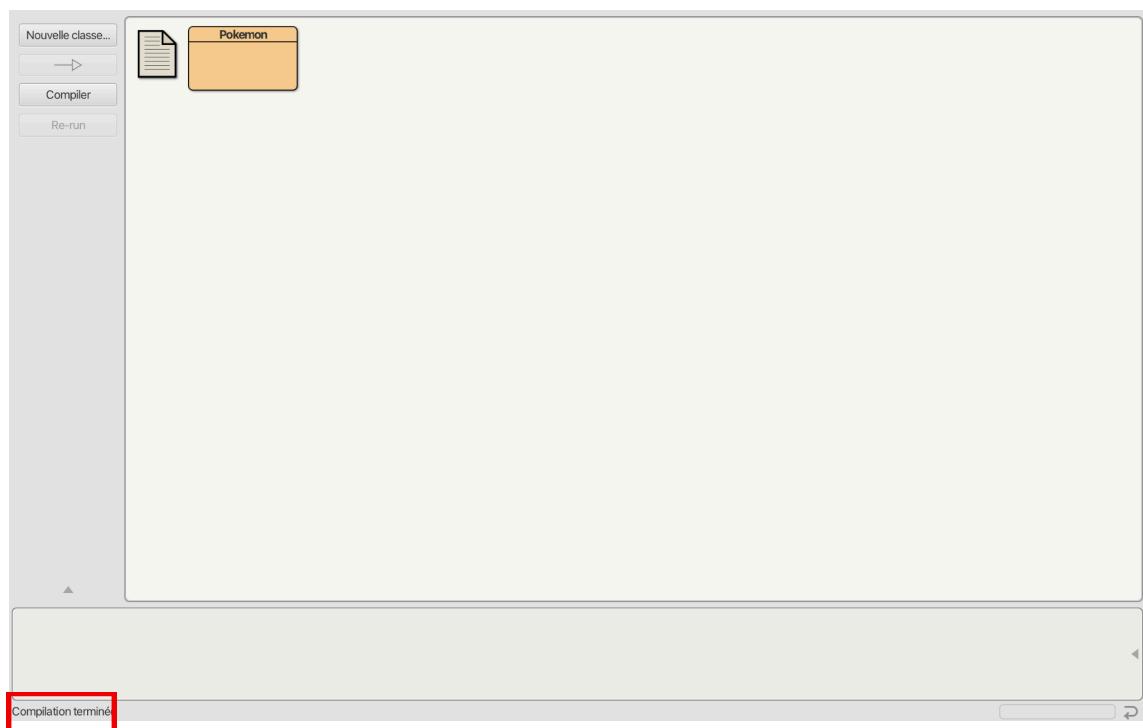
Bravo, tu viens de créer ta première classe !



Etape 5 : Compiler la classe



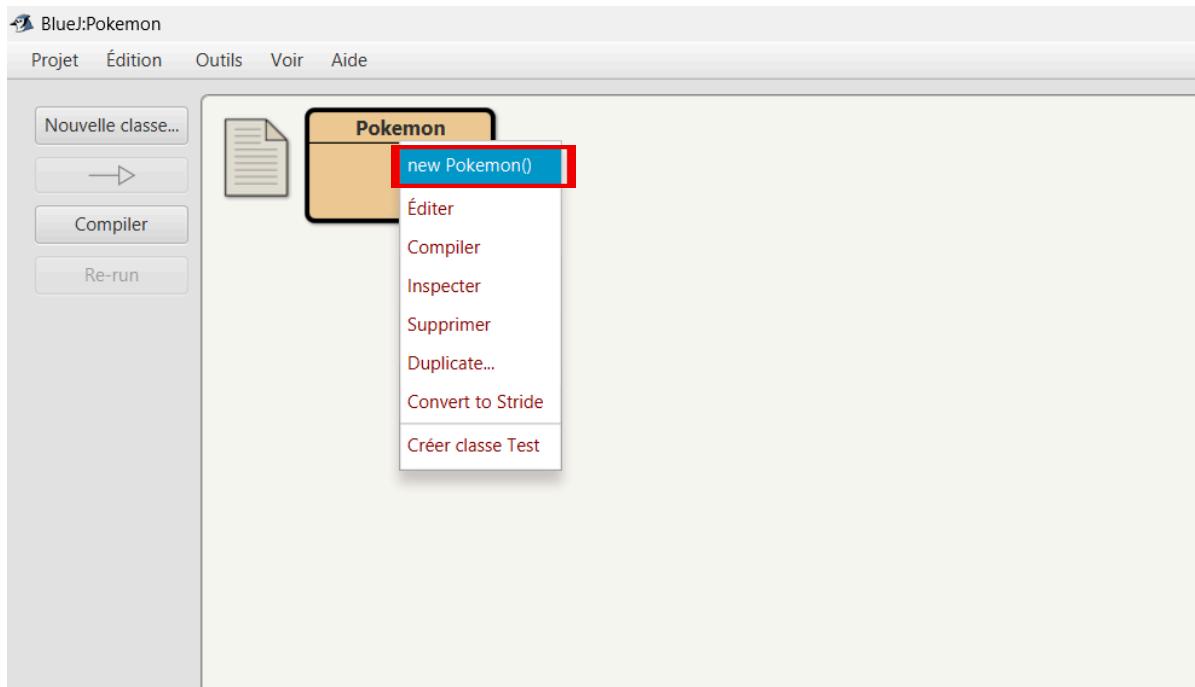
La classe Pokemon est apparue sur l'écran. Il faut maintenant la compiler en cliquant sur "Compiler". Quand la compilation est terminée, "Compilation terminée" est indiquée en bas à gauche de l'écran.



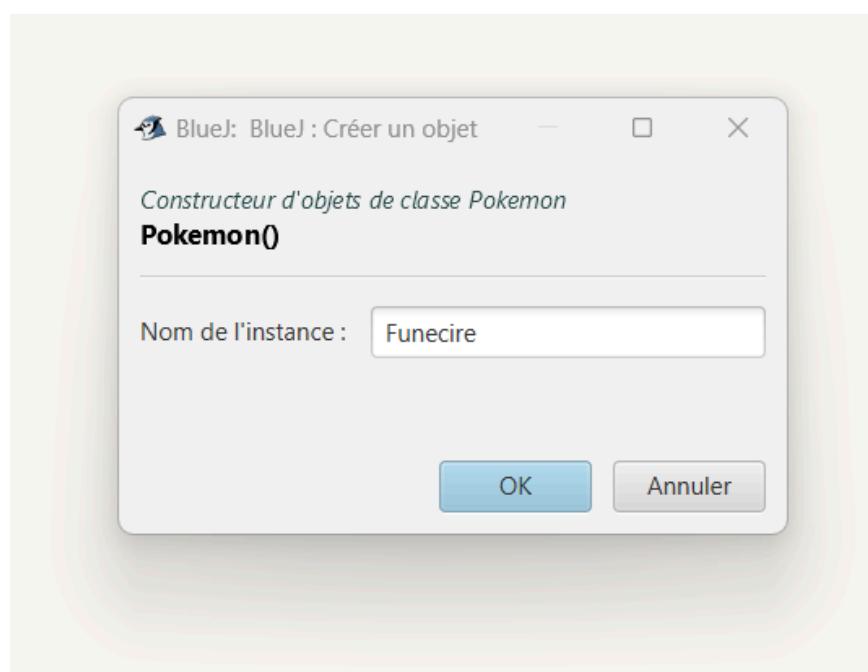


Etape 6 : Instancier la classe

Il est temps de créer notre premier Pokémons! Nous avons sélectionné pour cet exemple Funécire, un Pokémons de la cinquième génération.



Fais clic droit sur la classe "Pokemon" et cliquer sur "new Pokemon". Une fenêtre s'ouvre dans laquelle on indique Funécire, le nom du Pokémons. En cliquant sur "OK", Funécire est créé!





Etape 7 : Ajouter des attributs

Chaque Pokémons possède un ou deux types ! Il est temps de rajouter ces attributs à notre classe.



Fais un clic droit sur la classe "Pokemon" puis clique sur "Editer". Une interface s'ouvre :

```

7 */
8 public class Pokemon
9 {
10
11     private String type1;
12     private String type2;
13
14     public Pokemon(String type1, String type2){
15         this.type1 = type1;
16         this.type2 = type2;
17     }
18
19     public String getType1(){
20         return this.type1;
21     }
22
23     public void setType1(String newType1){
24         this.type1 = newType1;
25     }
26
27     public String getType2(){
28         return this.type2;
29     }
30
31     public void setType2(String newType2){
32         this.type2 = newType2;
33     }
34
35

```

The code editor shows the following structure:

- Section 1:** The constructor `public Pokemon(String type1, String type2)` is highlighted with a red box and circled with a blue circle labeled '1'.
- Section 2:** The getter method `public String getType1()` is highlighted with a red box and circled with a blue circle labeled '2'.
- Section 3:** The setter method `public void setType1(String newType1)` is highlighted with a red box and circled with a blue circle labeled '3'.

Dans cette interface, on définit plusieurs éléments :

1. Des attributs : ce sont les propriétés du Pokémons. Dans cet exemple, nous créons type1 et type2 ;
2. Un constructeur : pour pouvoir donner des types aux Pokémons dès leur création, crée un constructeur qui prend deux types en entrée ;
3. Des méthodes : pour manipuler les types créés, par exemple ajouter un type à un pokémons ou récupérer son type, il faut créer des méthodes.

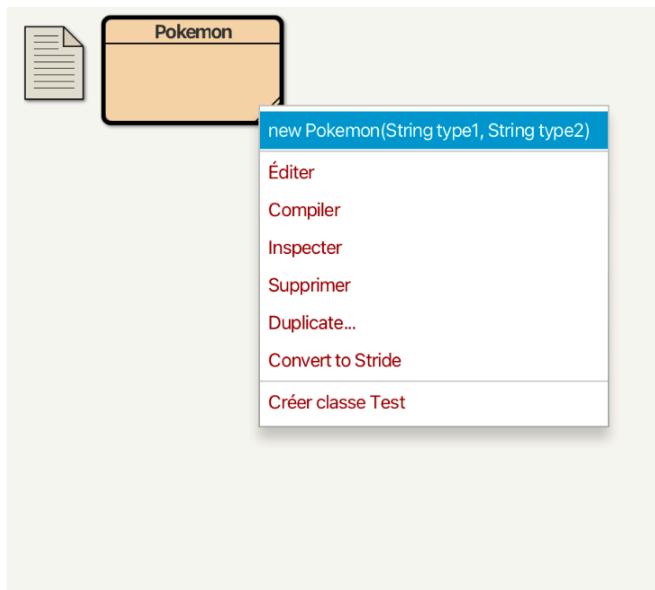
Une fois le paramétrage terminé, cliquer sur "Fermer".

N'hésite pas à définir d'autres attributs et les méthodes que tu souhaites.

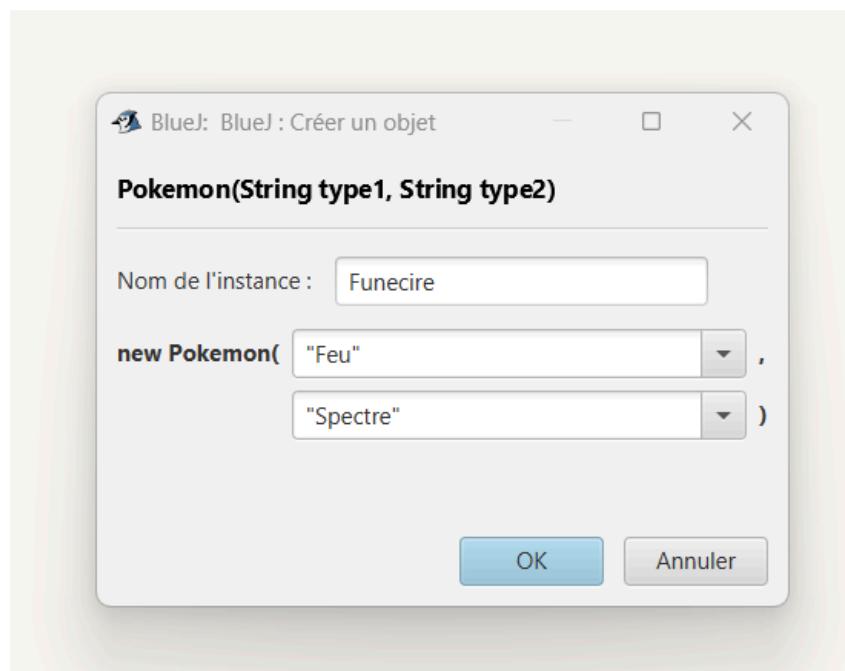


Etape 8 : Instancier la classe avec des attributs

Il est temps de créer de nouveau Funécire, mais avec ses types désormais (Feu et Spectre). Sur la classe "Pokemon", faire clic droit puis "new Pokemon(String type1, String type2)".



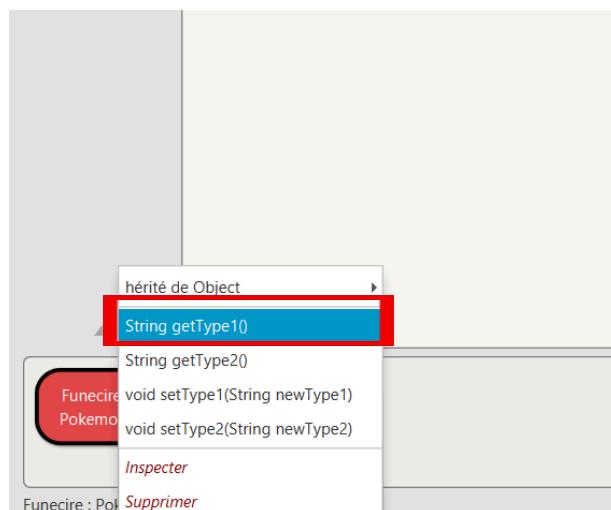
Dans l'interface de création, il faut désormais indiquer le nom de l'instance, mais aussi ses types. Clique sur "OK" pour créer l'objet.



Attention : Les attributs sont de type String, donc il faut marquer le nom des types entre guillemets.

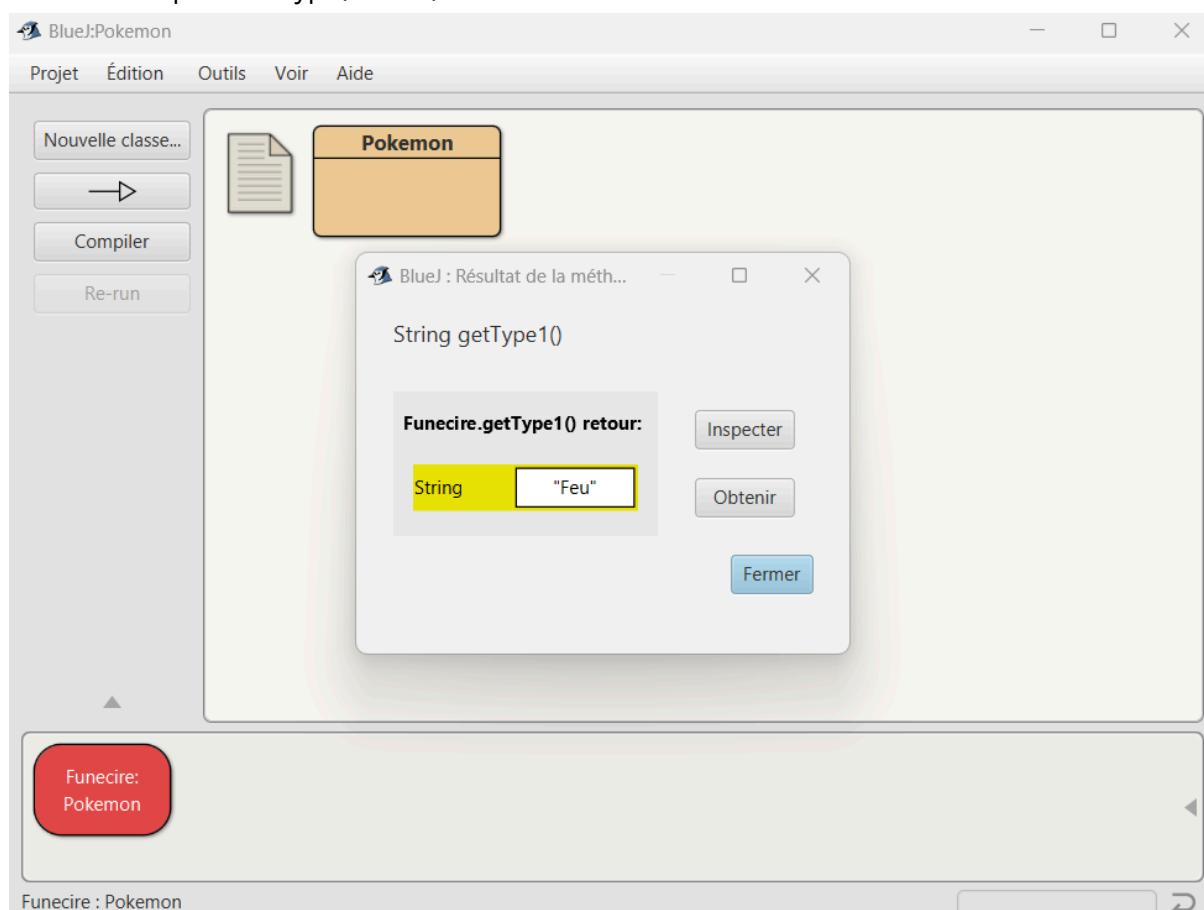


Les types de Funécire sont créés, essayons maintenant de récupérer la valeur du premier type via la méthode `getType1()`.



Faire clic droit sur l'instance créée et cliquer sur la méthode qu'on veut tester, ici `String getType1()`.

La valeur du premier type, "feu", est affichée dans la fenêtre du résultat de la méthode.

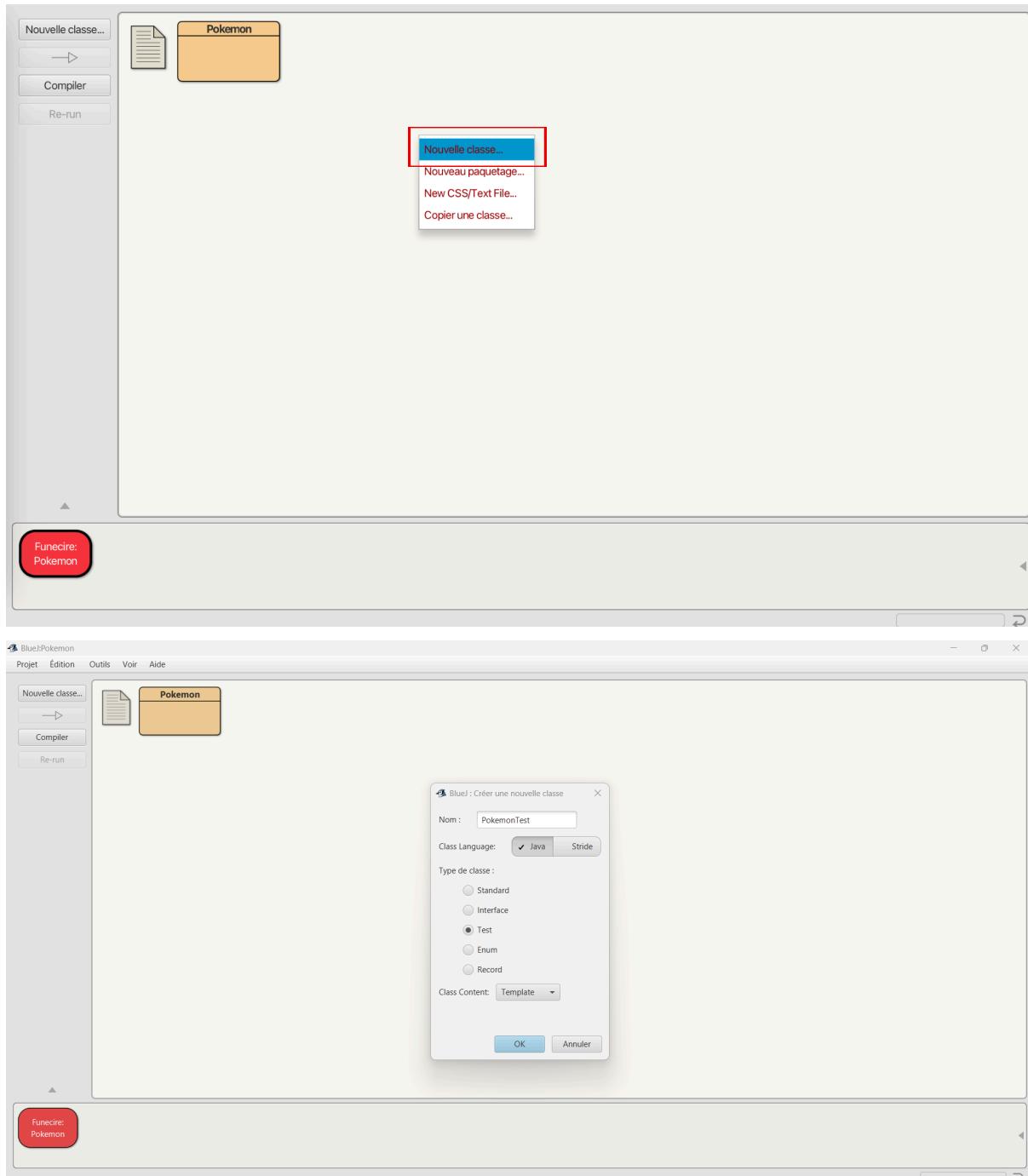




Etape 9 : Tests unitaires

Maintenant qu'on a créé la classe et nos méthodes, il est temps de les tester !

Pour cela on va créer une nouvelle classe Java qui sera une classe de test, on commence donc par créer une classe :

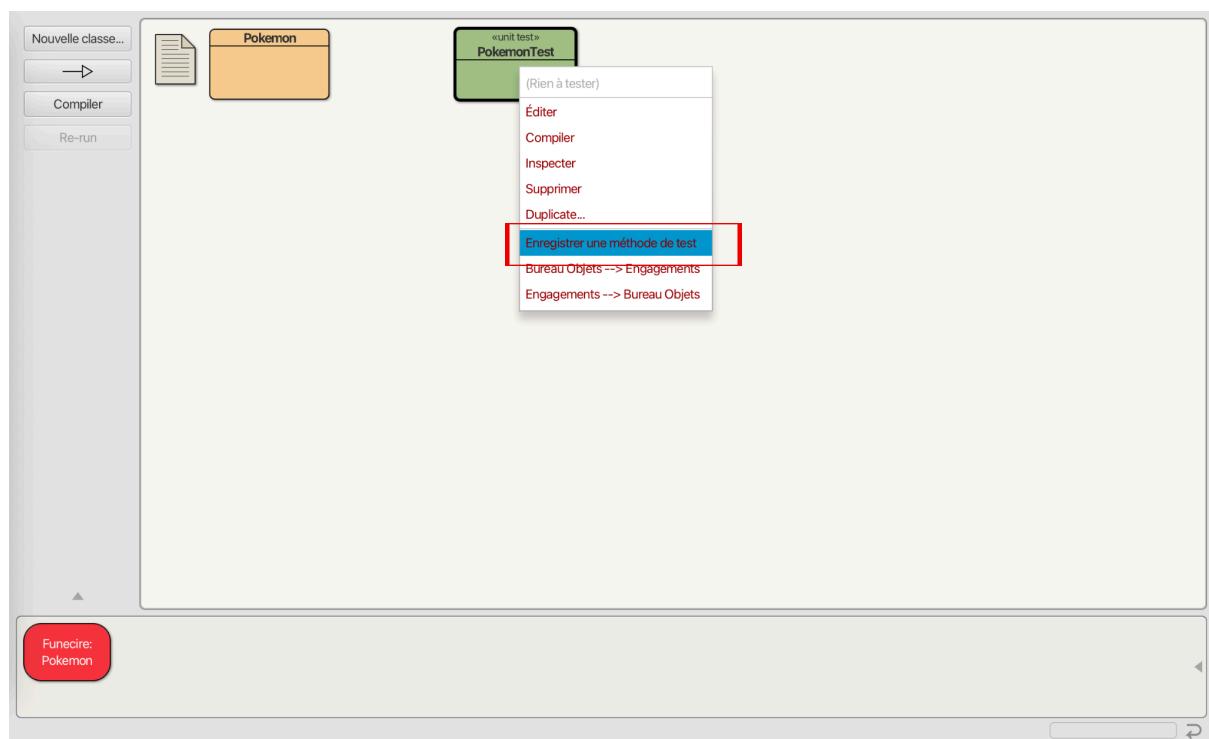


On va appeler cette classe `PokemonTest` mais cette fois, on ne va pas laisser le Type de classe "Standard" mais on va sélectionner "Test" comme type.



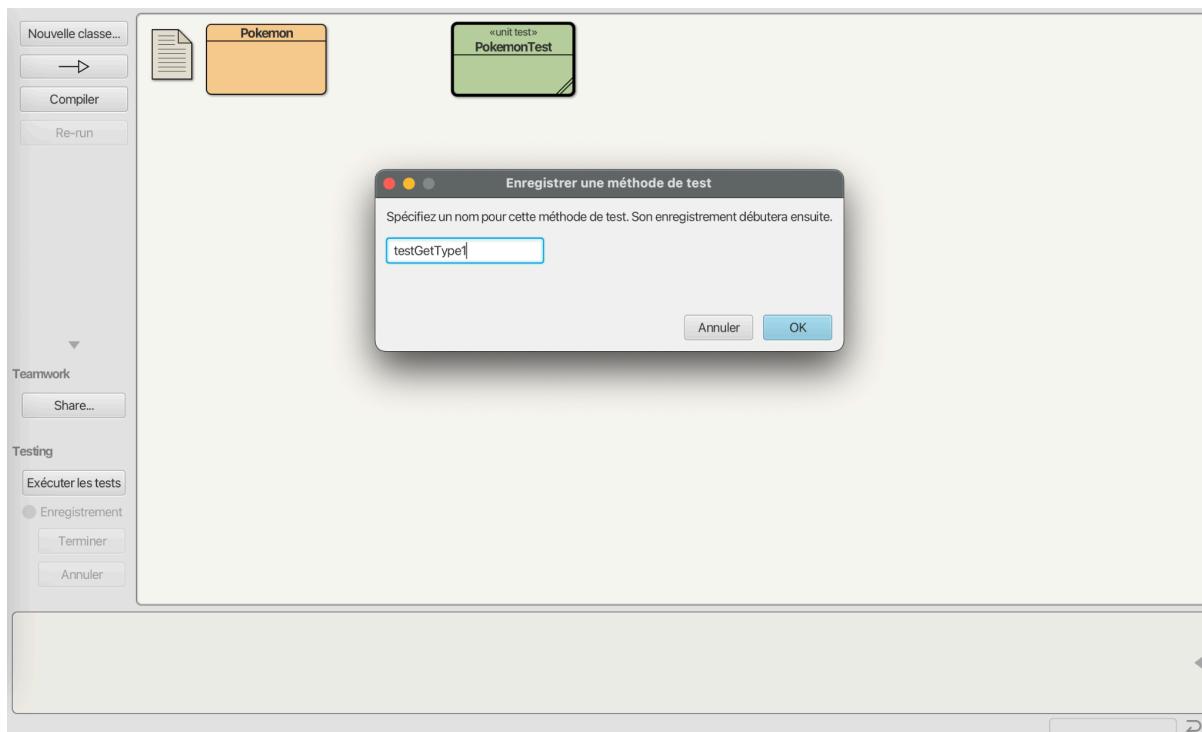
Une fois créée tu remarqueras que la classe s'affiche en vert et non pas en jaune comme pour la classe standard Pokemon.

Maintenant qu'on a créé notre classe test, il est temps de créer nos méthodes de tests ! Pour cela, fais un clic droit sur ta nouvelle classe test, et sélectionne "Enregistrer une méthode de test".



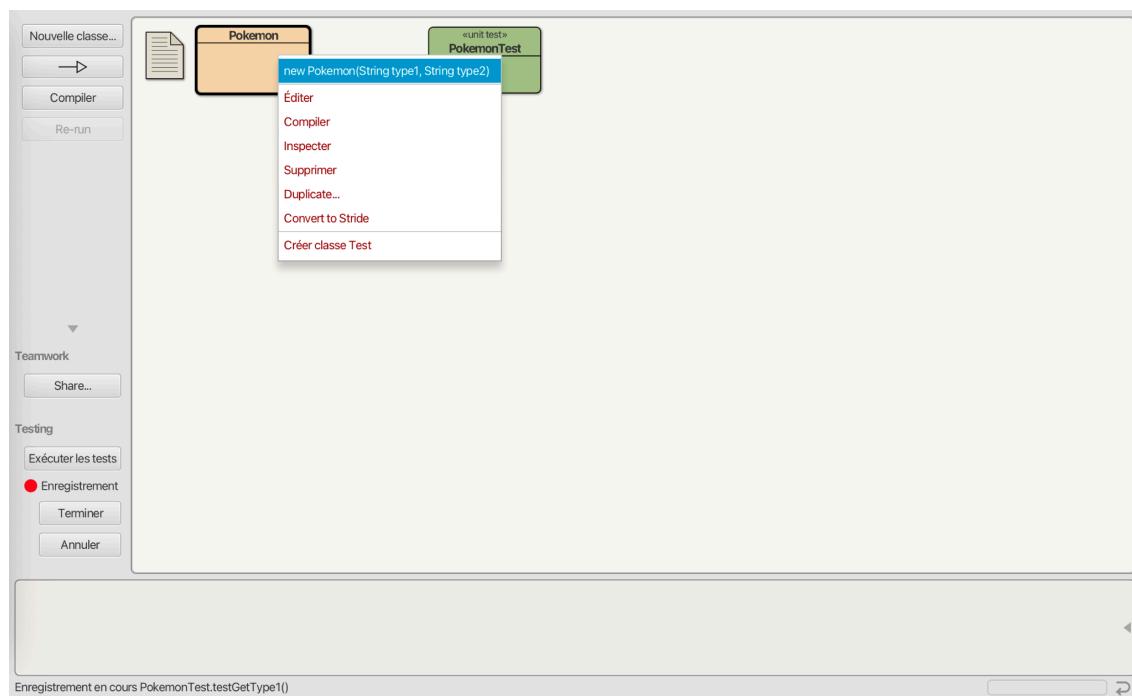


Pour le nom de la méthode, on ne va pas faire de folie, on va juste l'appeler "test" accolé au nom de la méthode que l'on souhaite tester.

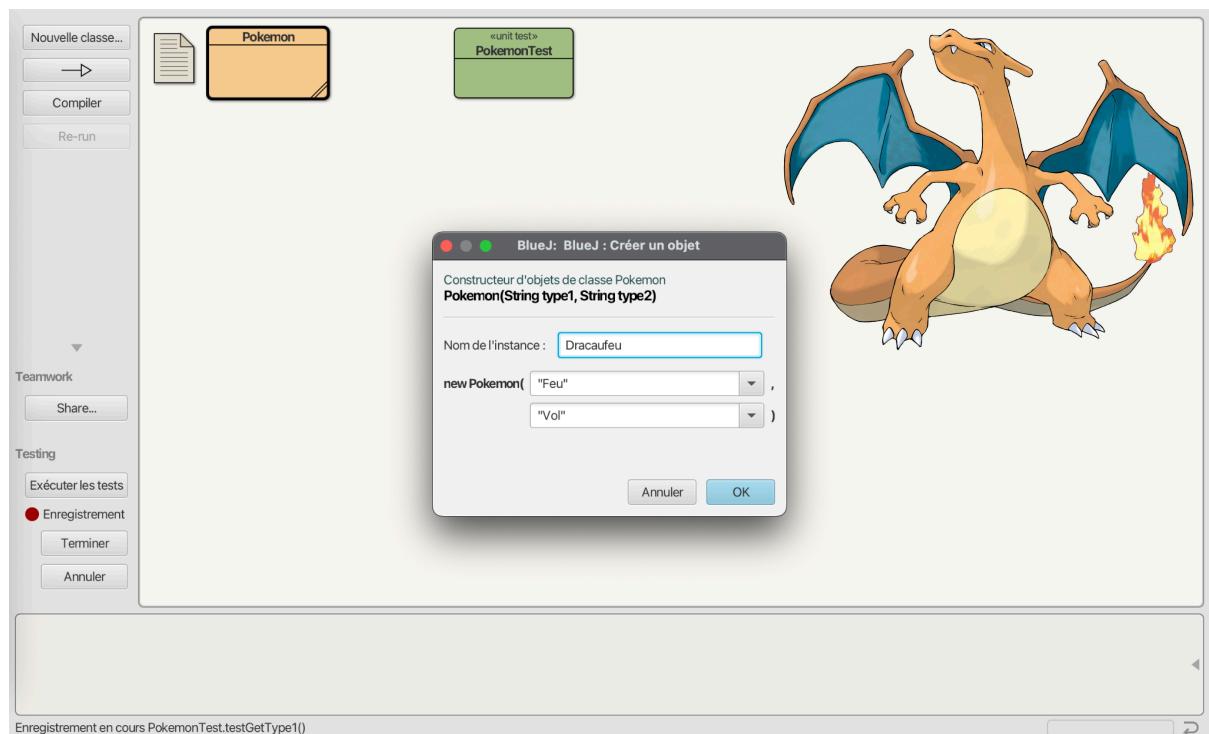


Une fois cela fait, tu vas voir qu'en bas à gauche de ton écran tu as un point rouge avec écrit "Enregistrement" à côté, cela veut dire que toutes les actions que tu feras à partir de maintenant, et ce jusqu'à cliquer sur le bouton "Terminer", seront enregistrés par BlueJ comme les actions que dois reproduire la fonction test.

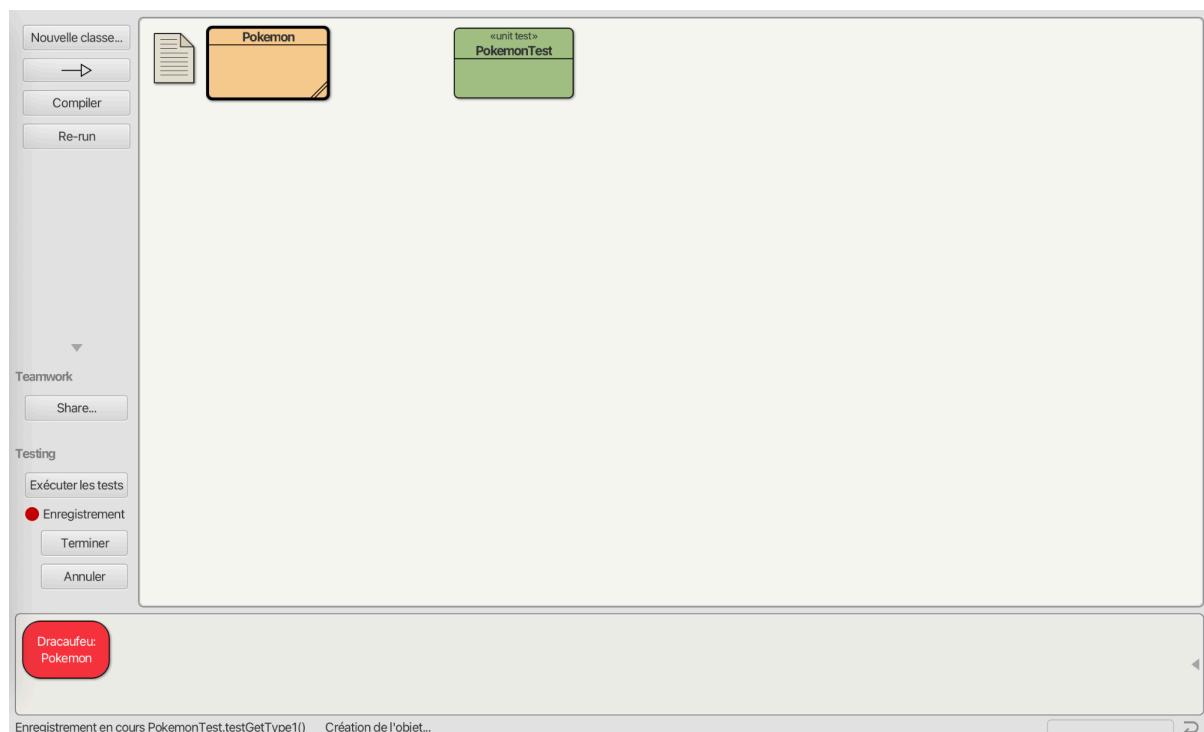
Tu remarqueras que notre Pokemon qu'on avait créé a disparu, en effet, les tests doivent être faits sur des objets qu'on va créer spécialement pour eux. Il faut créer un objet pour chaque test pour être sûr que les méthodes ne s'affectent pas entre elles, c'est le principe d'isolation.



On va donc créer un nouveau Pokémons.



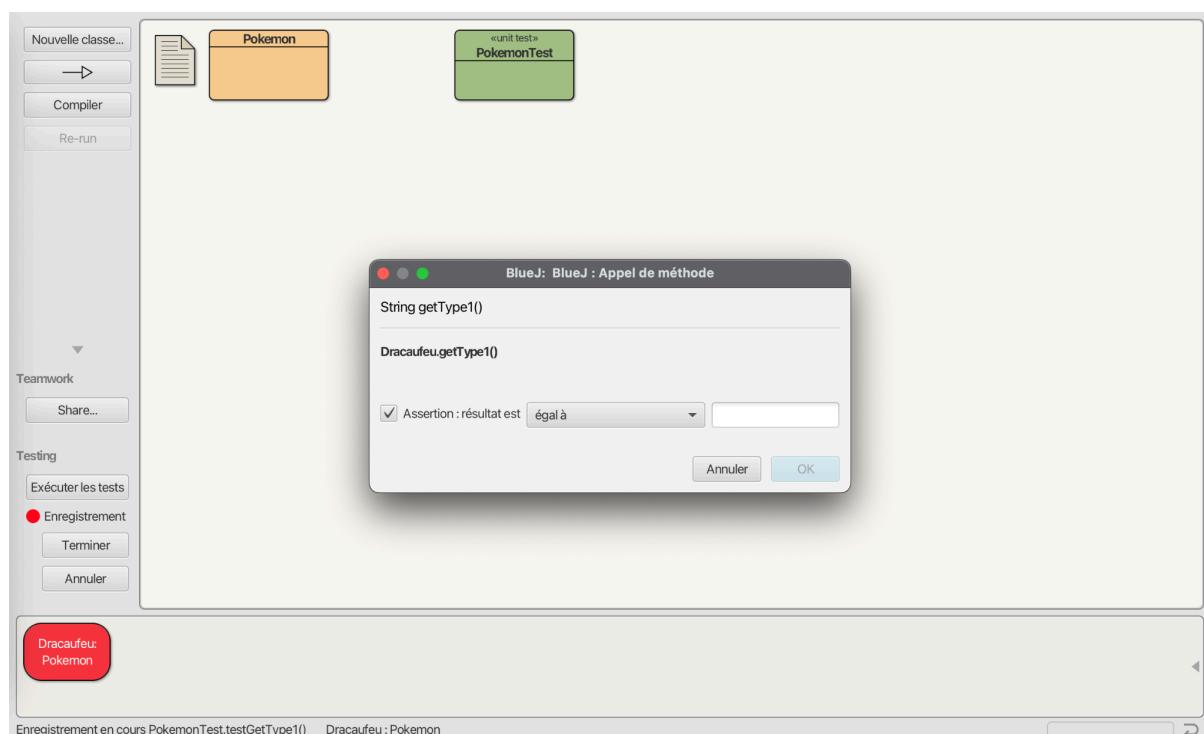
Cette fois ci, pour nos tests, on va utiliser le Pokémons Dracaufeu comme exemple. Dracaufeu c'est un Pokémons de la première génération de type Feu et Vol, on rentre donc "Feu" et "Vol" dans les types.



Le Pokémon créé pour le test va apparaître en bas à gauche de l'écran.

Ensuite, on va faire un clic droit sur le Pokemon et cliquer sur la méthode que l'on veut tester, ici `getType1`.

Quand on est entrain d'enregistrer une méthode de test et qu'on fait appel à une fonction qui renvoie quelque chose, BlueJ nous propose de vérifier que la fonction renvoie bien ce que l'on souhaite. C'est l'assertion, on vérifie que le résultat qu'on a obtenu est celui attendu.





Nouvelle classe... ➔ Compiler Re-run

«unit test» PokemonTest

Pokemon

Teamwork Share...

Testing Exécuter les tests Enregistrement Terminer Annuler

Dracaufeu: Pokemon

Enregistrement en cours PokemonTest.testGetType1() Dracaufeu : Pokemon

Quand on a créé Dracaufeu, son type1 était "Feu", donc on met "Feu" comme résultat attendu.

Nouvelle classe... ➔ Compiler Re-run

«unit test» PokemonTest

Pokemon

Teamwork Share...

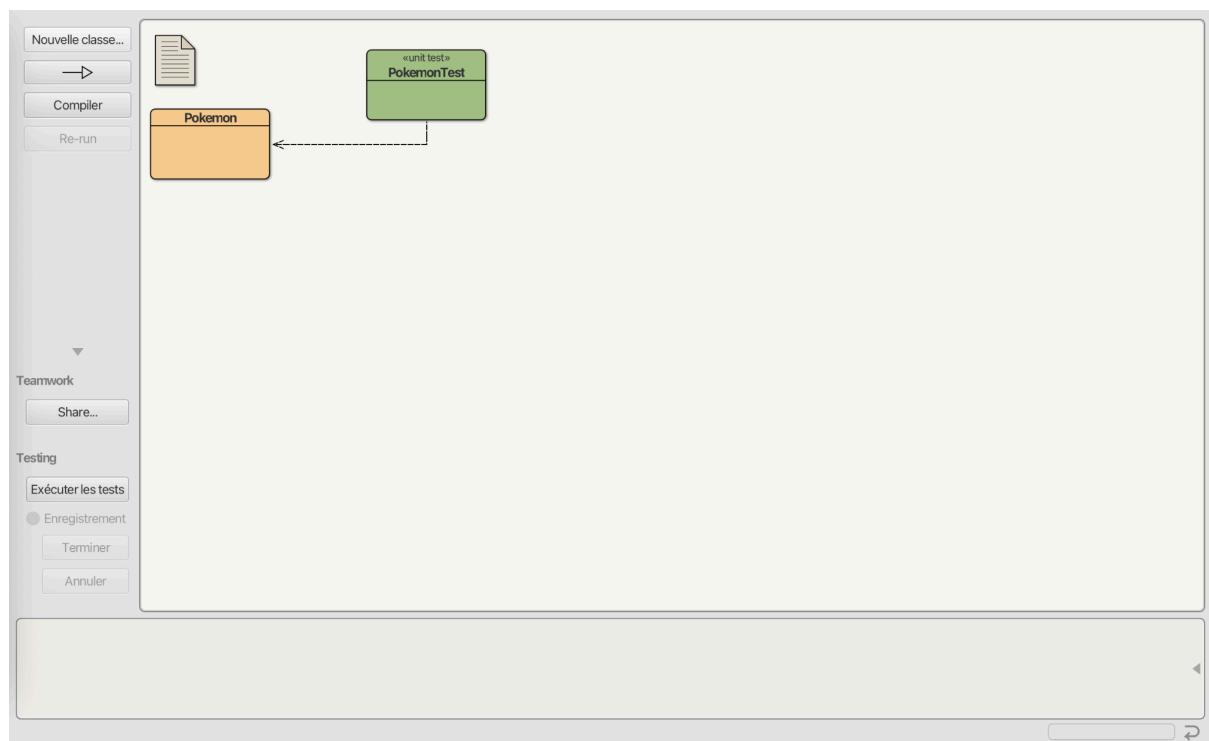
Testing Exécuter les tests Enregistrement Terminer Annuler

Dracaufeu: Pokemon

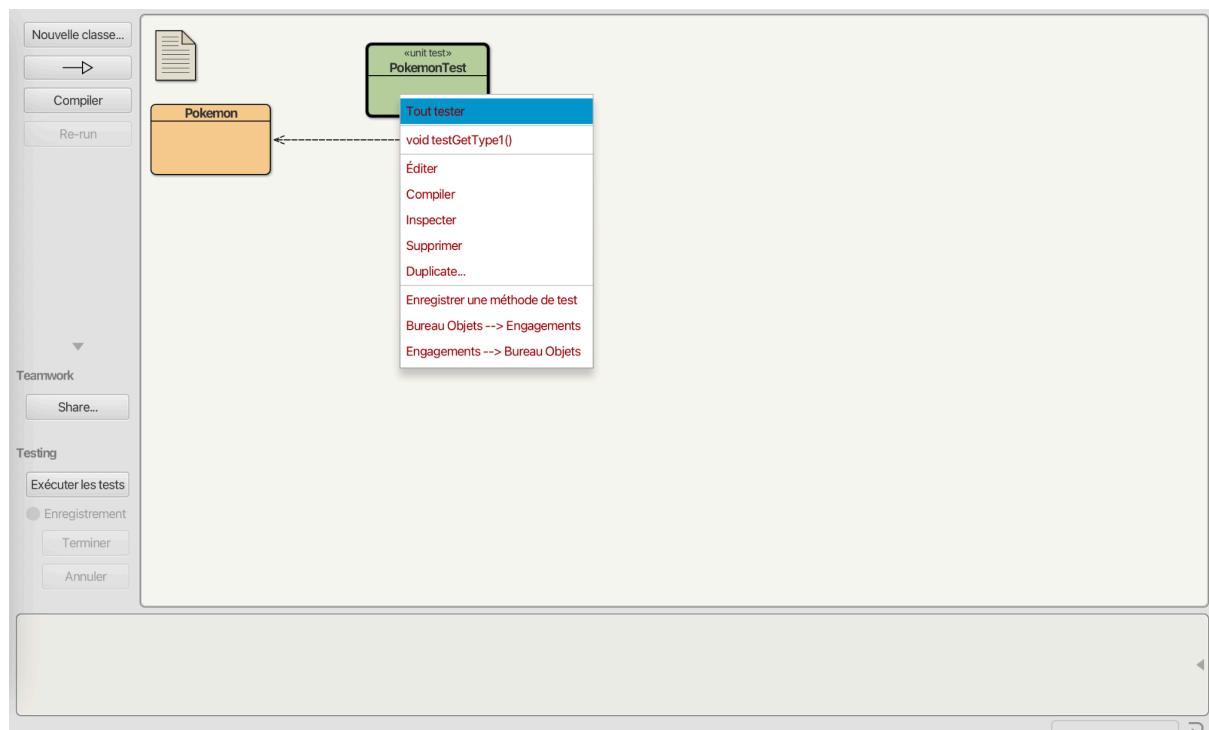
Enregistrement en cours PokemonTest.testGetType1() Dracaufeu : Pokemon



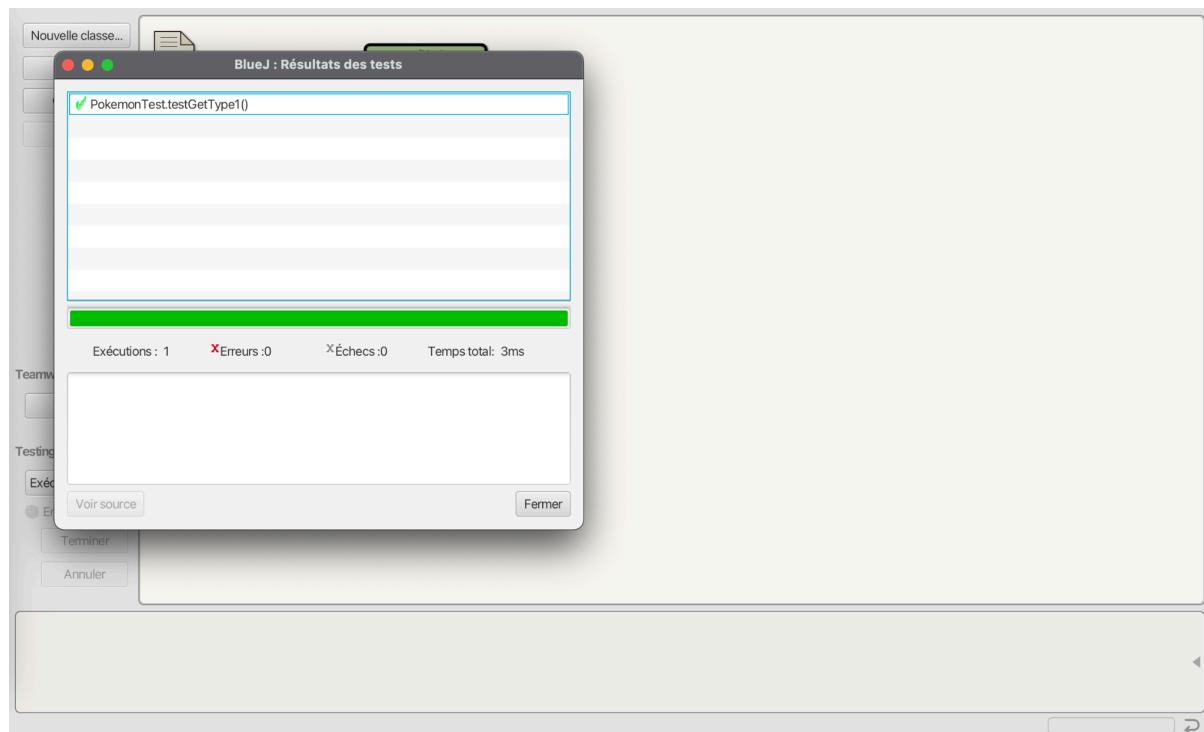
BlueJ affiche ensuite le résultat obtenu, cela te permet de savoir dès la création du test, si celui est réussi. Clique sur le bouton "Fermer" puis sur "Terminer" en bas à gauche.



Une fois terminé, tu verras qu'une flèche part désormais de ta classe PokemonTest vers ta classe Pokemon, c'est parce qu'on utilise la classe Pokemon dans une des méthodes tests définies dans PokemonTest.



Ensuite, tu peux cliquer sur "Tout tester" pour vérifier que le test s'effectue bien.

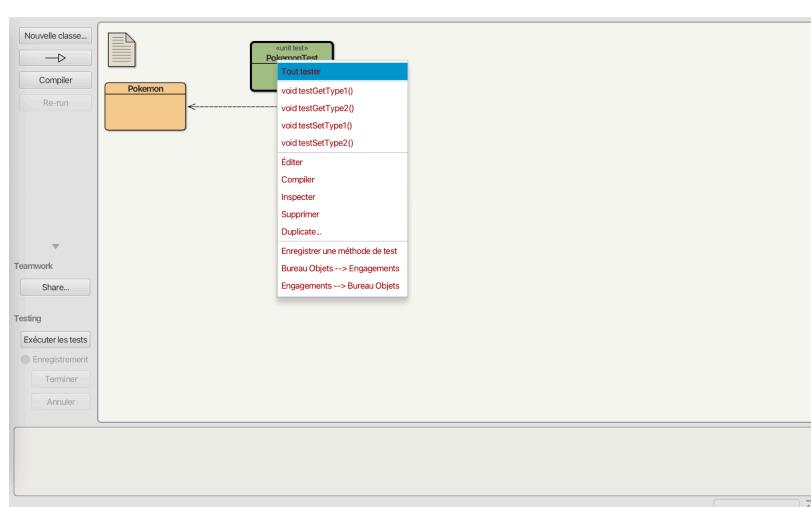


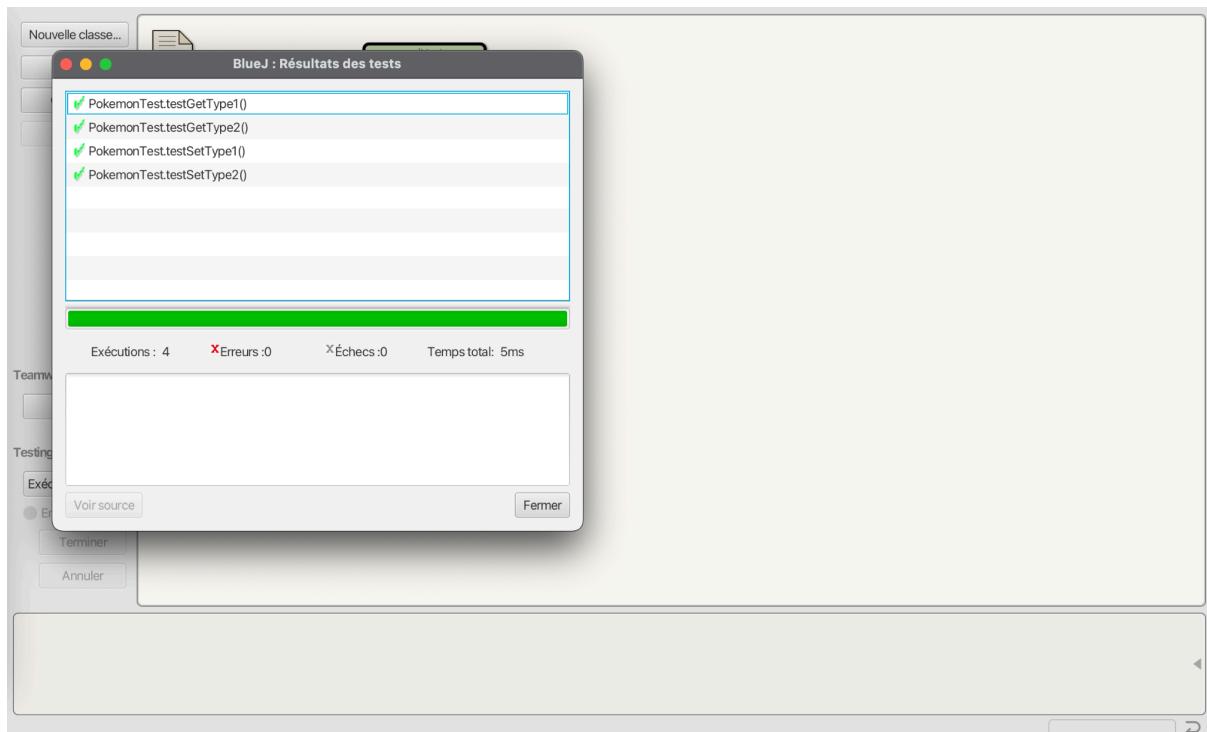
Normalement tu arriveras à cet écran, cela veut dire que le test fonctionne bien !

Exercice : Création des autres tests unitaires

Maintenant, il va falloir de créer les autres tests unitaires, pour cette partie, c'est à toi de travailler, tu dois créer les tests pour les méthodes `getType2`, `setType1` et `setType2`.

Indice : pour tester les méthodes `setType`, utilise les puis vérifie leur résultat en utilisant `getType`. (Crée "par erreur" un Dracaufeu de type eau vol par exemple, puis modifie son type avec `setType1` et vérifie avec `getType1` que son nouveau type1 est bien le type feu)





Tu devrais arriver à ces deux écrans, félicitations tu as réussi la première partie de ce tutoriel !

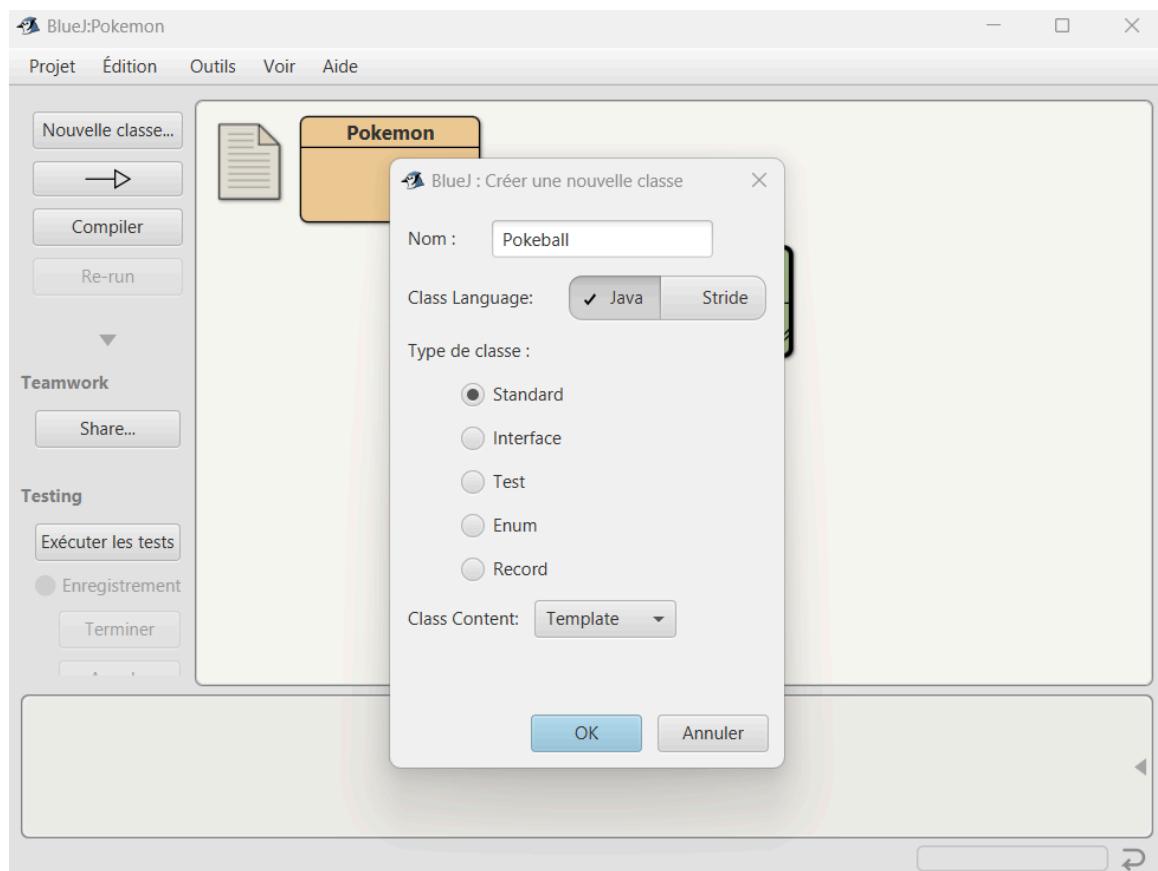
Etape 10 : Création d'une seconde classe Pokeball et association avec la classe Pokemon

Alors que tu te promenais dans les hautes herbes, tu tombes sur un Funécire sauvage ! Il a l'air d'être le compagnon parfait. Pour qu'il rejoigne ton équipe, il faut le capturer à l'aide d'une Pokeball.

Cela introduit le concept de la relation entre les objets :

- Un Pokémon peut être ou non dans une Pokeball
- Une Pokeball peut contenir ou non un Pokémon.

Il est désormais temps de créer la classe "Pokeball" de la même manière que la classe "Pokemon", et de la compiler.



Pour qu'un Pokémon puisse avoir une Pokéball, il faut lui rajouter un nouvel attribut "pokeball". Pour cela, faire clic droit sur Pokemon puis cliquer sur "Editer". Ajoute l'attribut pokeball et des méthodes pour y accéder.



```

Pokemon - Pokemon
Classe Édition Outils Options
Pokemon x
Compiler Défaire Couper Copier Coller Chercher Fermer Implémentation
6 * @version 1
7 */
8 public class Pokemon {
9
10
11     private String type1;
12     private String type2;
13     private Pokeball pokeball;
14
15     public Pokemon(String type1, String type2){
16         this.type1 = type1;
17         this.type2 = type2;
18     }
19
20     public Pokemon(String type1, String type2, Pokeball pokeball){
21         this.type1 = type1;
22         this.type2 = type2;
23         this.pokeball = pokeball;
24     }
25
26     public Pokeball getPokeball(){
27         return this.pokeball;
28     }
29
30     public void setPokeball(Pokeball newPokeball){
31         this.pokeball = newPokeball;
32     }

```



Fais la même chose pour la classe Pokeball qui peut contenir un Pokémons :

The screenshot shows a Java code editor window titled "Pokeball - Pokemon". The menu bar includes "Classe", "Édition", "Outils", and "Options". The toolbar contains "Compiler", "Défaire", "Couper", "Copier", "Coller", "Chercher", and "Fermer". A dropdown menu "Implémentation" is open. The code editor displays the following Java code:

```
8 public class Pokeball
9 {
10     private Pokemon pokemon;
11     private int prix;
12
13     /**
14      * Constructeur d'objets de classe Pokeball
15      */
16     public Pokeball(int prix)
17     {
18         this.prix = prix;
19     }
20
21     public Pokeball(int prix, Pokemon pokemon){
22         this.prix = prix;
23         this.pokemon = pokemon;
24     }
25
26     public Pokemon getPokemon() {
27         return this.pokemon;
28     }
29
30     public int getPrix() {
31         return this.prix;
32     }
33
34     public void setPokemon(Pokemon pokemon) {
35         this.pokemon = pokemon;
36     }
37
38     public void setPrix(int prix) {
39         this.prix = prix;
40     }
}
```



Etape 11 : Méthode collaborative

Maintenant qu'un Pokemon et une Pokeball peuvent être liés dans notre code, nous pouvons accéder aux informations du Pokéémon directement depuis la Pokeball elle-même !

C'est très pratique. Par exemple, à partir d'une Pokeball, on peut non seulement récupérer le Pokéémon contenu, mais plus précisément déterminer et afficher ses types. Il faut gérer les trois situations possibles :

1. Le Pokéémon n'a aucun type défini, la Pokeball est vide ;
2. Il est d'un type unique ;
3. Il est d'un double type.

Pour ajouter cette nouvelle capacité de trouver les types du Pokéémon à notre Pokeball, nous devons créer une nouvelle méthode.

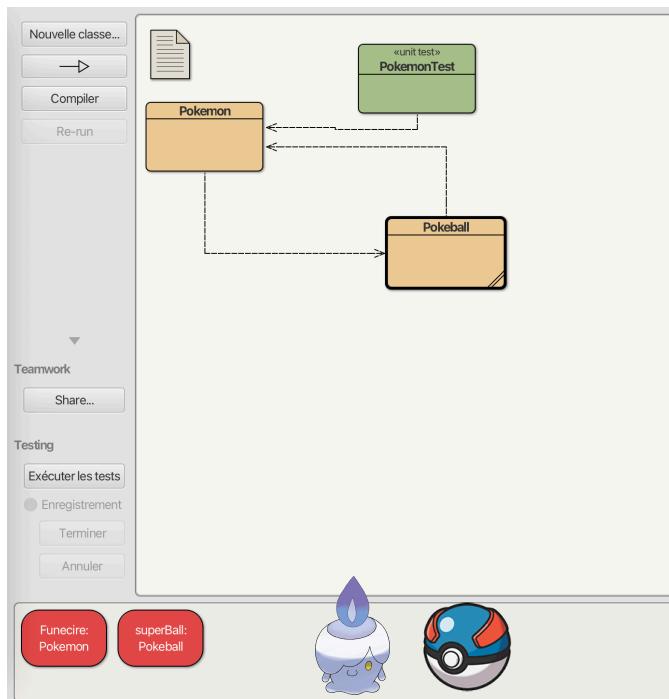
Ouvre l'interface d'édition de la classe Pokeball et ajoute une nouvelle méthode :

```
public String afficherTypePokemon() {  
    if (this.pokemon == null) {  
        return "Cette Pokeball est vide. Aucun type à afficher.";  
    }  
  
    String type1 = this.pokemon.getType1();  
    String type2 = this.pokemon.getType2();  
  
    if (type1 != null && type2 != null) {  
        return "De types " + type1 + " et " + type2 + ".";  
    } else if (type1 != null) {  
        return "De type " + type1 + ".";  
    } else {  
        return "Le Pokéémon dans cette Pokeball n'a pas de type défini.";  
    }  
}
```



Etape 12 : Fixturisation :

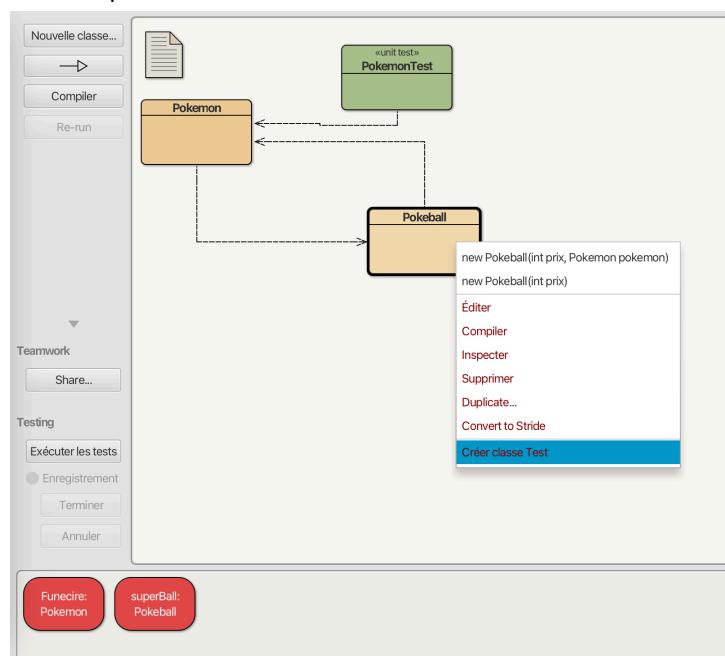
Maintenant, on va chercher à tester cette fonction, pour cela, plutôt que de faire comme dans la première partie et de créer un Pokémon pour chaque test que l'on va créer, on va créer un Pokémon et sa pokéball qui serviront ensuite d'environnement test.

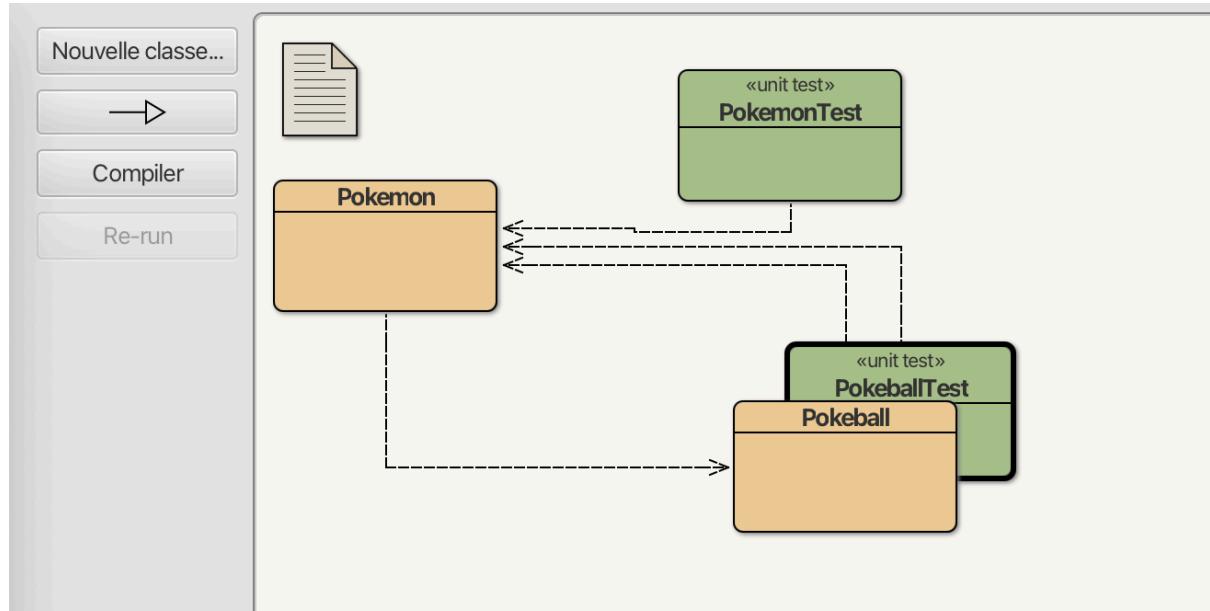


On va recréer un Funécire comme Pokémon de type Feu et Spectre et une Ball comme Pokéball dont le prix est de 600 dollars.

Attention, il faut bien mettre le Funécire comme attribut de la superBall et la superBall comme attribut du Funécire. Pour cela, tu devras forcément en créer au moins un premier vide et lui ajouter comme attribut le suivant. Utilise les setter qu'on a utilisés avant :)

Une fois que tu as créé ces deux objets comme ci-dessus, il faut que tu fasses un clic droit sur Pokeball et que tu cliques sur "Créer classe Test"





Cela aura pour effet de créer ta classe test directement liée à ta classe Pokeball.

Maintenant, pour "fixturiser", il faut faire un clic droit sur la classe PokeballTest et cliquer sur "Bureau Objets -> Engagements", cela va ajouter les objets que tu avais défini en bas, donc ici Funcerie et superBall, dans l'environnement de test de tous les tests de PokeballTest. Tu remarqueras que cela fait "disparaître" les objets de ta barre d'objets.

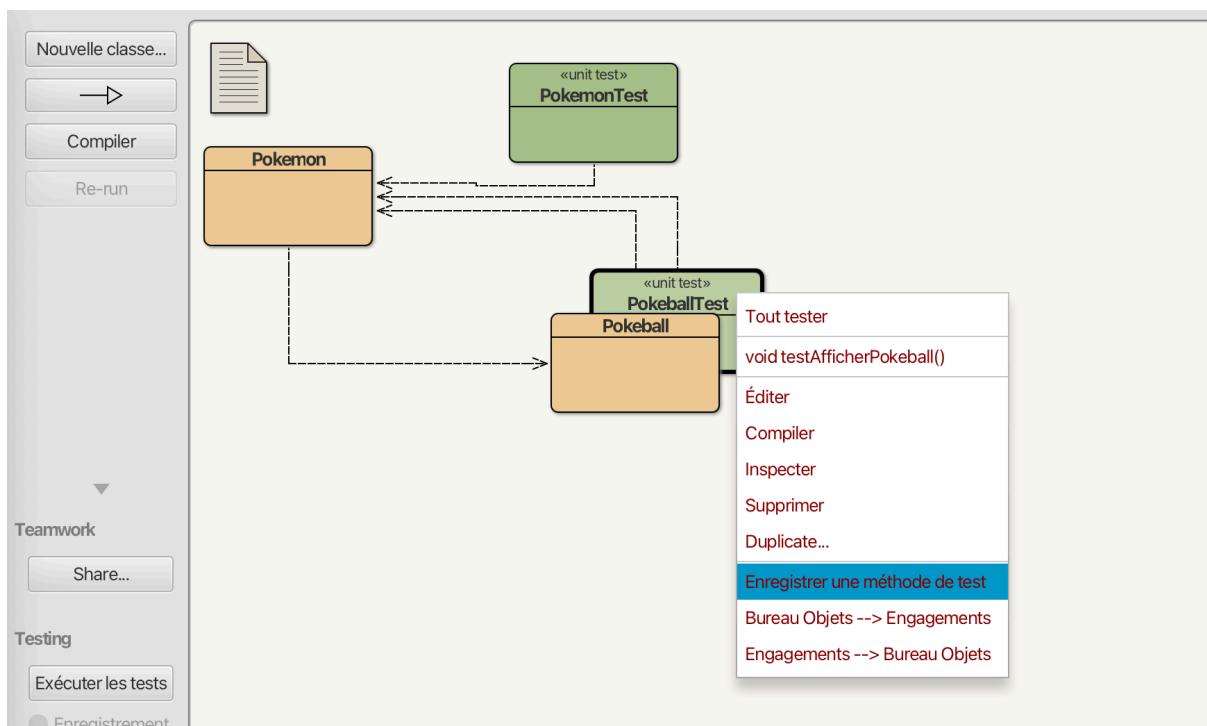




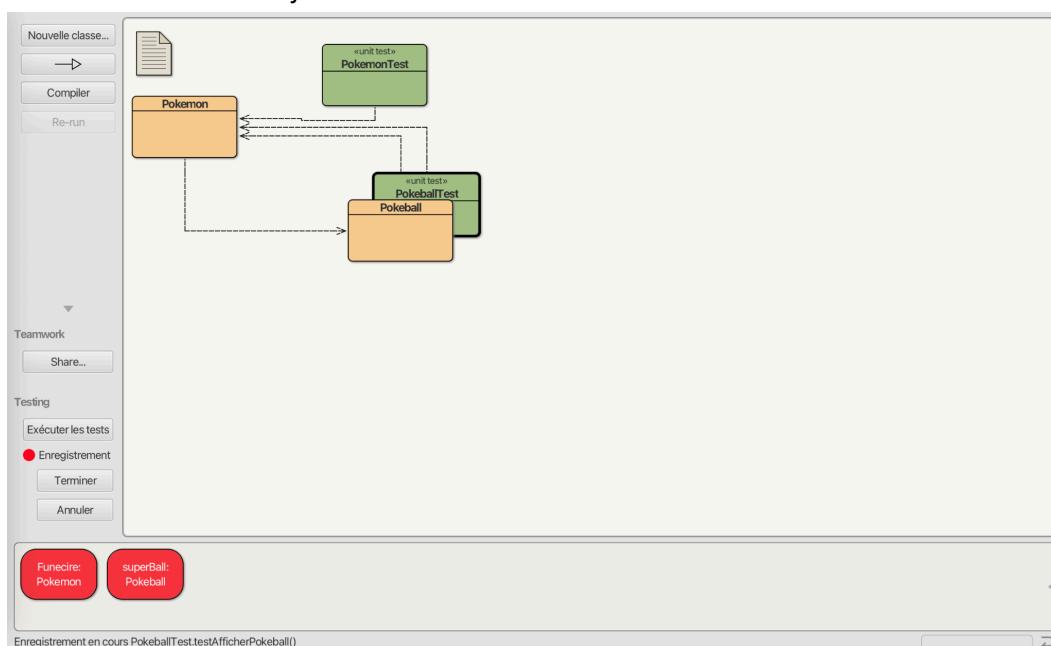
Etape 13 : Test en utilisant la fixture :

Maintenant, on va créer un test dans cet environnement que l'on a créé.

Pour cela comme dans la première partie, on va cliquer droit sur la classe test et cliquer sur "Enregistrer une méthode de test".



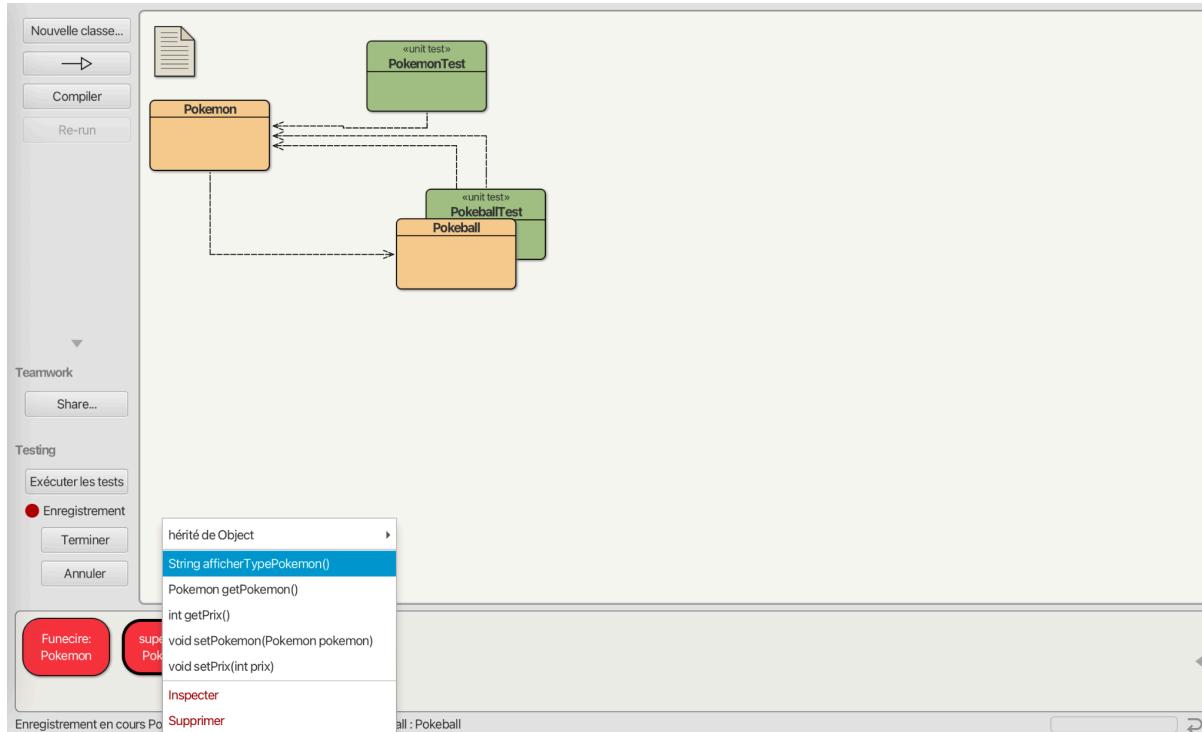
Une fois ce bouton cliqué, comme on est dans l'environnement test, tu remarqueras que les objets que l'on avait instancié et qui avait précédemment disparu, sont de nouveau présents dans la barre d'objets en bas.



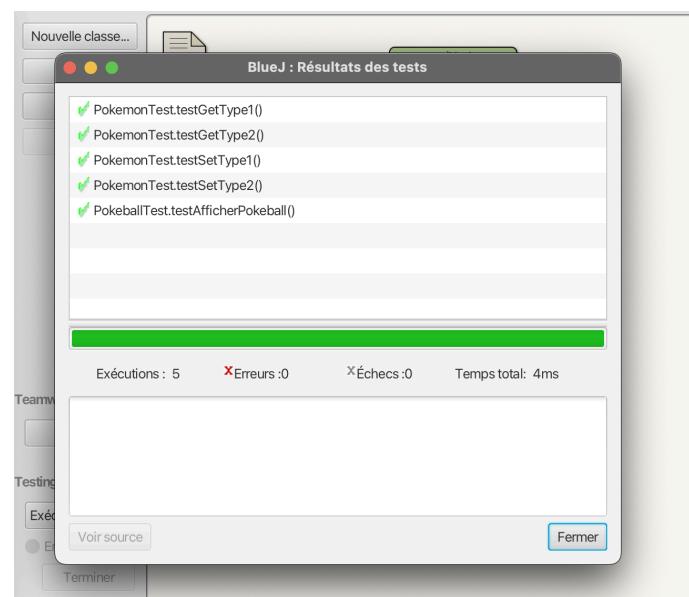


Maintenant, comme dans la première partie, on clique sur la fonction à tester, ici `afficherTypePokemon()`, on a un nouvel écran où préciser le résultat attendu, ici comme c'est `Funecire` (de type Feu et Spectre) qui est dans la Pokéball on attend comme résultat la phrase "De types Feu et Spectre".

Attention à ne pas oublier les guillemets et le point.



Ensuite, on peut terminer l'enregistrement de la fonction, et cliquer sur Exécuter les tests pour vérifier que tout fonctionne bien.



Si cela a bien fonctionné, tu devrais avoir quelque chose comme cet écran !