

Tutoriel : Tests unitaires

Logiciel : BlueJ

Langage : Java

Groupe :

DOUR Marcellino

PARDINI Raphaël

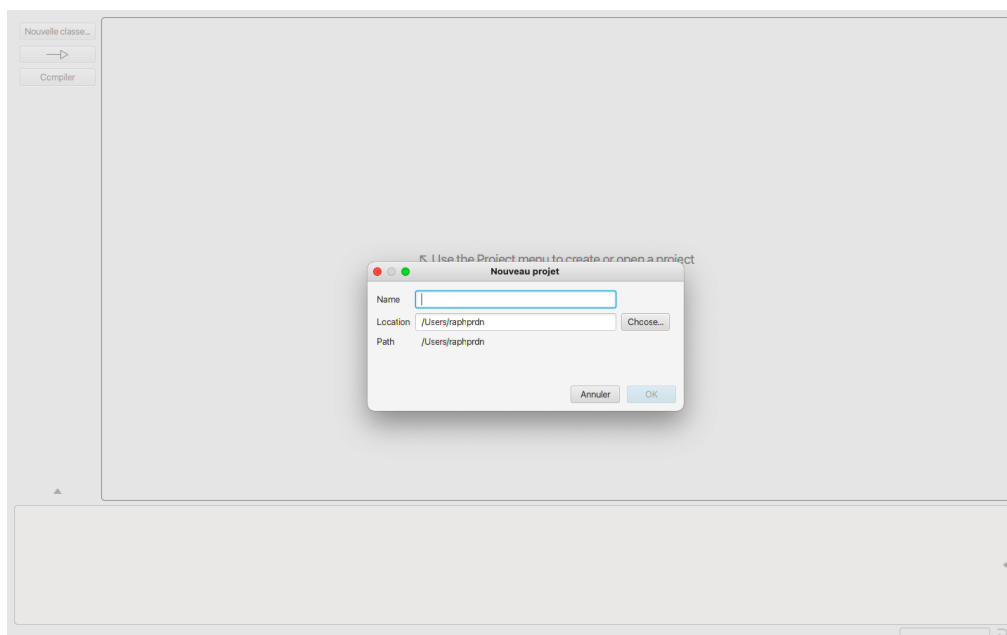
Résumé : Dans ce tutoriel, nous allons ensemble apprendre à comprendre les enjeux et la mise en place des tests unitaires dans le monde de la programmation. Dans ce contexte, nous allons utiliser l'outil BlueJ afin de se concentrer sur l'aspect conception de la programmation JAVA.

C'est parti !

Le GRAND PRIX de F1 à Melbourne (Australie) se tiendra le 10/04 prochain, mais, malheur ... notre application de réservation des écuries est tombée en panne. Nous avons donc besoin de reprendre le projet au plus vite ...

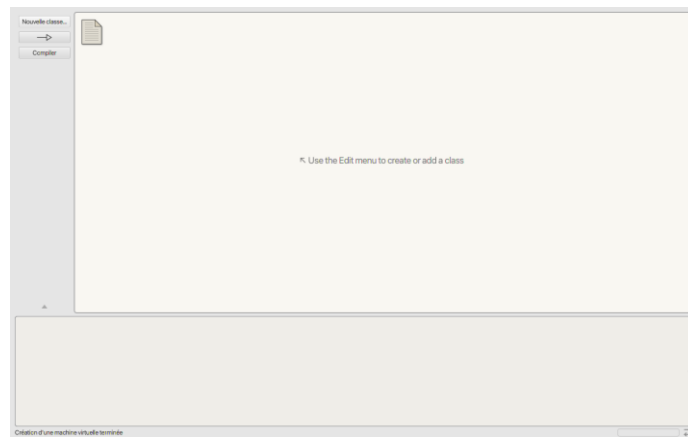
1. Créer un nouveau projet

Cliquer sur nouveau Projet.

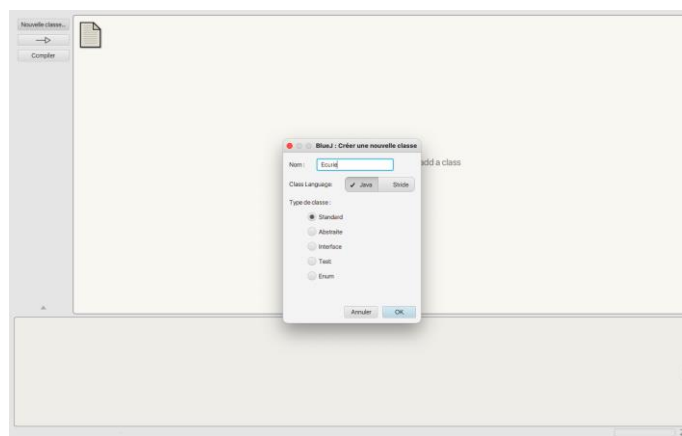


2. Créez une première classe

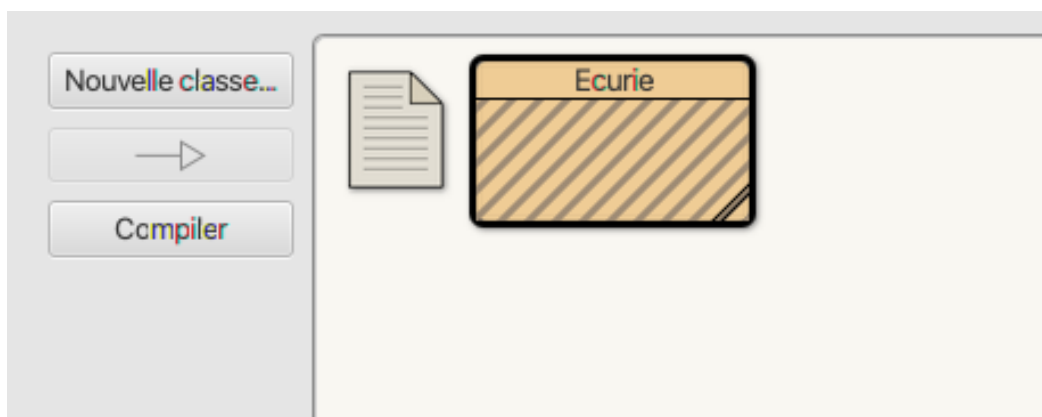
Cliquer sur nouvelle classe



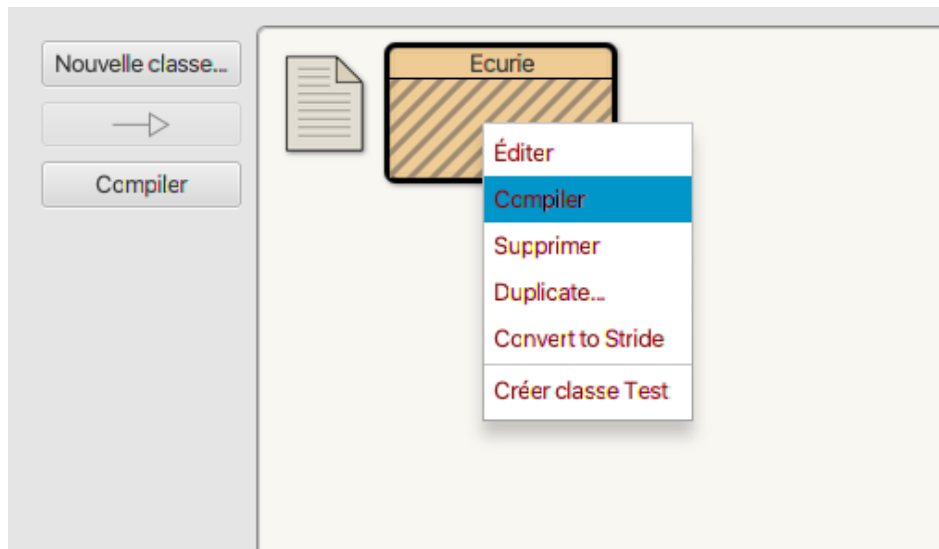
Rentrer le nom de la nouvelle classe et cliquer sur OK. Nous le nommerons écurie.



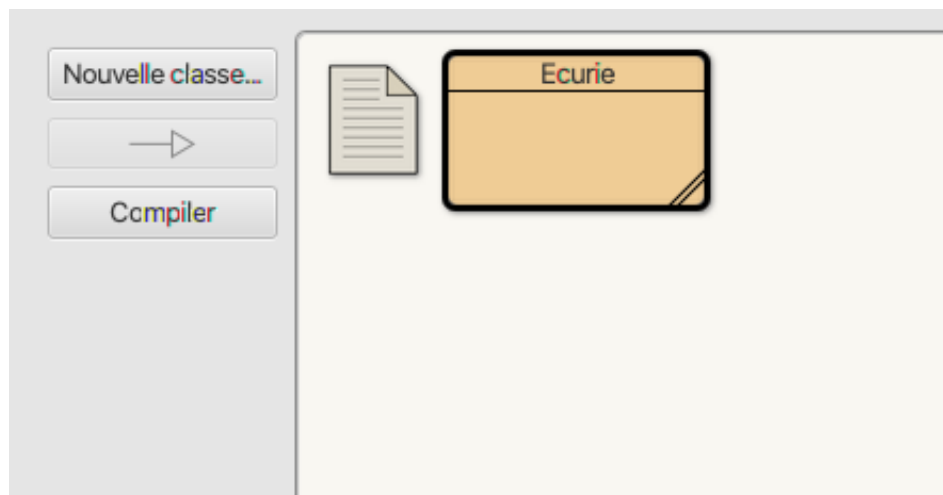
Notre objet écurie est créé. Ceci devrait apparaître :



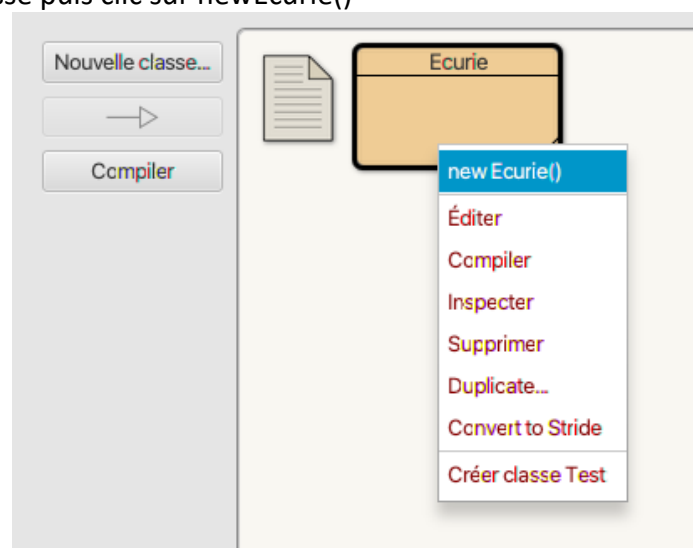
3. Compilez la classe et instanciez-là
Clic droit + compiler



Ceci devrait apparaitre :

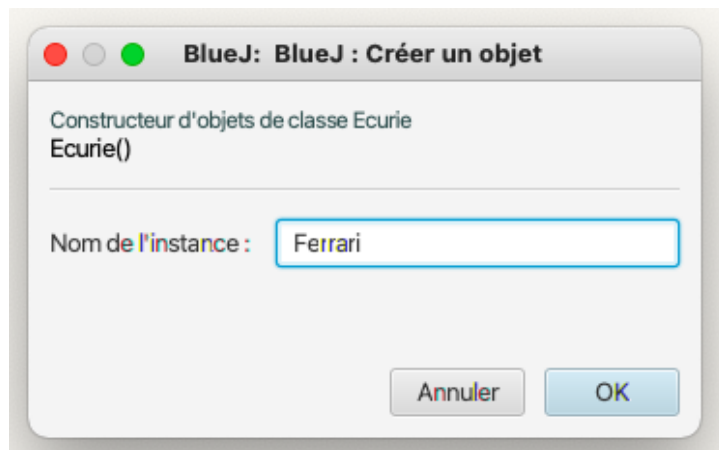


Clic droit sur la classe puis clic sur newEcurie()

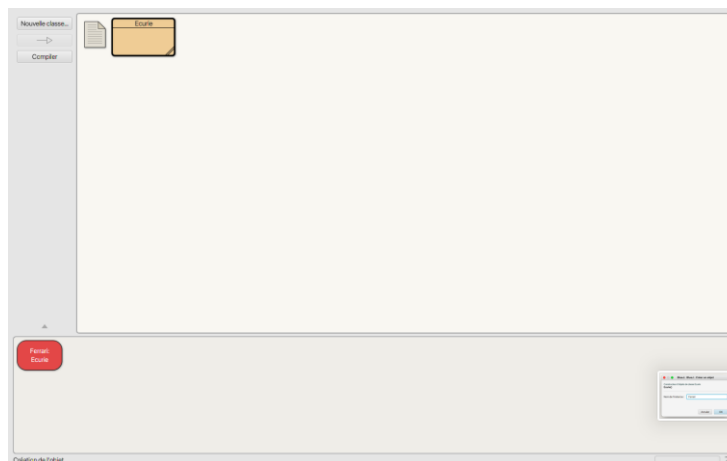


Rentrer le nom puis OK.

Nous avons reçu la confirmation de Ferrari. Enregistrons-là !



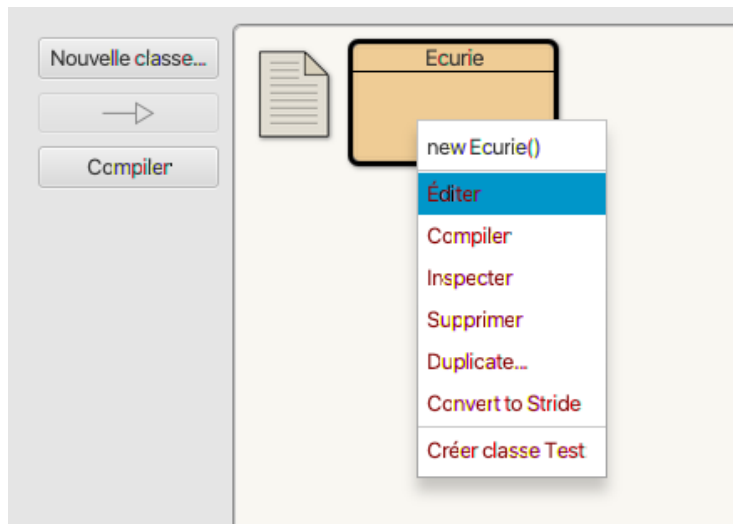
Elle va apparaître en bas.



4. Ajoutez 2 attributs, accesseurs et méthode qui les manipulent et instanciez de nouveau.

Nous avons besoin de connaître le nombre de victoire et le nom de ce dernier dans notre programme. Ajoutons donc les attributs associés ainsi qu'une fonction qui permet de topé les victoire (incréméntation) de ce dernier.

Maintenant, éditons la classe :



On rajoute deux attributs ainsi que les collecteurs et une méthode qui permet d'ajouter une nouvelle victoire.

```

Ecurie X
Compiler Défaire Couper Copier Coller Chercher Fermer Implémentation ▼

* @version (un numéro de version ou une date)
*/
public class Ecurie
{
    // variables d'instance - remplacez l'exemple qui suit par le vôtre
    private int victoire;
    private String nom;

    /**
     * Constructeur d'objets de classe Ecurie
     */
    public Ecurie()
    {
        // initialisation des variables d'instance
        victoire = 0;
        nom = null;
    }

    public void setVictoire(int vict)
    {
        victoire = vict;
    }

    public int getVictoire()
    {
        return victoire;
    }

    public void setNom(String n)
    {
        nom = n;
    }

    public String getNom()
    {
        return nom;
    }
}

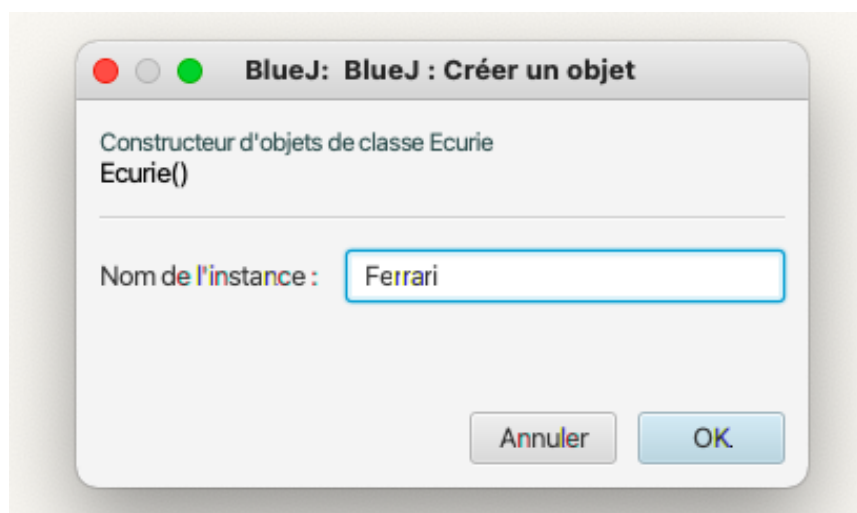
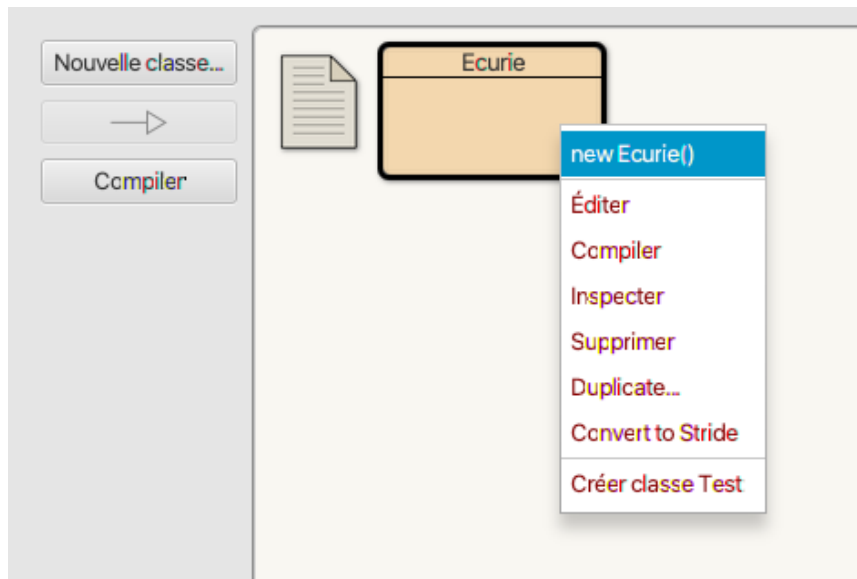
public void newVictoire()
{
    victoire +=1;
}

```

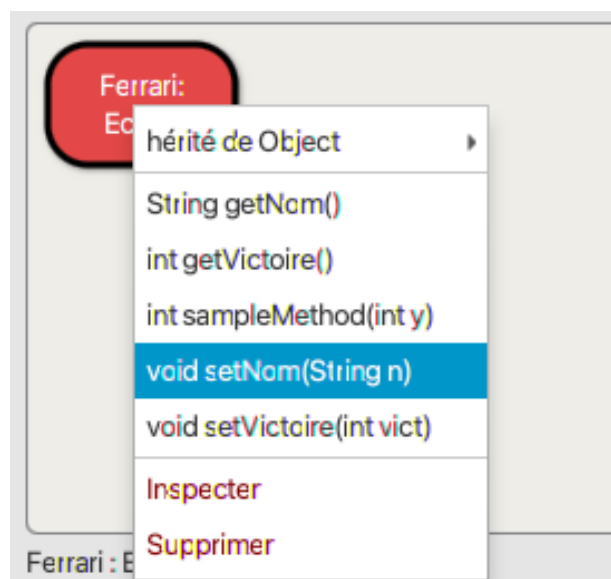
On compile la classe et on s'assure qu'il n'y a pas d'erreur de syntaxe

Classe compilée - pas d'erreur de syntaxe

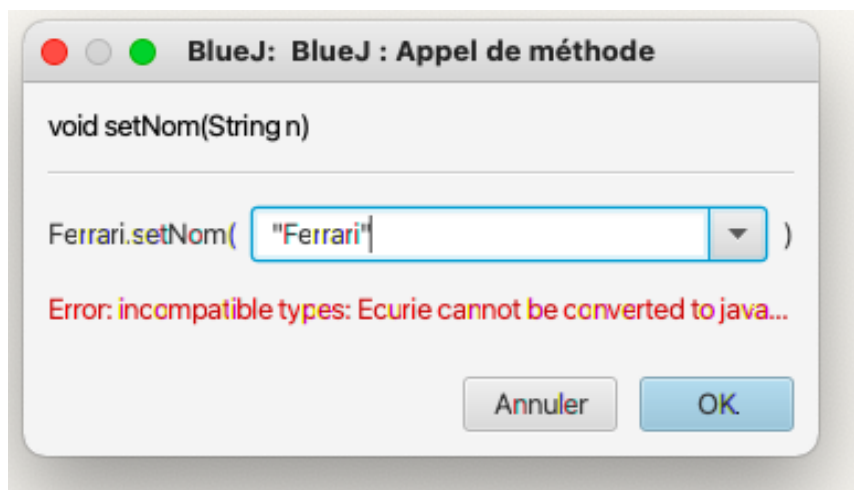
Notre instance a été supprimé, on l'a recréé donc de la même manière que la première fois :



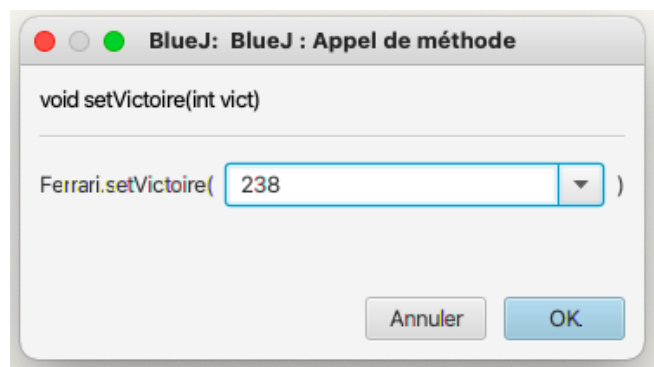
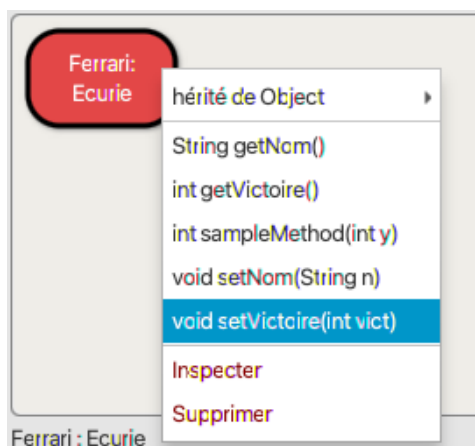
Lorsqu'on fait un clic droit, on a accès aux différentes fonctions créées. Cliquer sur setNom :



Rentrons le nouveau nom et validons.

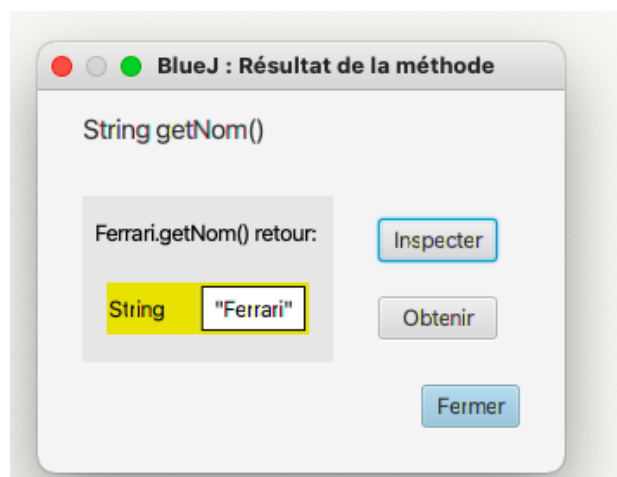
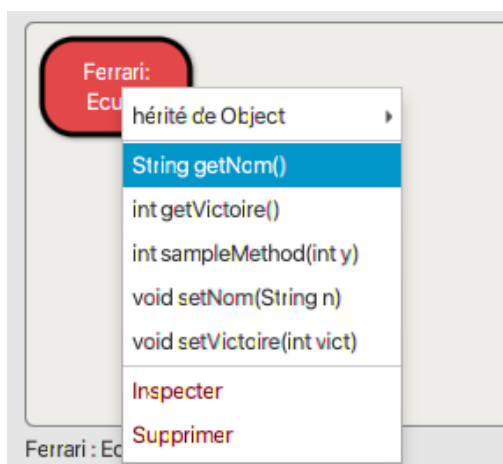


De même pour la victoire :

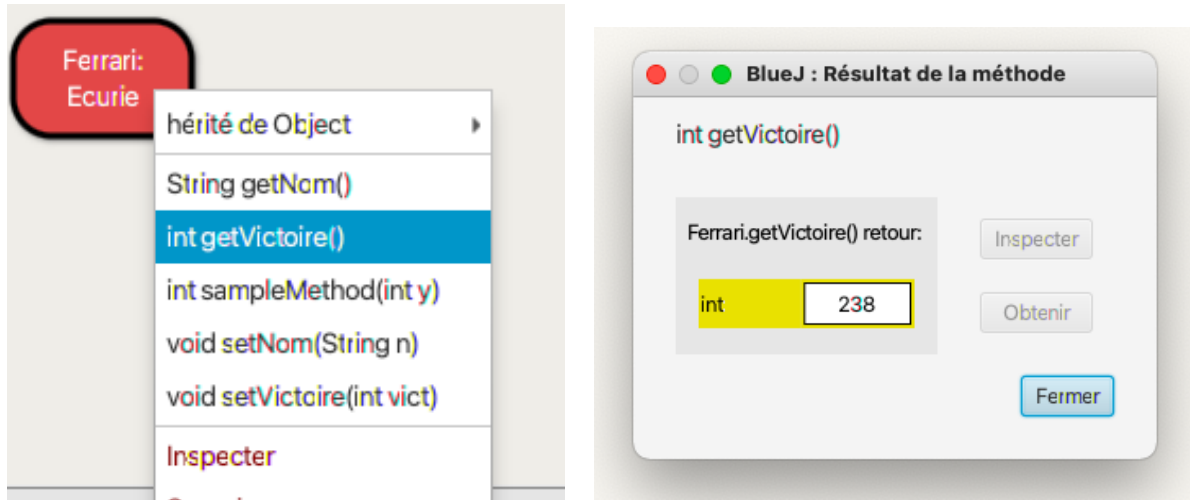


Maintenant que le tout est fait, vérifions si les données sont bien accessibles...

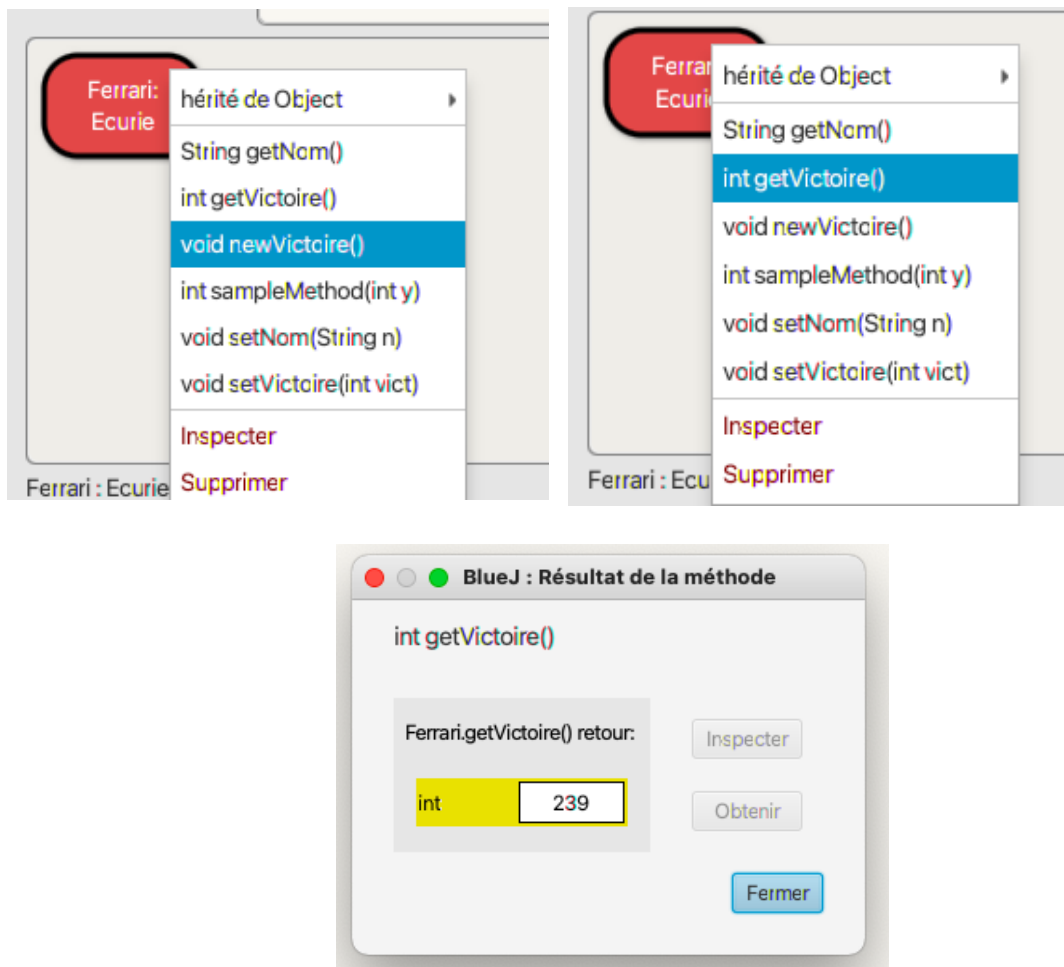
Cliquer sur getNom pour vérifier que nos modifications ont été prises en compte :



De même avec victoire :



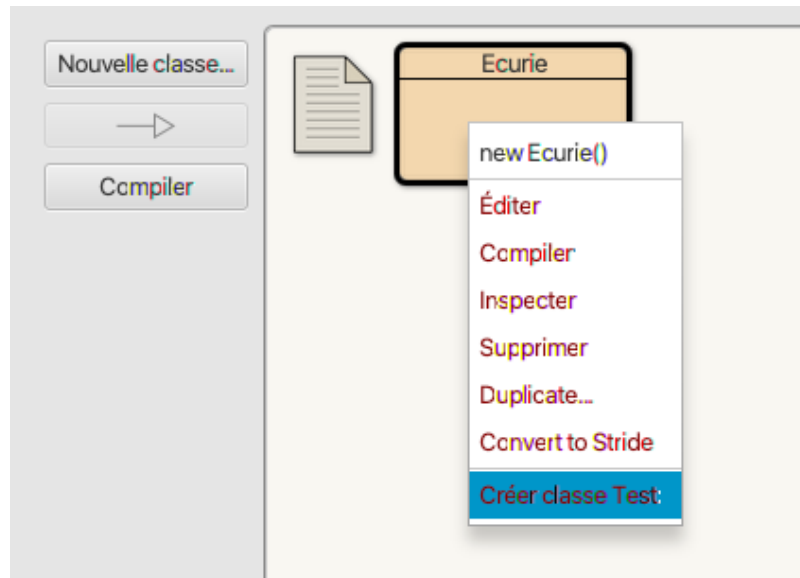
Puis, cliquer sur la methode newVictoire puis getVictoire afin de vérifier que le nombre de victoire a bien été incrémenté de 1 :



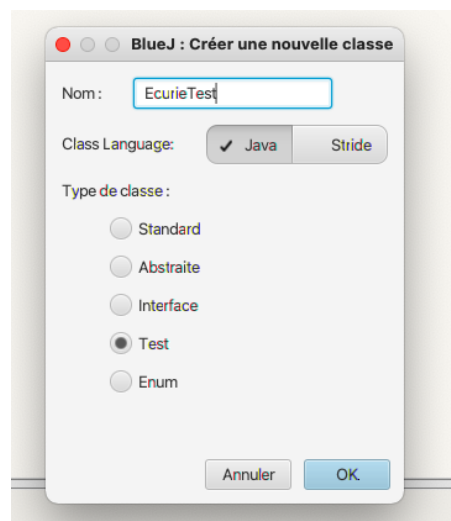
5. Testez unitairement votre classe

Maintenant que nous avons notre classe. Implémentons un test unitaire pour garantir de son fonctionnement. Vérifions ensemble si le nombre de victoire s'incrémente bien ...

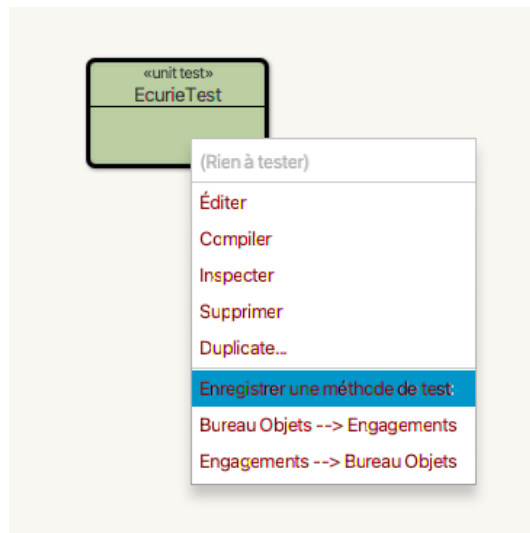
Clic droit sur la classe, et cliquer sur créer classe Test :



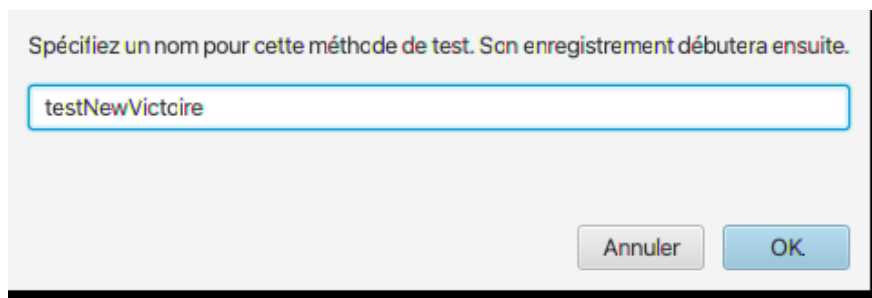
La nommer et cocher « Test » :



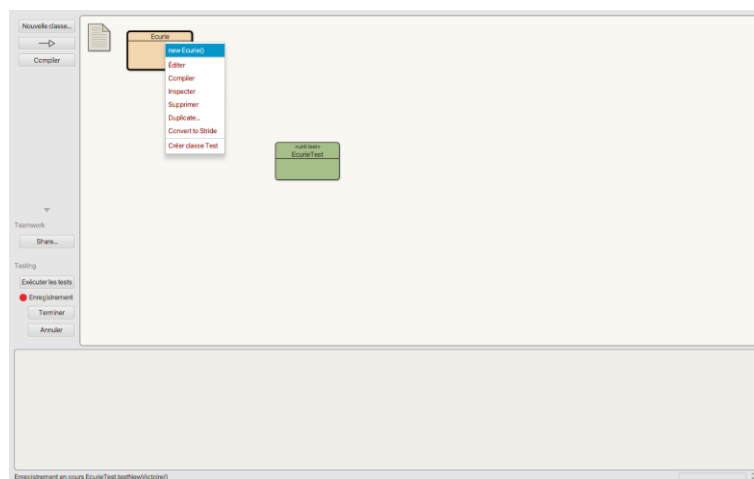
Cliquer sur la classe Test, puis sur enregistrer une méthode de test :

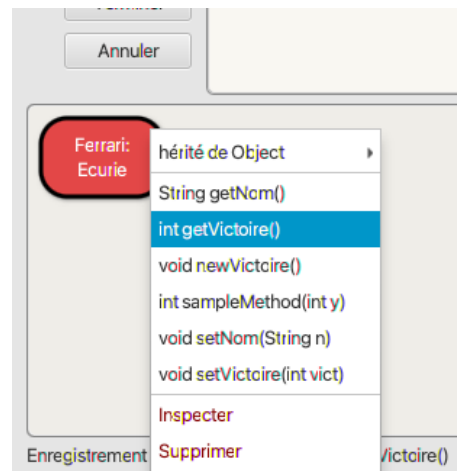
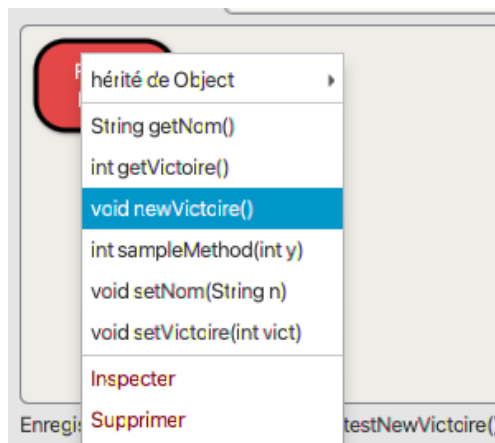
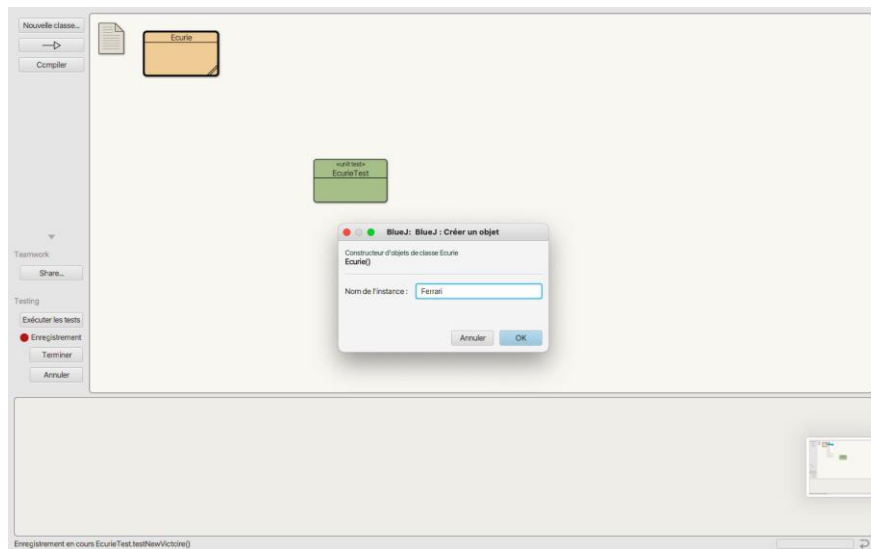


Nommer la méthode puis valider

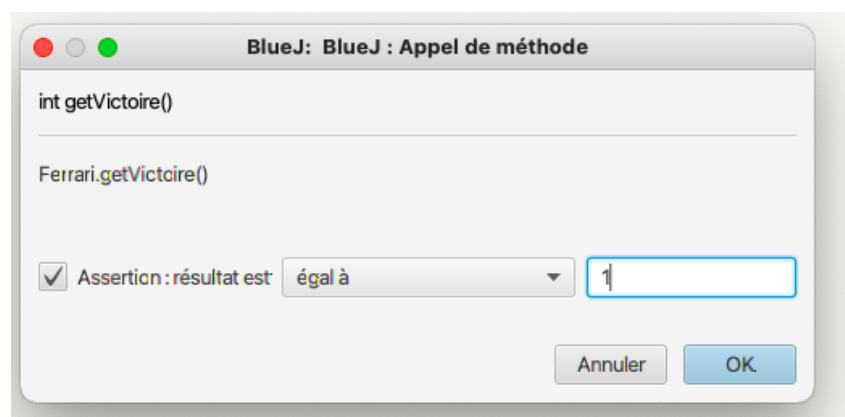


Une fois le bouton OK cliquer, chaque action est enregistrée et sera répété lors du test. Suivre les étapes ci-dessous (manipulation déjà effectué plus haut) :

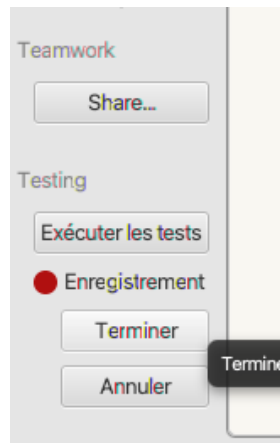




Ici, on rentre la valeur attendue du test :

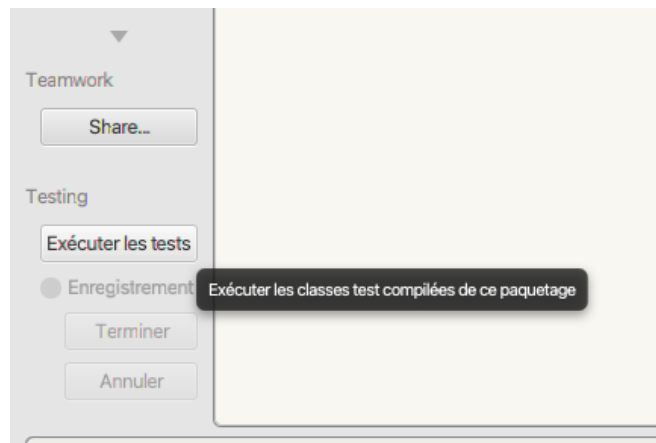


Puis, on termine l'enregistrement.

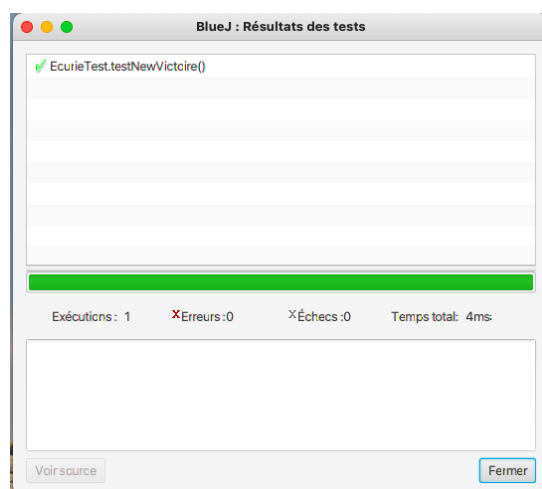


Vérifions si nous avons bien enregistré la victoire de Charles Leclerc du grand prix de Bahreïn.

On peut cliquer sur Exécuter les tests :



La barre verte nous indique que les tests ont été validés :

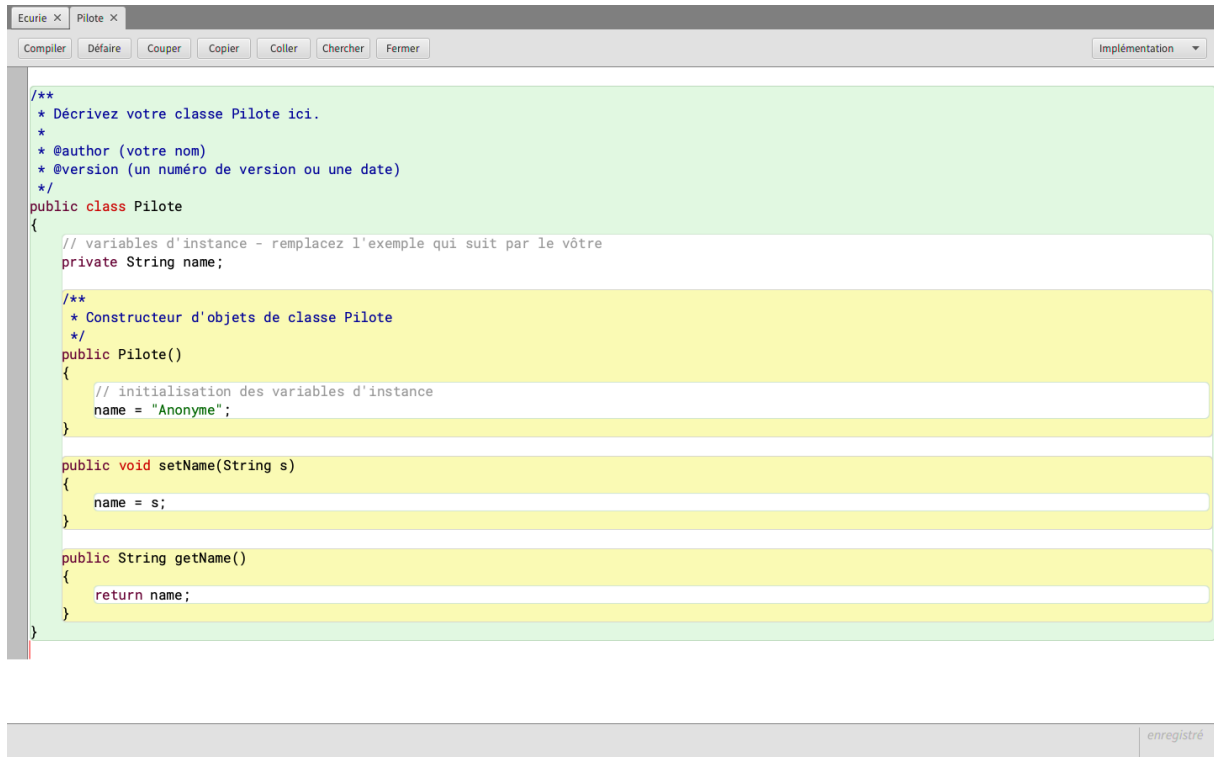


Charles peut donc aller se reposer tranquillement. Nous avons bien enregistré sa victoire !!

6. Ajoutez une seconde classe

Nous apprenons que Carlos Sainz a bien résigné chez notre leader rouge. Nous n'allons pas nous priver de cette information, et enregistrons là !

Nous allons maintenant créer une classe Pilote, contenant un attribut « name » de type String ainsi que des collecteurs. (Si besoin, revoir les captures d'écrans ci-dessus).



```
/**
 * Décrivez votre classe Pilote ici.
 *
 * @author (votre nom)
 * @version (un numéro de version ou une date)
 */
public class Pilote
{
    // variables d'instance - remplacez l'exemple qui suit par le vôtre
    private String name;

    /**
     * Constructeur d'objets de classe Pilote
     */
    public Pilote()
    {
        // initialisation des variables d'instance
        name = "Anonyme";
    }

    public void setName(String s)
    {
        name = s;
    }

    public String getName()
    {
        return name;
    }
}
```

enregistré

Puis, éditer la classe Ecurie afin de créer un lien avec la classe Pilote.

Pour cela, rajouter un attribut nommé pilote et de type Pilote. Mettre à jour le constructeur et les collecteurs en conséquence.

```
/**
 *
 */
public class Ecurie
{
    // variables d'instance - remplacez l'exemple qui suit par le vôtre
    private int victoire;
    private String nom;
    private Pilote pilote;

    /**
     * Constructeur d'objets de classe Ecurie
     */
    public Ecurie()
    {
        // initialisation des variables d'instance
        victoire = 0;
        nom = null;
        pilote = new Pilote();
    }

    public Pilote getPilote()
    {
        return pilote;
    }

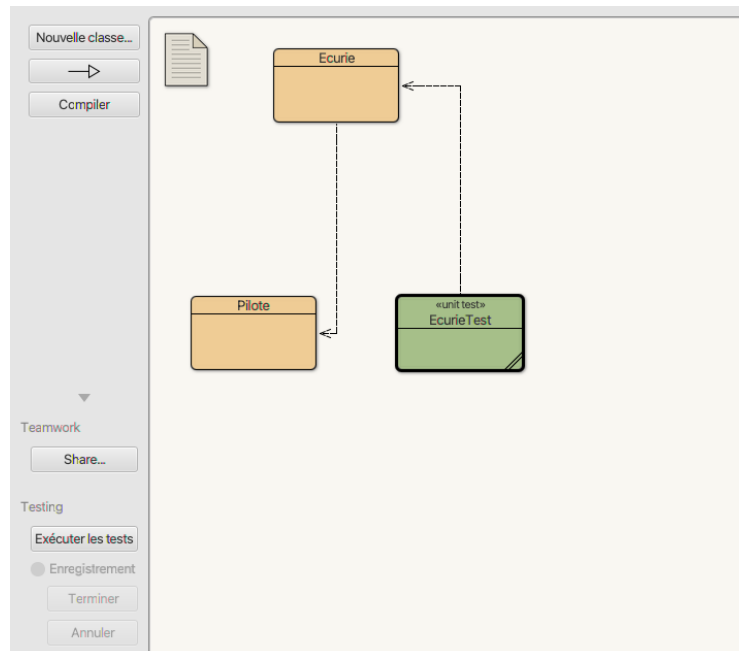
    public void setPilote(Pilote p)
    {
        pilote = p;
    }
}
```

```

    }
    public String getNamePilote()
    {
        return pilote.getName();
    }
}

```

Compiler à nouveau les différentes classes.

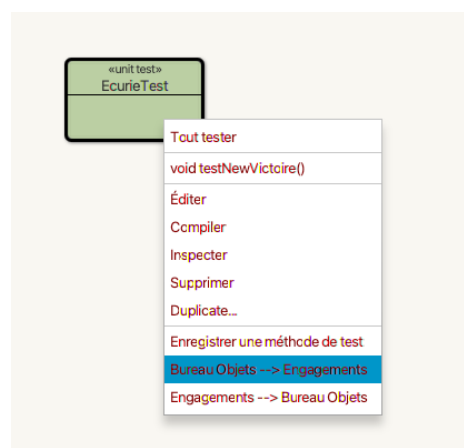


7. Sauvegardez les classes dans la fixture d'une classe de test.

Nous aimerions tester la possibilité de récupérer le nom de Carlos depuis l'objet Ecurie. Pour se faire, nous pourrions à chaque nouveauté appelé le standadiste du garage Ferrari pour e avoir l'information. Cependant, ne serait-il pas plus intéressant de garder cette information automatiquement à chaque test ? Nous allons donc installer notre setup de test !

Pour se faire nous allons instancier les classes comme vu précédemment. Nous enregistrons Ferrari et Carlos, sans oublier de set le pilote Carlos à son écurie Ferrari.

Cliquer sur Bureau Objets → Engagements pour initialiser le setup.



Nous pouvons donc observer que le setup de la classe test a été généré automatiquement.



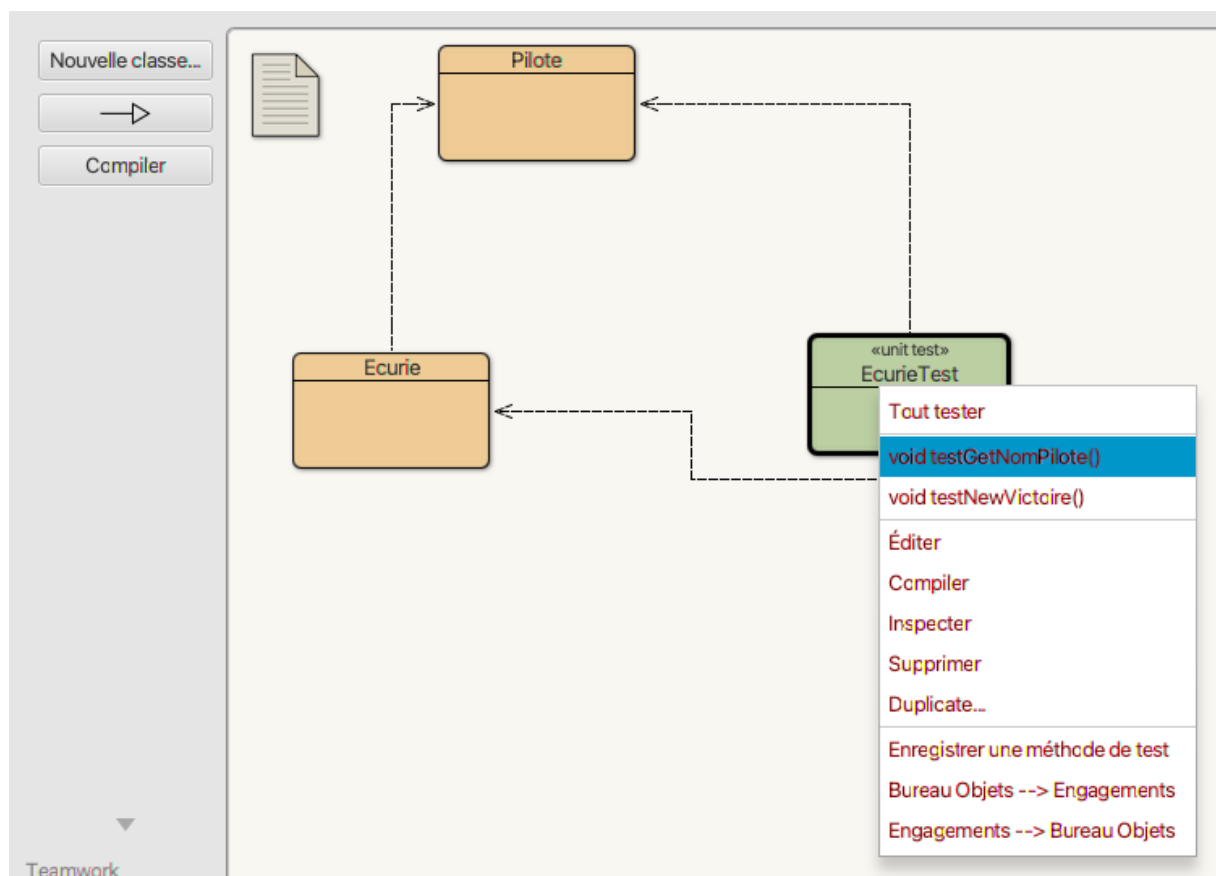
```
// Vérifiez les variables d'instance nécessaires à vos engagements.
// Vous pouvez également les saisir automatiquement du présentoir
// à l'aide du menu contextuel "Présentoir --> Engagements".
// Notez cependant que ce dernier ne peut saisir les objets primitifs
// du présentoir (les objets sans constructeur, comme int, float, etc.).

/**
 * Constructeur de la classe-test EcurieTest
 */
public EcurieTest()
{
}

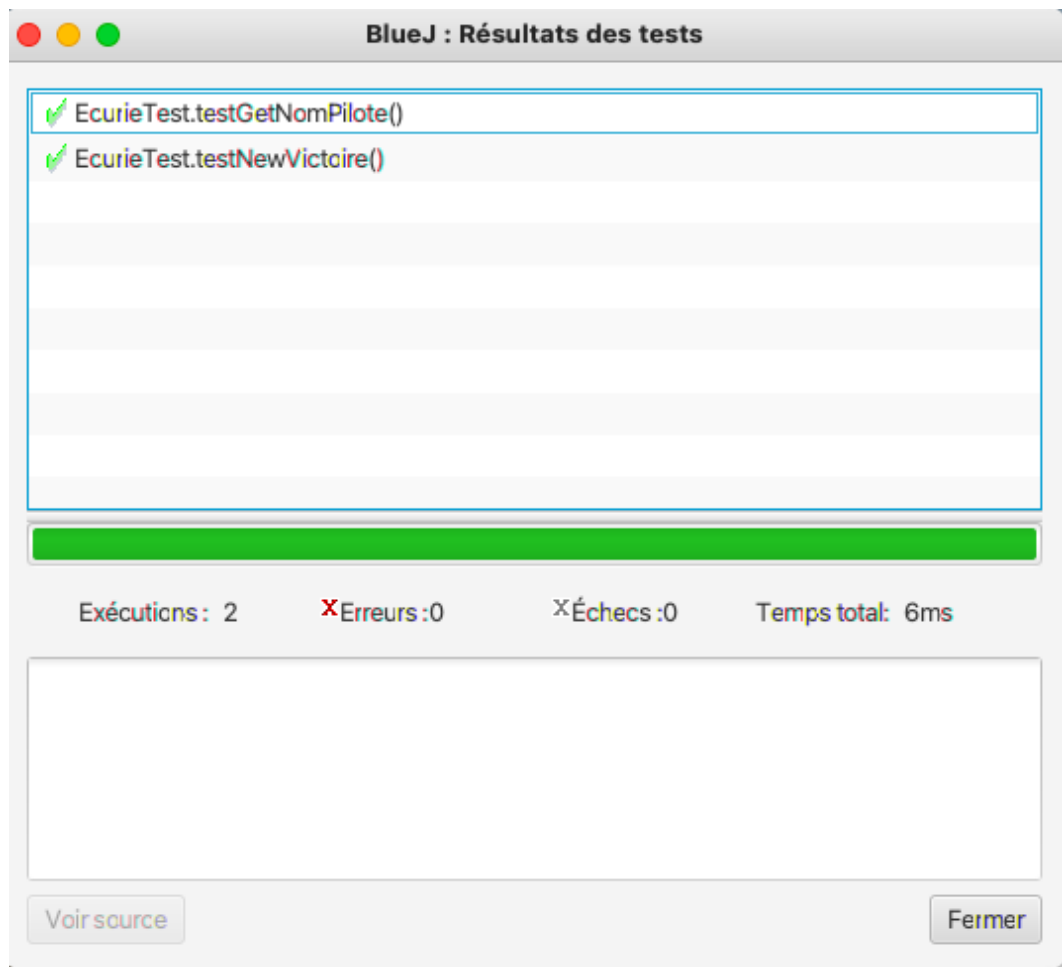
/**
 * Met en place les engagements.
 *
 * Méthode appelée avant chaque appel de méthode de test.
 */
@BeforeEach
public void setUp() // throws java.lang.Exception
{
    pilote1 = new Pilote();
    ecurie1 = new Ecurie();
    pilote1.setName("Carlos");
    ecurie1.setNom("Ferrari");
    ecurie1.setPilote(pilote1);
}

/**
 * Supprime les engagements
 *
 * Méthode appelée après chaque appel de méthode de test.
 */
@AfterEach
```

Testons la classe GetNomPilote.



Vérifions ce que peuvent bien dire les tests unitaires ...



Tous les feux sont aux verts ! A vos marques, prêts