DUBOSCQ Maxime & LONGHI Nicolas
M2 MIAGE SITN ALT

Tutoriel BlueJ

Bonjour et bienvenue dans ce tutoriel ! 😃

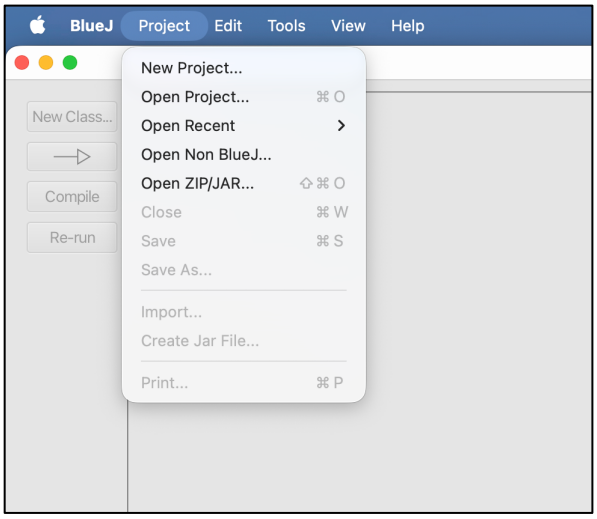
Dans celui-ci, nous allons apprendre à utiliser BlueJ, à coder en Java et à utiliser JUnit.

Voici les étapes :

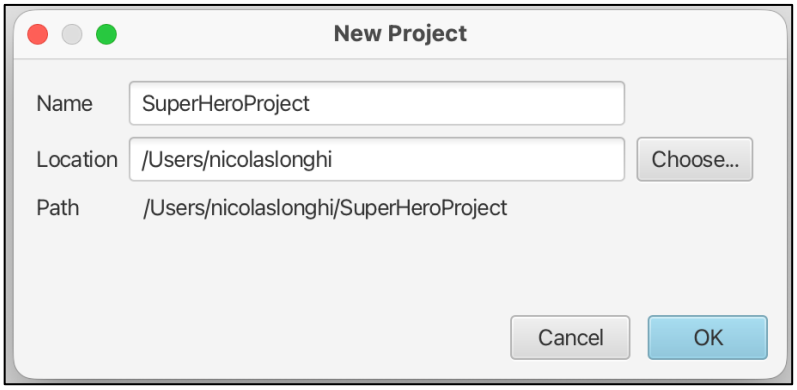
1. Il faut déjà télécharger le logiciel BlueJ via le lien suivant : http://www.bluej.org/. Veillez à prendre la version correspondante à votre système d'exploitation (Windows, MacOS ou encore Linux).

2. Suite à cela, il faut installer le logiciel en suivant les étapes.

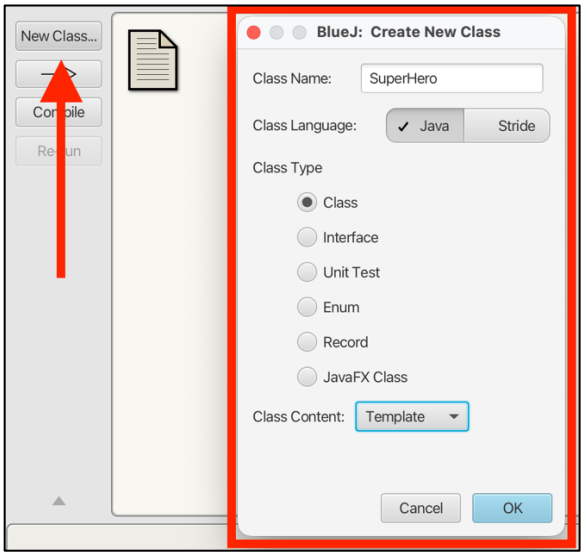
3. Créez un projet via le bouton Project→New project



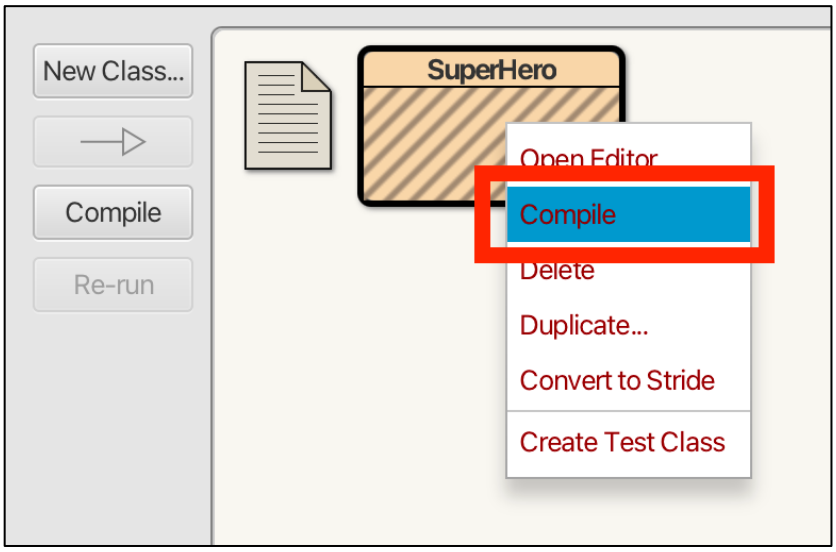
Puis, entrez le nom du projet : ***SuperHerosProject***



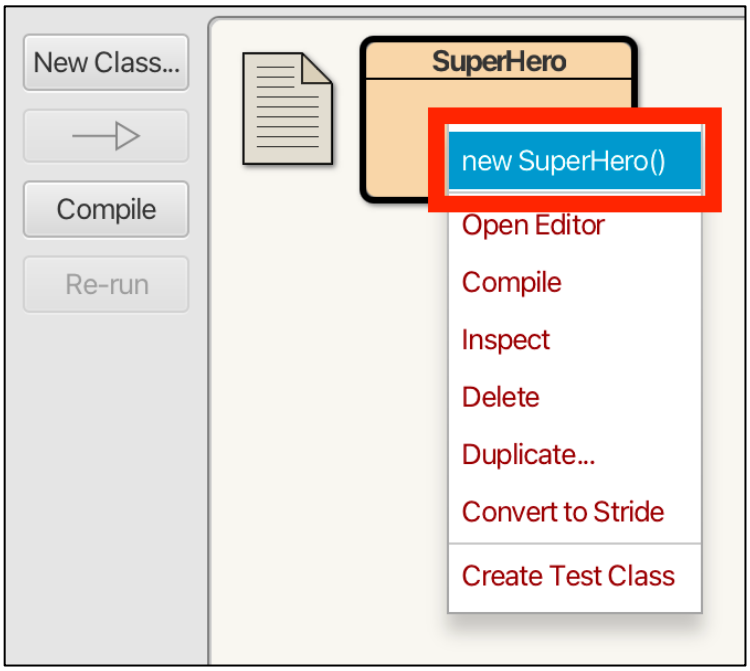
4. Il faut maintenant créer la classe SuperHeros en appuyant sur le bouton **New Class**. Ensuite, entrez le nom de la classe et appuyez sur **OK**.



5. Veuillez compiler la classe via le clic droit puis **Compile.** La compilation permet d'enregistrer et de générer le code afin d'exécuter notre logiciel.



6. Veuillez instancier la classe via le clic droit **new SuperHeros()** et entrez le nom **SpiderMan**, un carré rouge devrait apparaître en bas. La classe est le moule et l'instance est le gâteau. Nous allons créer un gâteau SpiderMan à partir du moule SuperHeros.



7. En faisant clic droit sur le carré orange **SuperHeros,** et appuyez sur **Open Editor**. Une fenêtre s'ouvrira alors avec un code par défaut. Il faut modifier le code pour :

a. **Ajouter 2 attributs** : un nom et un pouvoir. Pour ce faire, il faut remplacer *private int x;* par *private String nom;* et *private String pouvoir;* Ensuite, il faut modifier le constructeur qui commence par *public SuperHeros*. Voici le à quoi elle doit ressembler :

```
public SuperHero(String unNom, String unPouvoir)
{
    nom = unNom;
    pouvoir = unPouvoir;

}
```

b. **Créer les getters** : Ce sont des fonctions qui permettent de retourner les valeurs des attributs. Voici les 2 fonctions à créer qui correspondent aux 2 attributs :

```
public String getNom()
{
    return this.nom;
}
```

```
public String getPouvoir()
{
    return this.pouvoir;
}
```

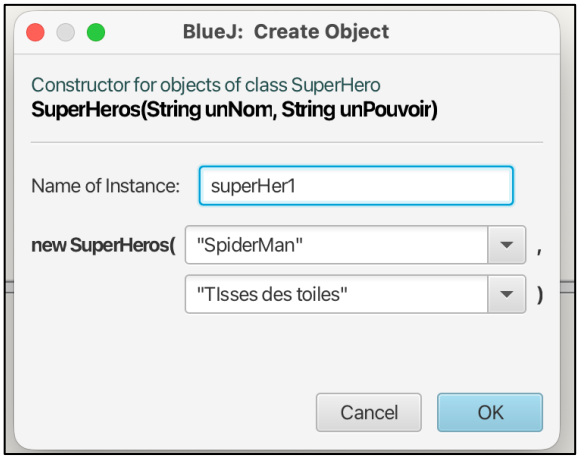
c. **Créer les setters** : Ce sont des fonctions qui permettent de modifier les valeurs des attributs. Voici les 2 fonctions à créer qui correspondent aux 2 attributs :

```
public void setNom(String unNom)
{
    this.nom = unNom;
}
```

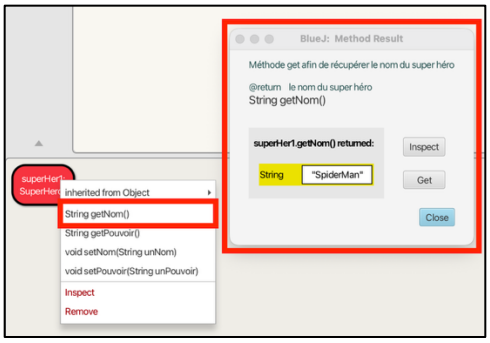
```
public void setPouvoir(String unPouvoir)
{
    this.pouvoir = unPouvoir;
}
```

Enfin, il faut cliquer sur le bouton **Compile** pour enregistrer les modifications.

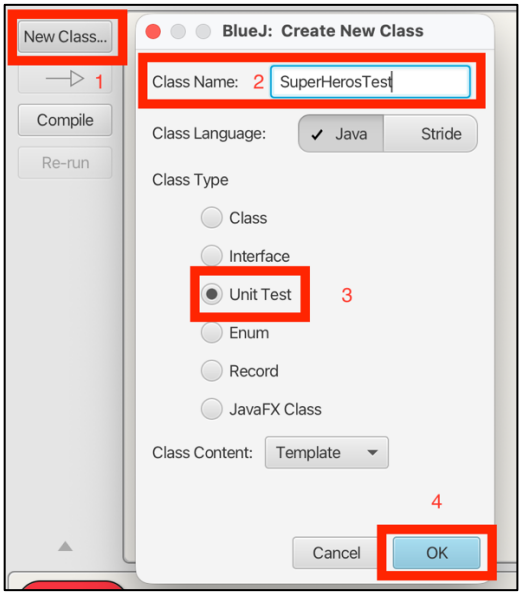
8. Maintenant, il faut réinstancier la classe. Puisqu'elle a été modifiée, une fenêtre apparaitra afin de nous demander le nom et le super pouvoir du super héros en cours de création. Veuillez rentrer un nom et un pouvoir de votre choix entre guillemets (car il s'agit de chaîne de caractère) puis cliquez sur le bouton **OK**.



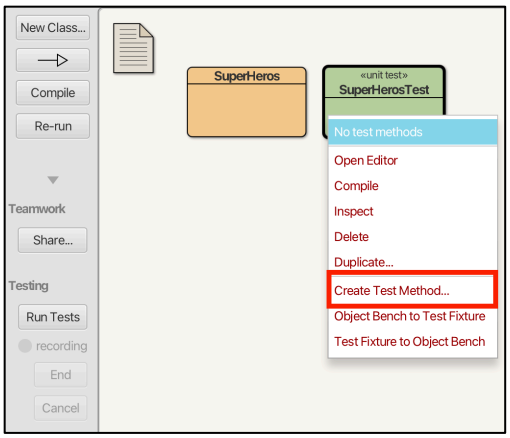
Pour tester nos modifications de la question 7, nous pouvons cliquer sur le carré rouge en bas à gauche et cliquer sur getNom(), une fenêtre indiquant le nom rentré précédemment est censé être affiché. Nous pouvons aussi tester les setters pour modifier les valeurs.



9. Nous allons maintenant créer une classe afin de tester la classe **SuperHeros**. Pour cela, il faut créer une nouvelle classe que nous appelerons *SuperHerosTest*, de type Unit test.



Comme vous pouvez le remarquer, un nouvel élément vert a été créé. Celui-ci est donc la classe de test. Ensuite, faites clic droit sur cette nouvelle classe pour appuyer sur **Create Test Method**.



Veuillez entrer le nom de votre test. Après avoir cliqué sur **OK**, le logiciel va enregistrer tout ce que nous faisons (comme une macro dans Excel) afin de pouvoir relancer toutes ces actions automatiquement pour tester tout le comportement de la classe. Cela nous évite de ressaisir SpiderMan à la main à chaque fois qu'on modifie le code.

Vous remarquerez que le bouton clignote en rouge à côté de recording sur la partie gauche. Nous allons donc instancier un nouvel objet (cf étape précédente). Ensuite, nous appuierons sur le bouton **End** pour terminer l'enregistrement.

Nous pouvons alors relancer le tests (et d'autres si créé) à tout moment en faisant un clic droit sur la classe de test et en appuyant sur **Test All**.

10. Un Super Héros n'existe pas dans le vide. Pour lier notre héros à un monde, nous allons créer un objet **Univers** qui servira de conteneur. Pour cela, nous allons créer une nouvelle classe (cf étape 4). Ensuite, nous allons ajouter un attribut dans la classe SuperHeros de type Univers et nous allons créer le setter et le getter associé. Voici le code de la classe Univers :

```
private String nom;

/**
 * Constructor for objects of class Univer
 */
public Univers(String unNom)
{
    this.nom = unNom;
}

/**
 * Méthode set qui modifie le nom de l'univers
 *
 * @param  unNom  le nom de l'univers
 */
public void setNom(String unNom)
{
    this.nom = unNom;
}


/**
 * Méthode get afin de récupérer l'univers du super héro
 *
 * @return    l'univers du super héro
 */
public String getNom()
{
    return this.nom;
}
```

Voici les éléments ajoutés dans la classe SuperHeros :

```
private String nom;
private String pouvoir;
private Univers univers;
```

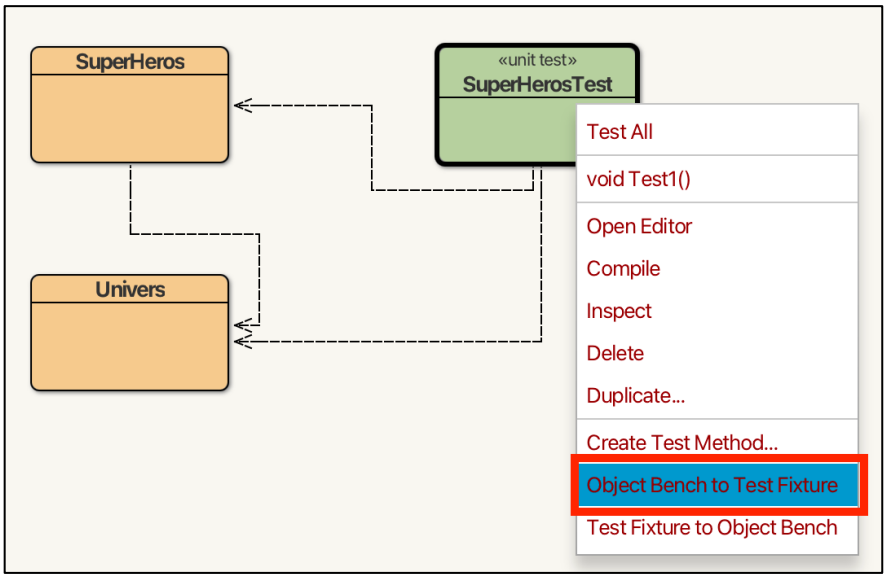
```
public Univers getUnivers()
{
    return this.univers;
}
```

```
public void setUnivers(Univers unUnivers)
{
    this.univers = unUnivers;
}
```

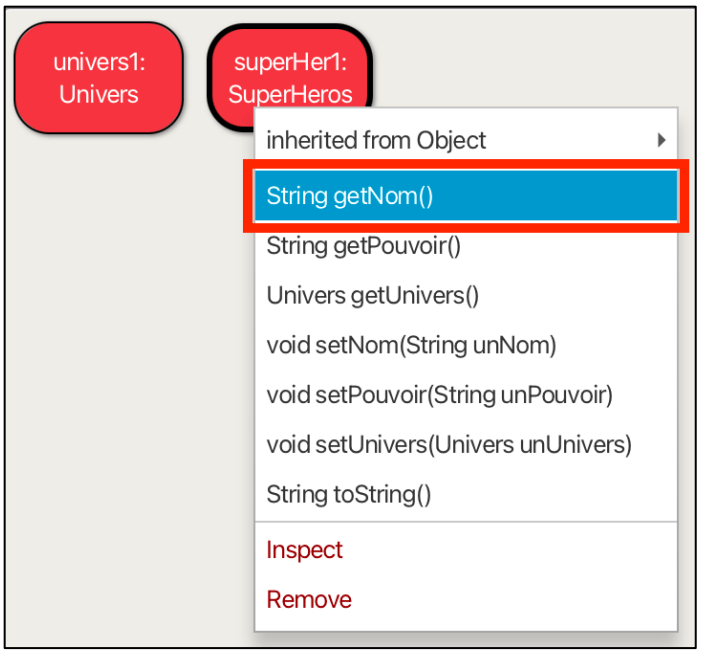
11. Nous allons désormais créer une méthode toString(). Celle-ci permet d'afficher les attributs d'une classe. C'est ce que nous allons faire pour la classe SuperHeros. Voici le code de la méthode :

```
public String toString() {
    return "Je suis " + this.nom + " dans l'univers " + this.univers.getNom() + " et je " + this.pouvoir;
}
```

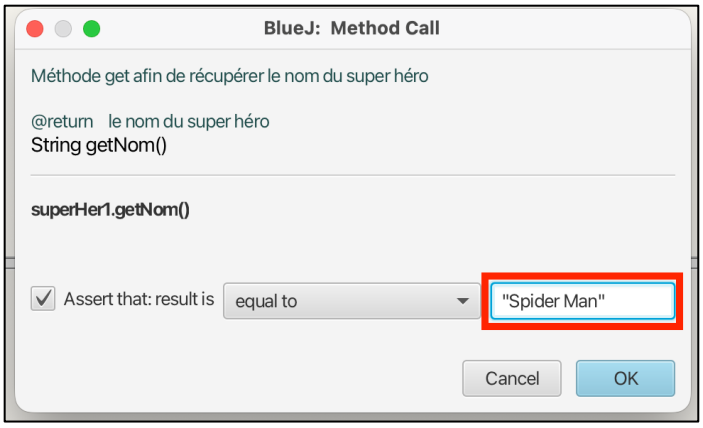
12. Concernant les tests unitaires, nous avons la possibilité d'effectuer plusieurs actions (comme par exemple instancier des objets) avant chaque test. Pour tester, nous allons instancier un objet SuperHeros et Univers (cf étape 6) et ainsi utiliser la fonction *setUnivers()* en effectuant un clic droit sur l'objet SuperHeros créé (le carré rouge en bas) et en cliquant sur la fonction correspondante. Enfin, pour inscrire toutes ces étapes dans la classe de test, nous allons effectuer un clic droit dessus et appuyer sur *Object Brench to Test Fixure*. Les étapes de création en amont seront transformées en code.



13. Enfin, nous allons tester tout cela via JUnit. Pour ce faire, nous allons créer un nouveau test (cf étape 9). Nous remarquons qu'au moment de la création de celui-ci, les objets créés précédemment se réaffichent. Ensuite, pour le test, nous allons faire clic droit sur l'objet SuperHeros (carré rouge en bas) et cliquer sur *getNom()* pour tester cette fonction.

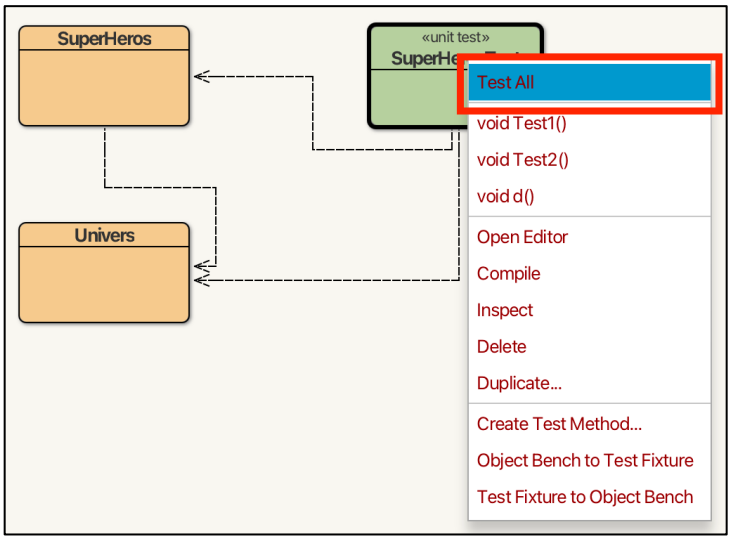


Le résultat attendu est donc le texte « Spider Man », c'est ce que nous allons rentrer dans la fenêtre suivante :



Nous avons la possibilité de modifier l'opérateur en fonction du résultat attendu. Ici, on s'attend à ce que le nom soit égal à Spider Man. Ensuite, la fenêtre suivante nous affiche le résultat de la fonction.
Nous pouvons effectuer d'autres tests ou alors les arrêter en cliquant aussi sur le bouton **End**.

Après cela, nous avons la possibilité de lancer les 2 tests (celui de l'étape 9 et celui-ci) en faisant clic droit sur la classe de test et en appuyant sur **Test All**.



Cela aura pour effet de lancer les tests en entier tout en s'assurant qu'à chaque test la macro de l'étape précédente se réexécute.

Merci pour d'avoir suivi ce tutoriel 👍