DAUPHINE
UNIVERSITÉ PARIS

MAIL – TP Test Unitaire

Ce tutoriel permet de comprendre par un exemple simple et ludique les notions de classes et d'objet. L'exemple utilisé se base sur l'univers de « StarWars © », tout droit détenus par Walt Disney Company.

Imane BOUKA

Hervé QUINIOU

Encadré par :

Michel ZAM

Sommaire

I. Présentation du projet

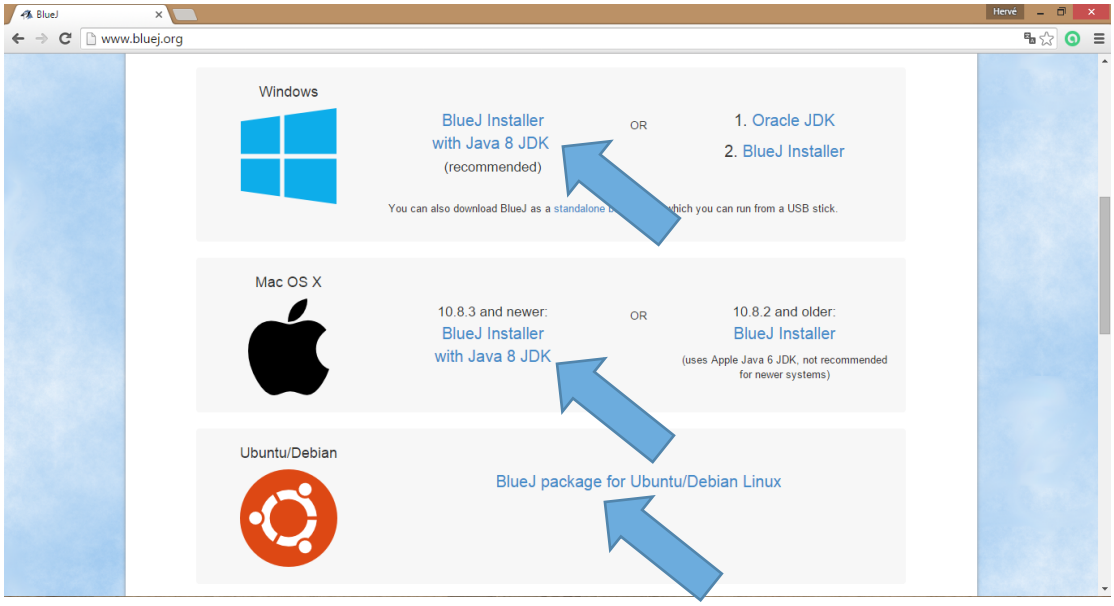
Ce projet entre dans le cadre du cours de méthode agile d'ingénierie logiciel du master MIAGE 2, 2015/2016.

Ce tutoriel permet de comprendre par un exemple simple et ludique les notions de classes et d'objet. L'exemple utilisé se base sur l'univers de « Star Wars » dont les droits sont détenus par Walt Disney Company.
Il est composé de 20 étapes séparées en deux parties. La première se base sur l'utilisation de BlueJ et la deuxième Eclipse et JUnit.

II. Première partie BlueJ

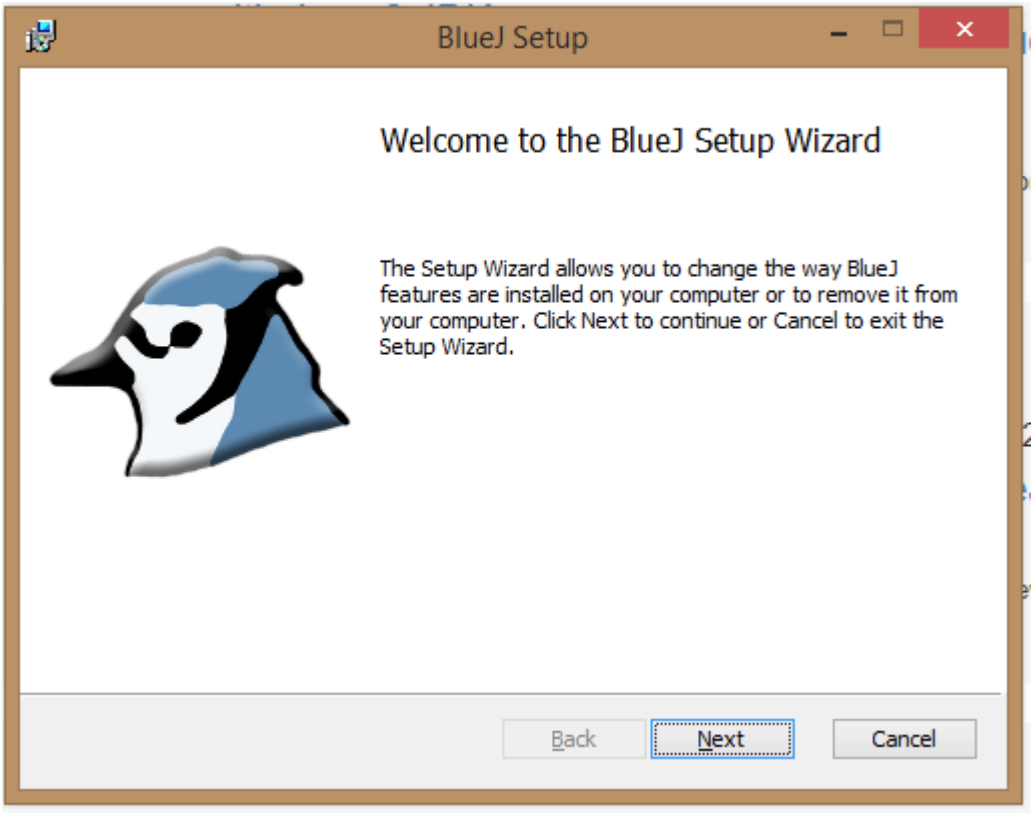
1. Télécharger BlueJ sur votre machine.

Pour cela connectez-vous sur www.bluej.org puis télécharger BlueJ suivant votre système d'exploitation.



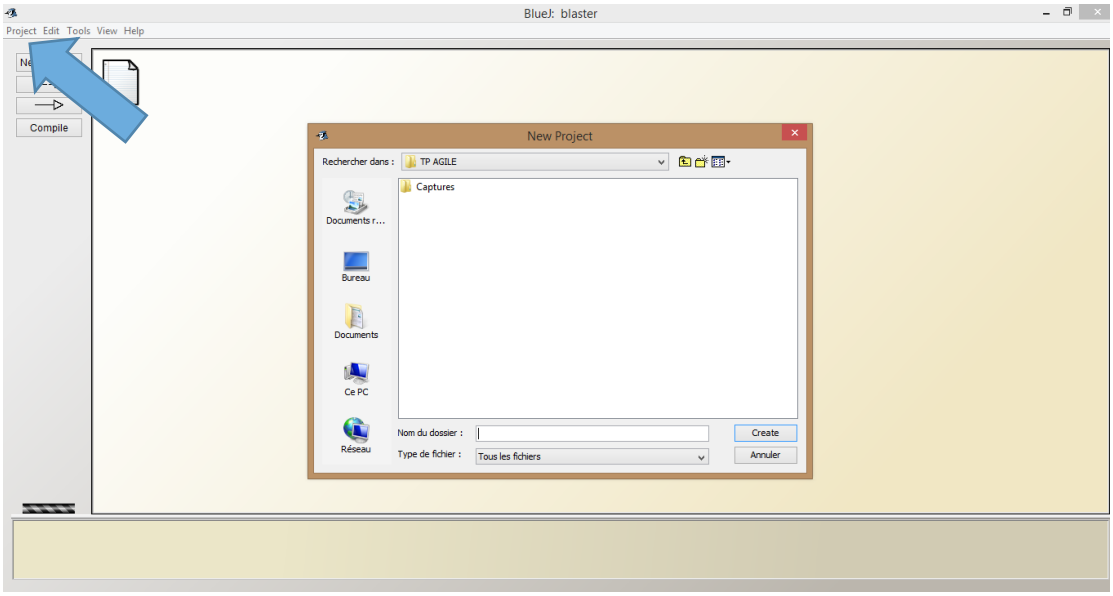
2. **Installez-le sur votre machine.**

Après avoir cliqué et télécharger l'installateur, ouvrez-le puis suivez les instructions.



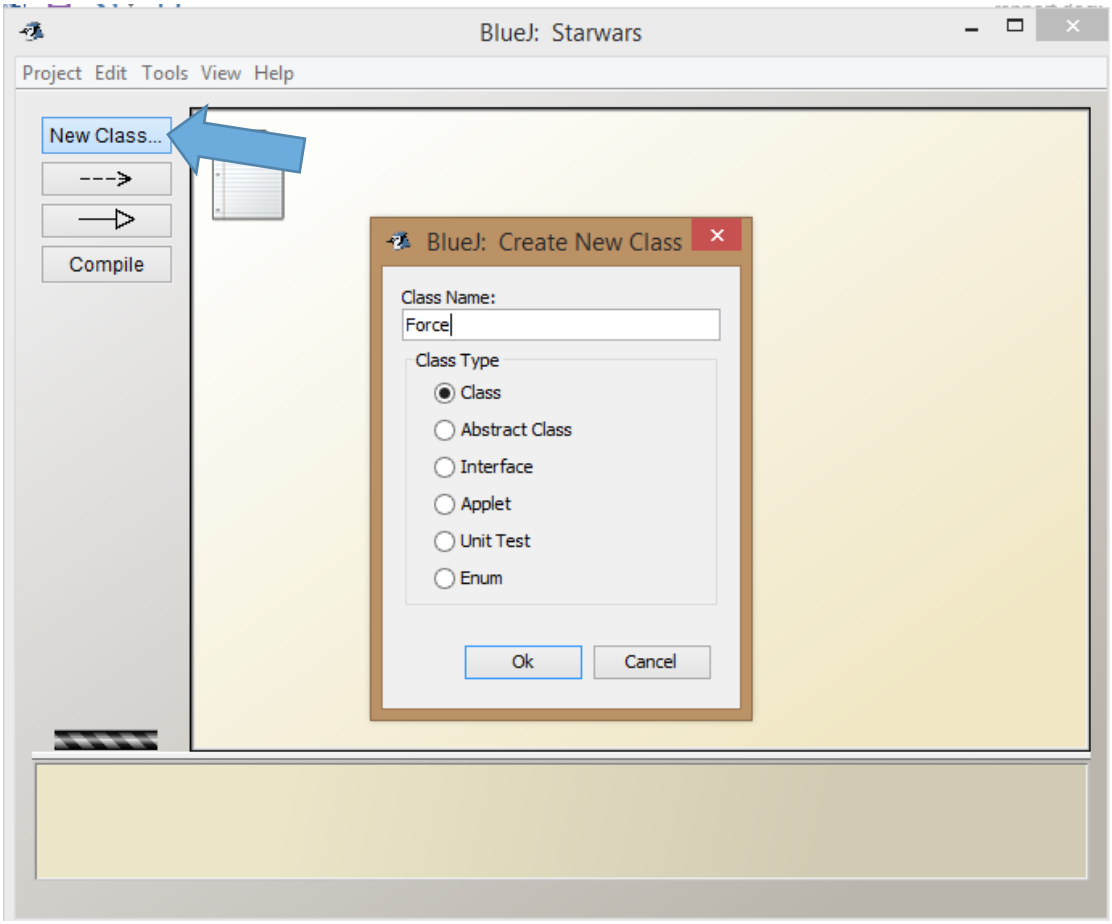
3. Création du projet « starwars ».

Après l'installation, ouvrer le logicielle BlueJ. Cliquer sur « Projet » en haut à gauche puis « New Project… » puis désignez le dossier qui contiendra le projet. Dans notre exemple nous créons le projet « Starwars » dans le dossier TP Agile. Une nouvelle fenêtre BlueJ avec titre « BlueJ : Starwars » s'ouvre alors.



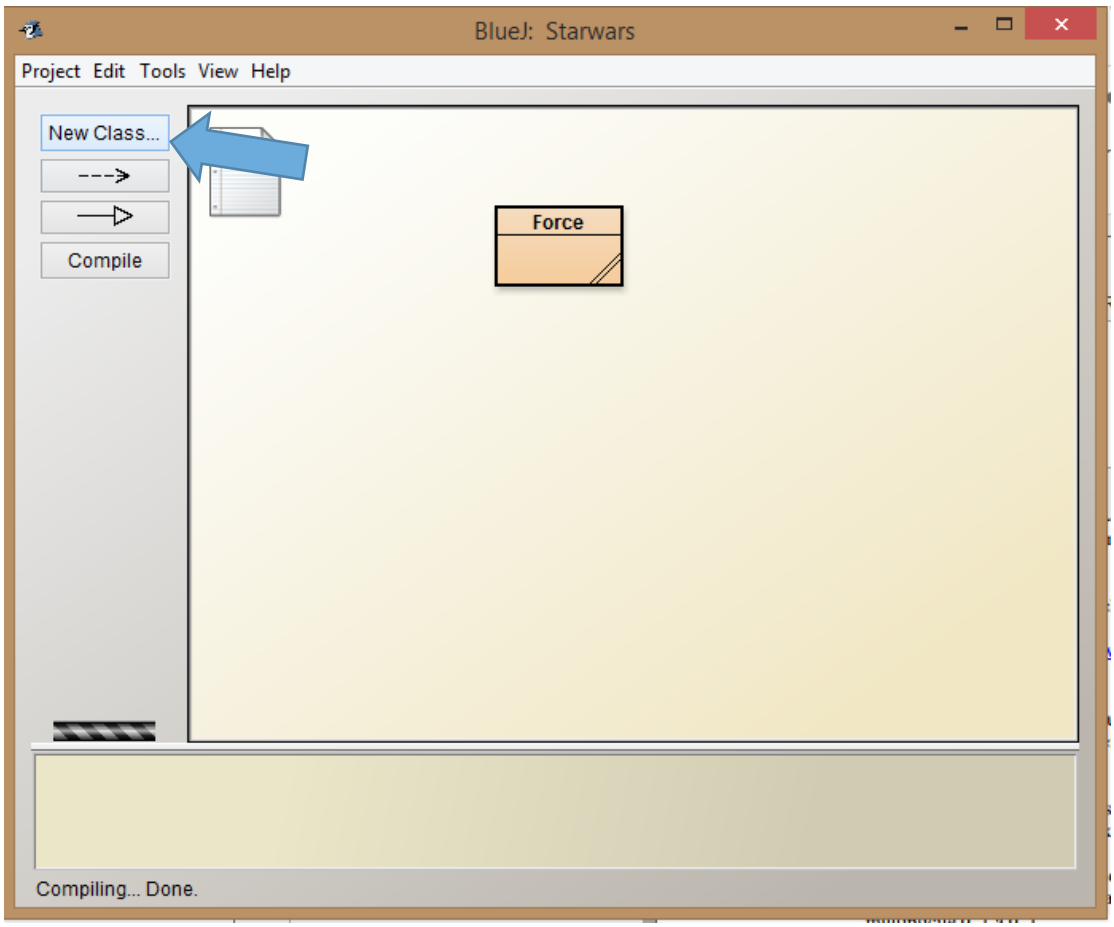
4. **Création de notre classe « Force », notre classe fétiche.**

Cliquez sur « New Class » et nommez la « Force » puis cliquez sur ok.



5. **Compiler la classe**.

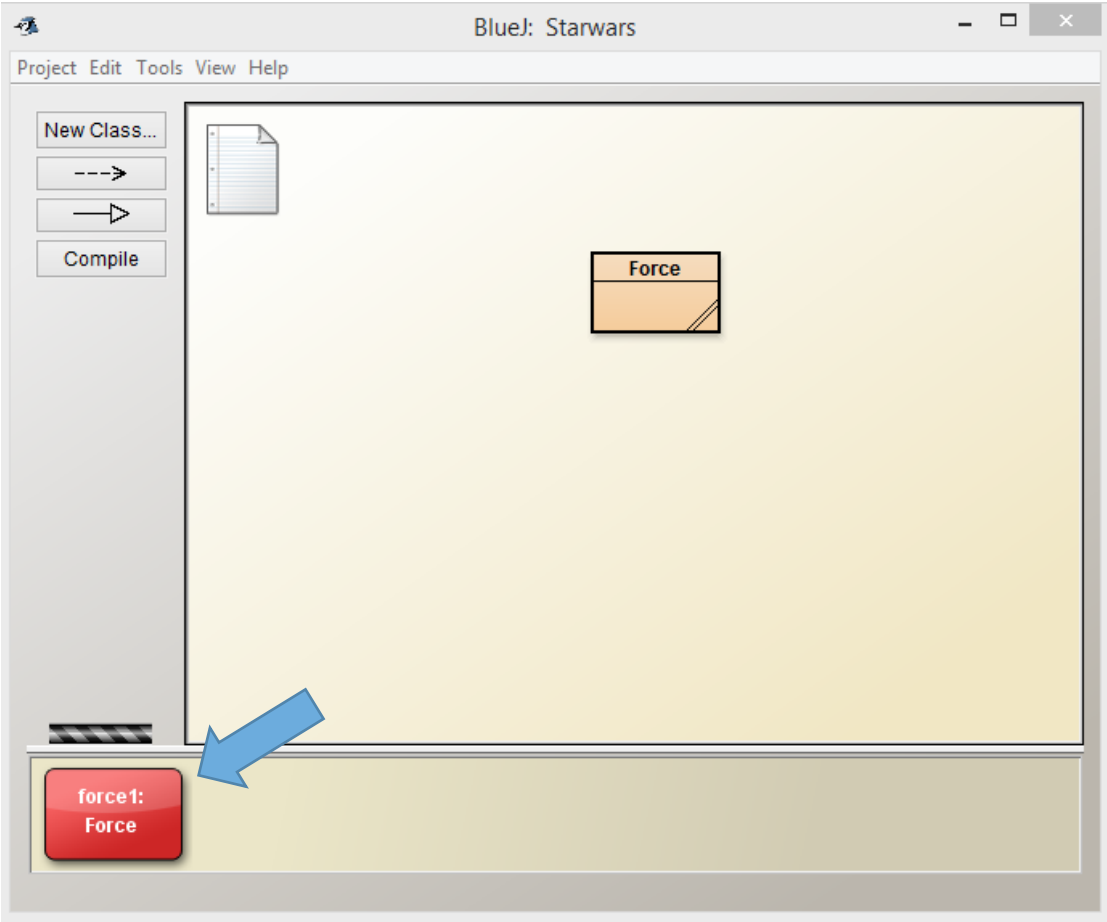
Pour compiler la classe et ainsi pouvoir l'instancier, cliquer sur « Compile ».



Maintenant que la force est instauré dans la galaxie nous pouvons créer des utilisateurs de la force, c'est-à-dire des objets de la classe force.

6. Instanciation de la classe « force ».

Cliquez droit sur la classe puis sur new Force(). Par défaut l'objet est appelé force1 et est visible par le carré rouge.



Nous venons de créer notre premier utilisateur de la force, cependant ce dernier n'a ni nom, ni statut en tant qu'utilisateur de la force et n'a pas encore choisi de quel côté de la force il désire dessiner son destin.

7. Ajout de deux attributs « coteDeLaForce », « nom » et « status » ainsi que la méthode entrainement() .

Cliquez droit sur la classe Force puis Open Editor. Vous pourrez alors ajouter les attributs et méthodes comme sur ci-dessous.

```
public class Force
{
    // instance variables - replace the example below with your own
    private int x;
    String coteDeLaForce;
    String nom;
    int status;

    /**
     * Constructor for objects of class Force
     */
    public Force(String coteDeLaForce, String nom, int status )
    {
        // initialise instance variables
        x = 0;
        this.coteDeLaForce=coteDeLaForce;
        this.nom=nom;
        this.status=status;
    }
```

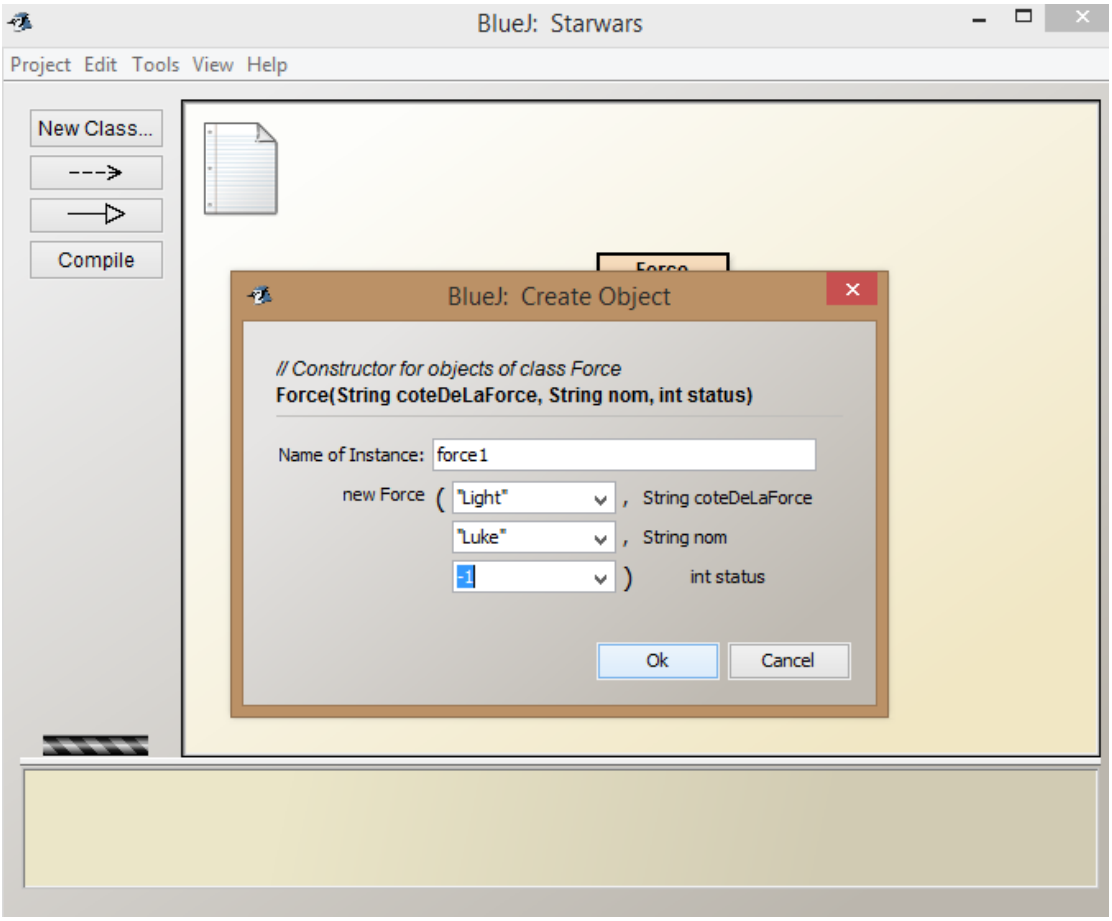
```
    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y   a sample parameter for a method
     * @return     the sum of x and y
     */
    public void entrainement(){
        this.status=status+1;
    }
    public void setCoteDeLaForce(String coteDeLaForce){
        this.coteDeLaForce=coteDeLaForce;
    }
    public void setNom(String nom){
        this.nom=nom;
    }
    public String getCoteDeLaForce(){
        return this.coteDeLaForce;
    }
    public String getNom(){
        return this.nom;
    }
    public void setStatus(int status){
        this.status=status;
    }
    public int getStatus(int status){
        return this.status;
    }

}
```

Voilà, nous somme enfin prêt à créer des utilisateurs de la force, future maitre Jedi ou Sith ainsi qu'à les entrainer dans le maniement de leur pouvoir.

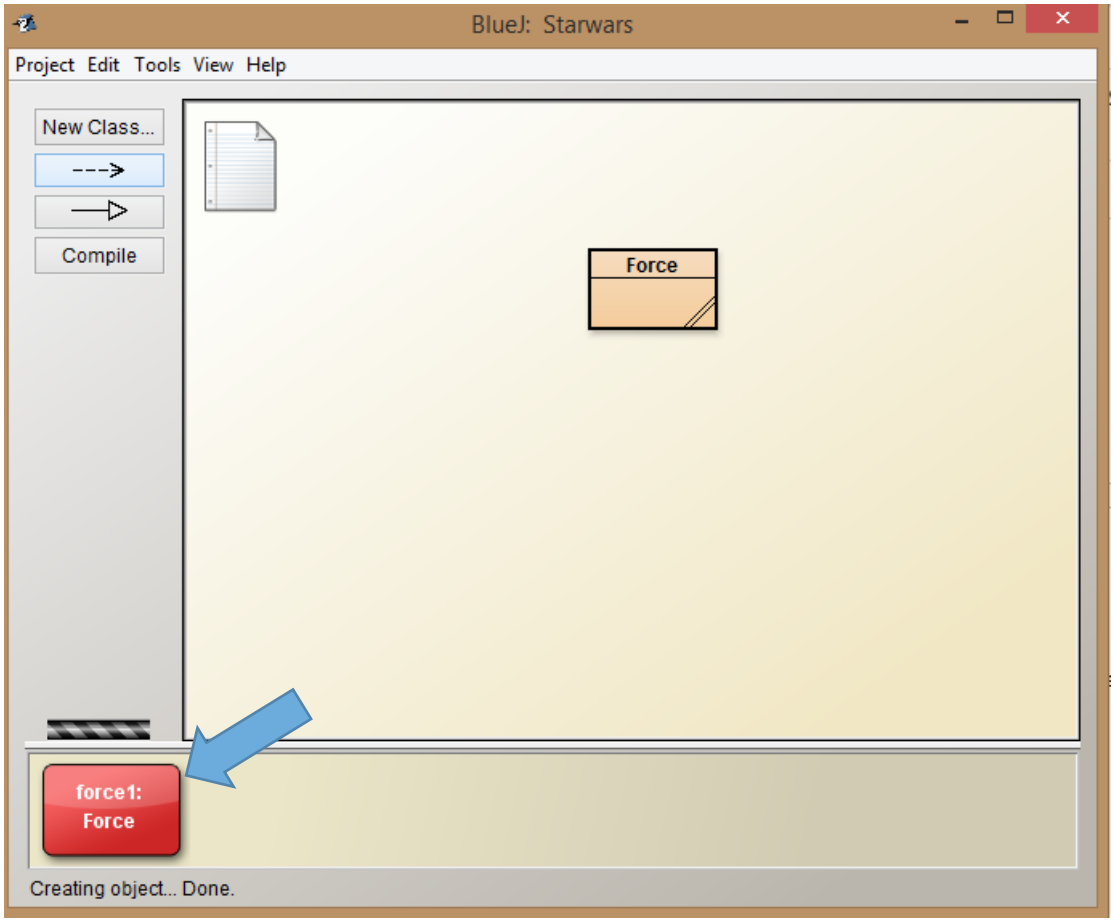
8. Nouvelle instanciation avec exécution de la méthode « entrainement () »

Cliquez sur la classe Force puis sur le bouton Compile. Maintenant que la classe est compilée avec les nouveaux attributs, nous allons créer un objet en cliquant droit sur la classe Force puis new force(…). Vous aurez alors une fenêtre vous demander d'entrer les caractéristiques de votre objet. Une fois celles-ci remplis cliquez sur « ok ».



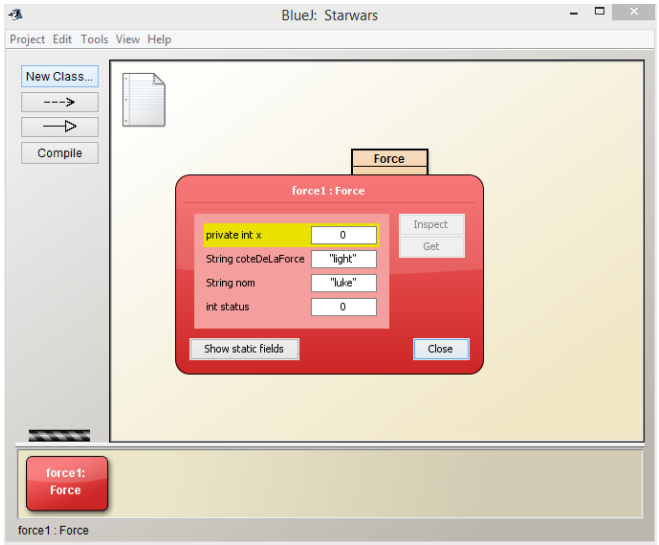
Dans notre exemple, nous donnons naissance à Luke qui à choisit le côté lumineux de la force et qui n'ayant encore reçu aucun entrainement possède un status -1. A noté qu'un utilisateur de la force peut aussi être « indécis » quant à son choix de destin, ainsi il est possible d'entrer « indécis » comme String pour l'attribut « coteDeLaForce ».

L’objet force1 est alors visible dans la partie basse de votre fenêtre.



Maintenant nous allons utiliser la méthode entrainement() sur l’objet force1. Pour cela cliquez droit sur force1 et sélectionner entrainement().

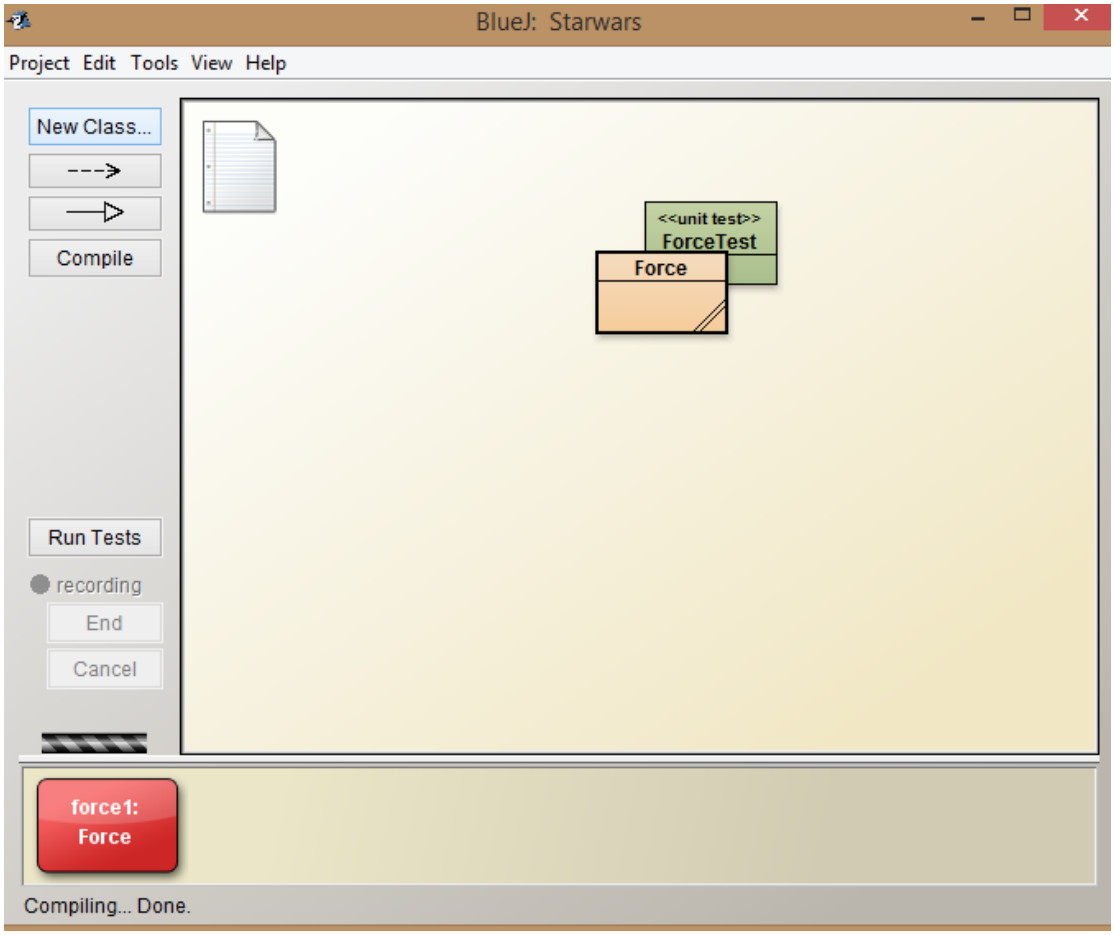
Cliquez droit sur l’objet force1 puis sur Inspect. Vous devriez obtenir la fenêtre suivante :



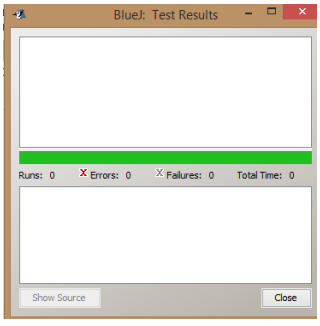
Ainsi on voit que Luke a maintenant un status égal à 0 donc la méthode entrainement à bien augmenter l'attribut status de 1.

9. Test unitaire de la classe « Force »

Pour effectuer les tests unitaires nous ferons apparaître le bouton « run test » sur l'interface en cliquant sur tools/preferences/interface/Show unit testing tool puis sur « ok ». Maintenant que nous pouvons directement lancer les tests depuis l'interface, faites clic droit sur la classe Force et create Test Class. Vous devriez obtenir l'écran suivant :



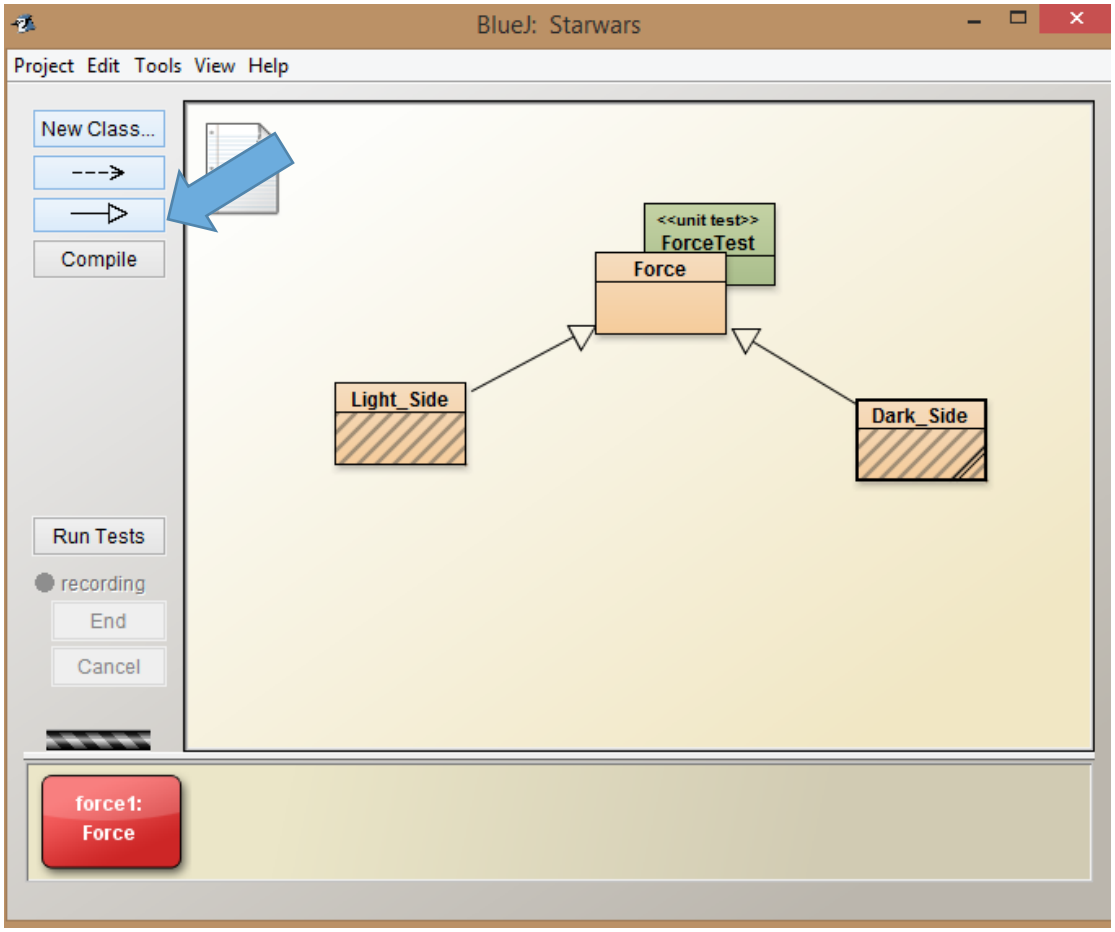
Lancer maintenant le test en cliquant sur Run Tests. Vous devriez obtenir la fenêtre suivante :



La barre verte montre que le test n'a pas trouvé d'erreur.

10. Création des classes « Good_Side » et « Dark_Side » relié à la classe « Force »

Dans cette section nous allons créer deux nouvelles classes reliées à la classe Force. Pour cela cliquer sur New Class puis choisir le nom de la classe, dans notre exemple Light_Side. Répéter cette opération une deuxième fois en appelant la classe Dark_Side. Maintenant cliquer sur la deuxième flèche à gauche pour relier les deux classes avec la classe Force. Vous devriez obtenir l'écran suivant :



Cet héritage signifie que chaque objet de Light_Side ou Dark_Side (les jedis et Siths sont aussi des utilisateurs de la force mais qu'un utilisateur de la force peut être indécis ou ne choisir aucun des deux côtés).

11. Création des méthodes « entrainementJedi() » et « entrainementSith() ».

La méthode entrainementJedi() est une méthode de Light_Side qui utilise la méthode entrainement de Force mais n'est accessible que pour les instances de la classe Light_Side (il n'est pas envisageable qu'une personne n'étant pas un Jedi puisse en recevoir l'enseignement). Il en va de même pour la méthode entrainementSith() qui est une méthode de Bad_Side utilisant la méthode entrainement() de Force. De plus ces méthodes vont manipuler les attributs propres aux classes Light_Side et Dark_Side.

Dans un premier temps, ouvrer l'éditeur de la classe Light_Side et ajouter les lignes suivantes.

```
public class Good_Side extends Force
{
    // instance variables - replace the example below with your own
    private int x;
    protected boolean lightsaber;
    /**
     * Constructor for objects of class Good_Side
     */
    public Good_Side()
    {
        // initialise instance variables
        super();
        lightsaber=false;
    }

    public Good_Side(String nom, int status, boolean lightsaber){
        super("Light",nom,status);
        this.lightsaber=lightsaber;
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y   a sample parameter for a method
     * @return     the sum of x and y
     */
    public void entrainementJedi(){
        this.entrainement();
        if((status==2)&&(lightsaber==false)){
            lightsaber=true;
            System.out.println(this.nom+"reçoit un saber laser de couleur verte et passe padawan");
        }
        if(status==4)System.out.println(this.nom+"passe au rang de maitre jedi");
    }
}
```

Puis de même pour la classe Dark_Side :

```
public class Dark_Side extends Force
{
    // instance variables - replace the example below with your own
    private int x;
    protected boolean lightsaber;
    /**
     * Constructor for objects of class Dark_Side
     */
    public Dark_Side()
    {
        // initialise instance variables
        super();
        lightsaber=false;
        this.coteDeLaForce="Dark";
    }
    public Dark_Side(String nom, int status, boolean lightsaber){
        super("Dark",nom,status);
        this.lightsaber=lightsaber;
    }
```

```
    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y   a sample parameter for a method
     * @return     the sum of x and y
     */
    public void entrainementSith(){
        this.entrainement();
        if((status==2)&&(lightsaber==false)){
            lightsaber=true;
            System.out.println(this.nom+"reçoit un saber laser de couleur rouge et passe apprenti Sith");
        }
        if(status==4)System.out.println(this.nom+"passe au rang de Sith");
    }
}
```

Maintenant nous possédons trois types de personnes utilisant la force. Les instances de la classe Force n'ayant pas choisi leur côté, les jedis et les siths. Ces deux dernières peuvent utiliser leur entrainement et évoluer dans leur côté respectif, obtenir un sabre laser de couleur particulière au status 2 ainsi qu'un statut de padawan ou apprenti. Au statut 4, les padawan deviennent Jedi et les apprentis sith deviennent sith.

Nous pouvons maintenant faire évoluer nos personnages et les entrainer.

12. Instanciation et sauvegarde des objets reliés dans la fixture d'une classe test.

Dans un premier temps nous allons lier les instances des classes entre elles par l'ajout d'un attribut rival dans les classes Light_Side et Dark_Side.
On ajoute alors dans la classe Light_Side :

```
13      protected Dark_Side rival;
```

Ainsi que les méthodes get et set de ce nouvel attribut :

```
44  public void setRival(Dark_Side rival){
45      this.rival=rival;
46   }
47  public Dark_Side getRival(){
48      return rival;
49   }
```

Et dans la classe Dark_Side :

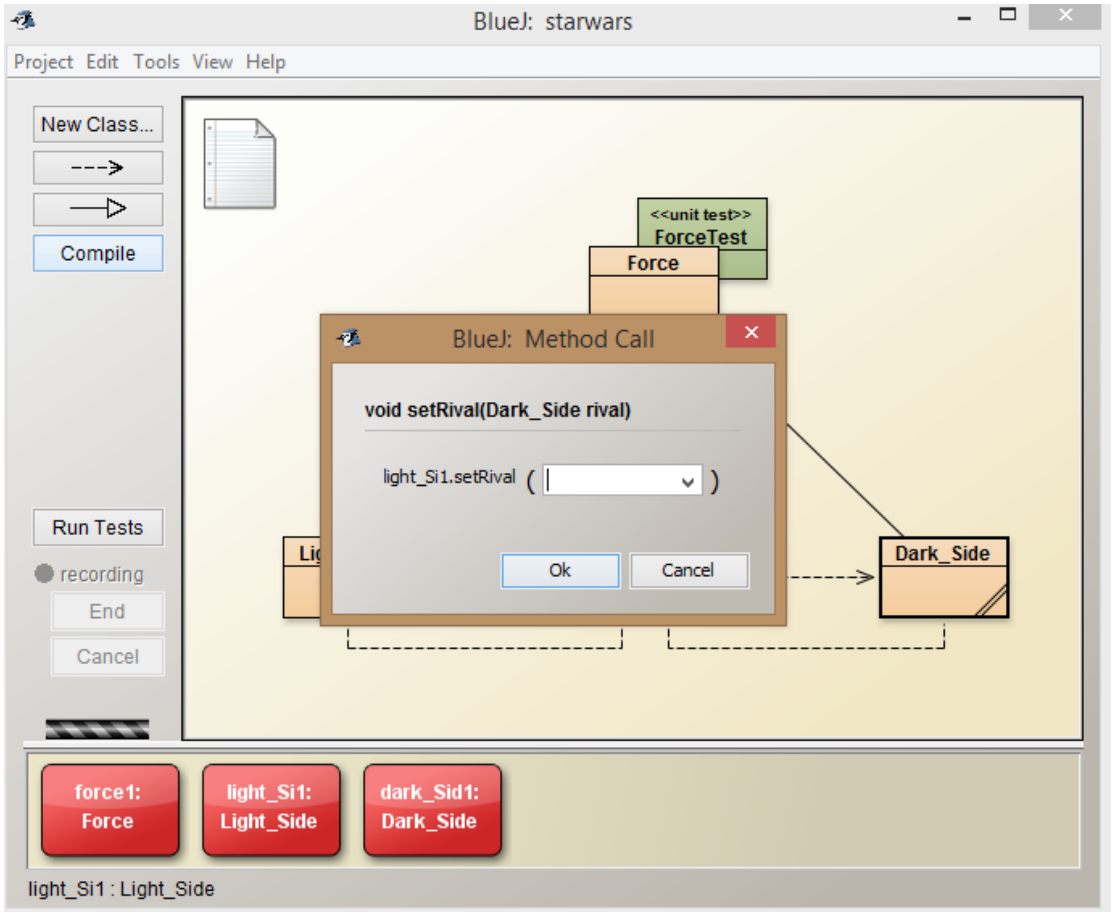
```
13      protected Light_Side rival;
```

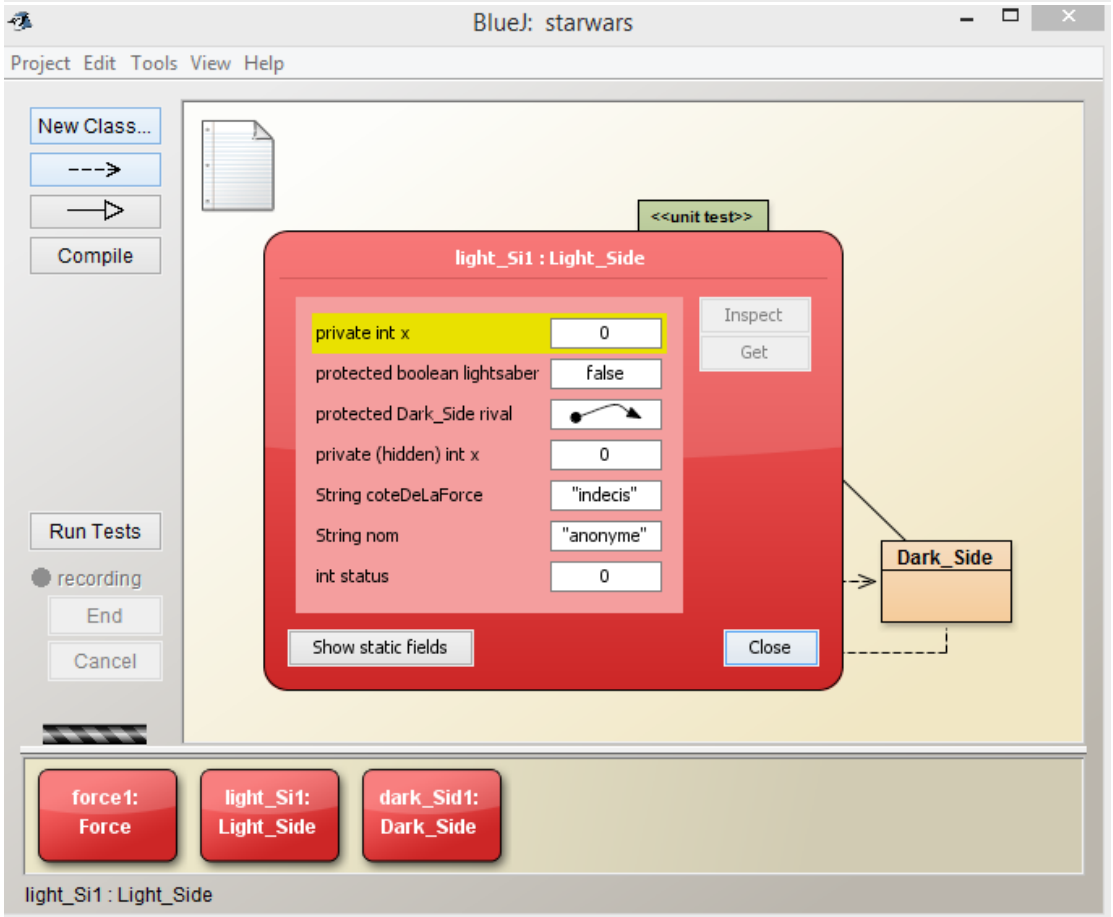
De même que pour la classe Light_Side, les méthodes get et set pour ce nouvel attribut :

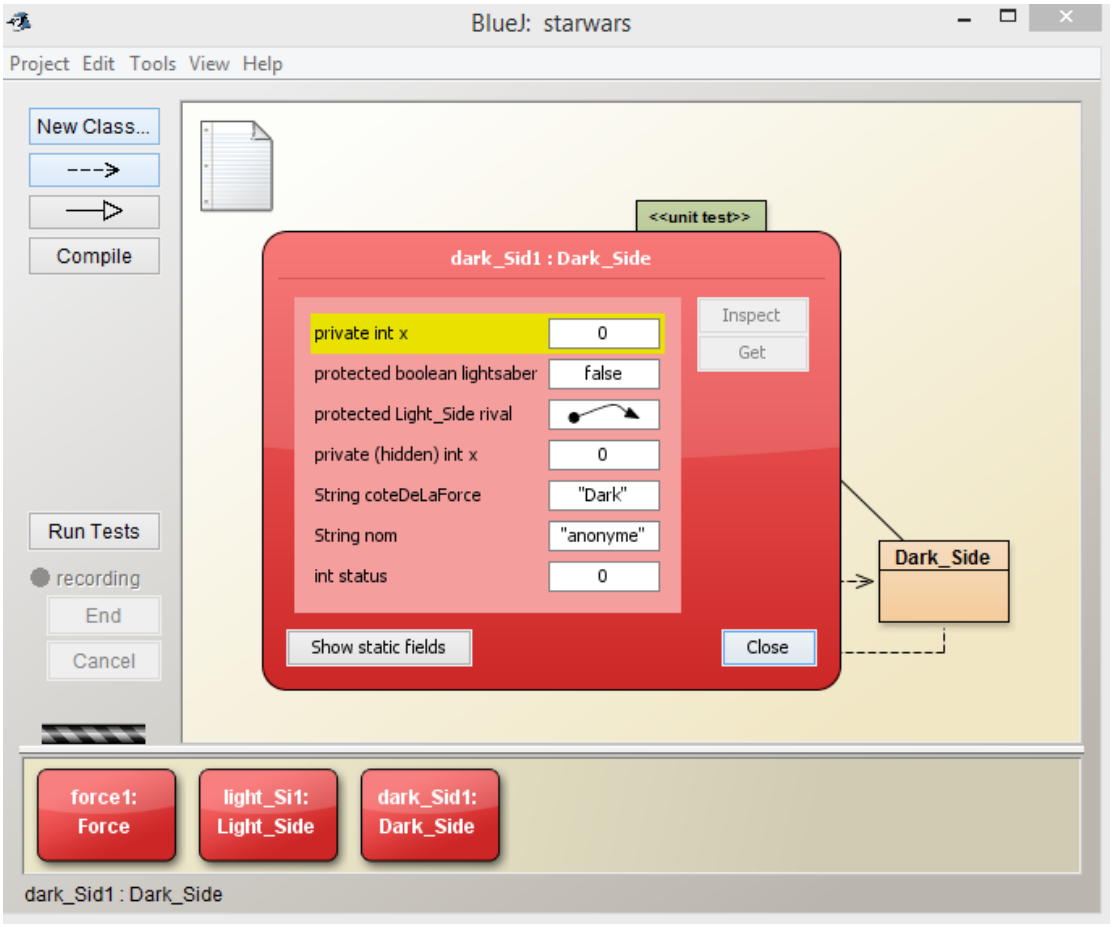
```
42  public void setRival(Light_Side rival){
43      this.rival=rival;
44  }
45  public Light_Side getRival(){
46      return rival;
47  }
```

Puis compiler les deux classes.

Dans un deuxième temps nous allons instancier les différentes classes créées. Créons dans un premier temps une instance de la classe Force, une de la classe Light_Side et une de Dark_Side. Puis utiliser les méthodes setRival(…) des objets light_Si1 et dark_Si1, pour cela clic droit sur l'objet puis sélectionner la méthode setRival(Dark_Side rival) pour l'objet light_Si1 (respectivement la méthode setRival(Light_Side rival) pour l'objet dark_Si1)

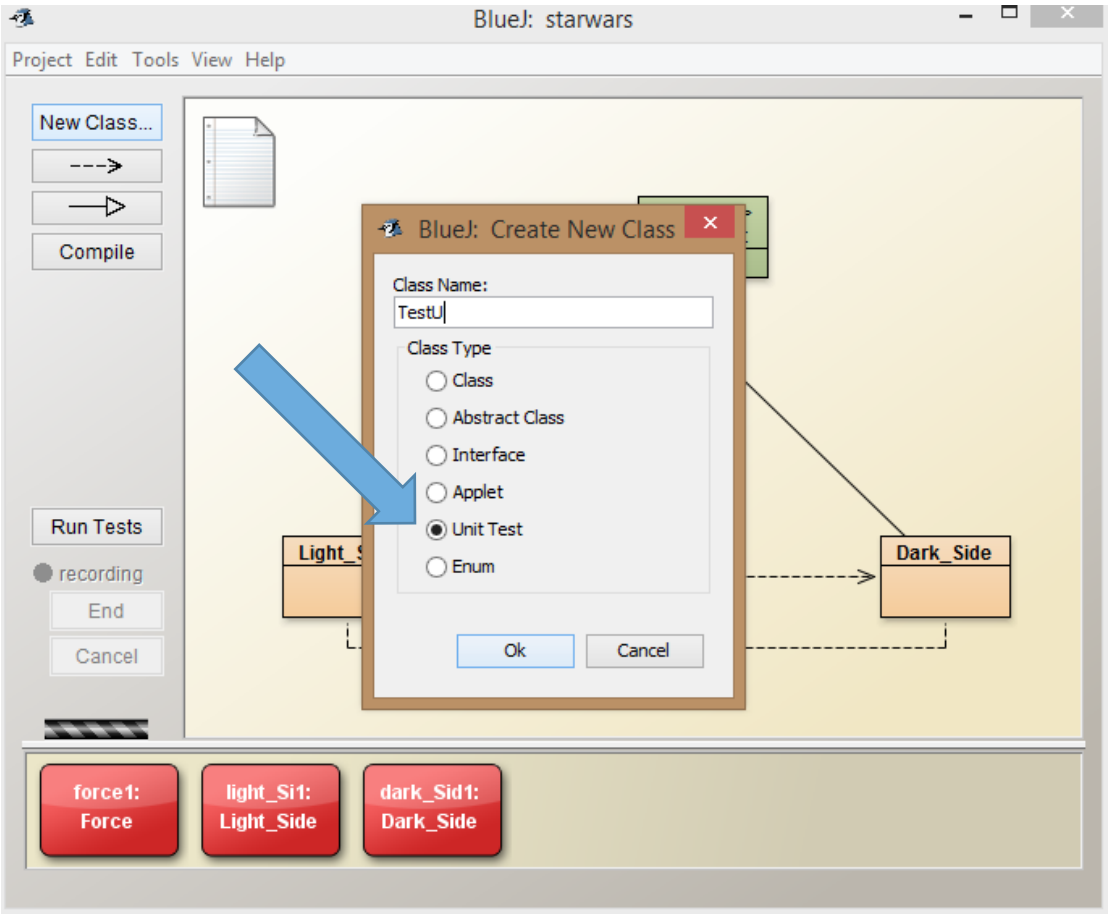

BlueJ: starwars
Project Edit Tools View Help
New Class...
Compile
Run Tests
recording
End
Cancel
<<unit test>>
ForceTest
Force
BlueJ: Method Call
void setRival(Dark_Side rival)
light_Si1.setRival ( )
Ok
Cancel
Dark_Side
force1:
Force
light_Si1:
Light_Side
dark_Sid1:
Dark_Side
light_Si1 : Light_Side


BlueJ: starwars
Project Edit Tools View Help
New Class...
Compile
Run Tests
recording
End
Cancel
<<unit test>>
light_Si1 : Light_Side
private int x 0
protected boolean lightsaber false
protected Dark_Side rival
private (hidden) int x 0
String coteDeLaForce "indecis"
String nom "anonyme"
int status 0
Inspect
Get
Show static fields
Close
Dark_Side
force1:
Force
light_Si1:
Light_Side
dark_Sid1:
Dark_Side
light_Si1 : Light_Side



Nous avons maintenant un Jedi et un Sith liés par un pacte de rivalité grâce à ce nouvel attribut.

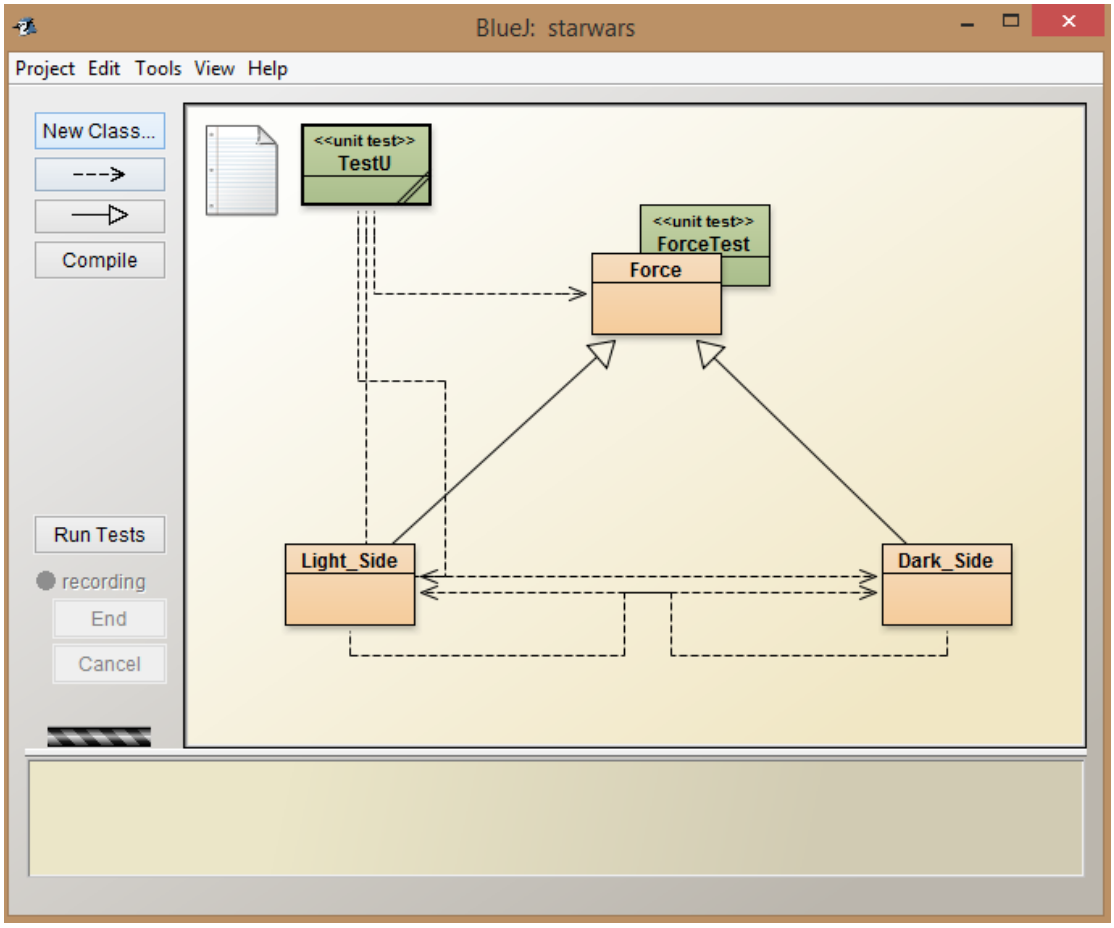
Maintenant nous allons sauvegarder tous ces objets dans une classe test. Pour cela créer une classe test en cliquant sur New Class et choisir la catégorie Unit Test.



Puis cliquer sur ok.

Ensuite clic droit sur la classe test récemment créé puis « Object Bench to Test Fixture ».

Vous devriez alors obtenir l'écran suivant :

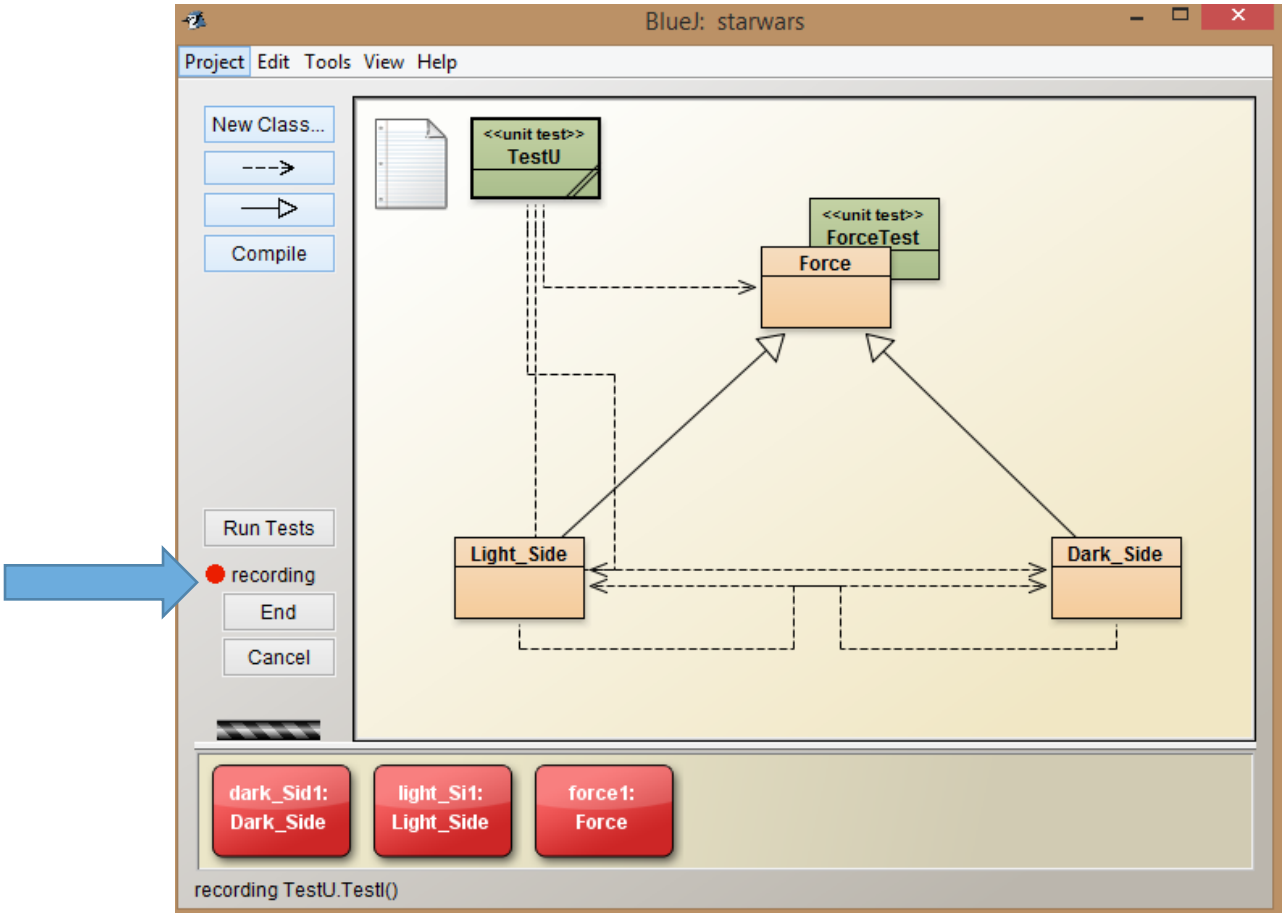


Voilà ce que vous devriez retrouver dans votre class test, ce qui représente les différentes opérations effectuées sur les objets sauvés dans la fixture.

```
@Before
public void setUp()
{
    force1 = new Force();
    light_Si1 = new Light_Side();
    dark_Sid1 = new Dark_Side();
    light_Si1.setRival(dark_Sid1);
    dark_Sid1.setRival(light_Si1);
}
```

13. Création d'une méthode interactive de test se basant sur la fixture de l'étape 12 et exécution du test.

Afin de créer une méthode de test intéractive, clic droit sur la classe TestU puis Create Test Method et nommez là TestI.

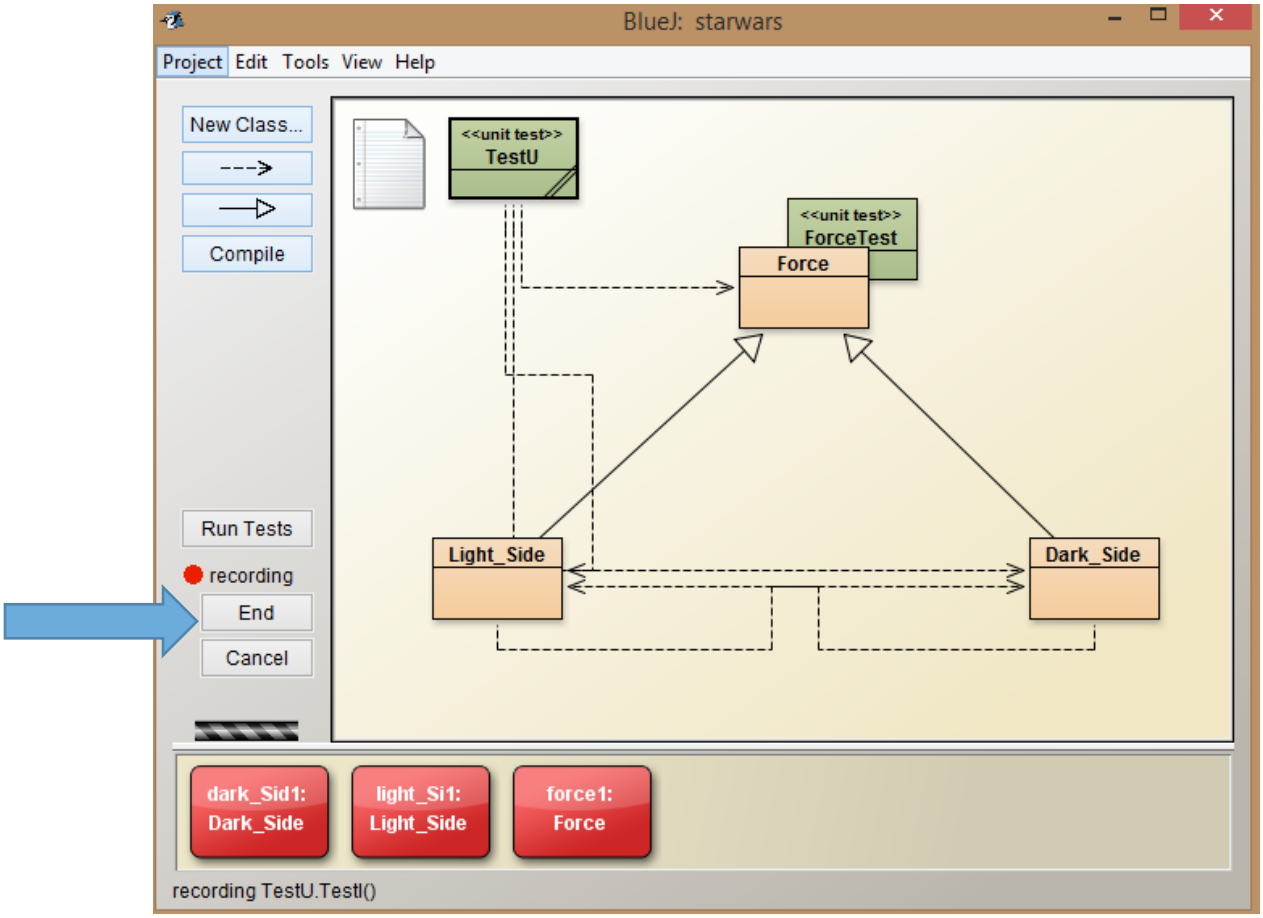


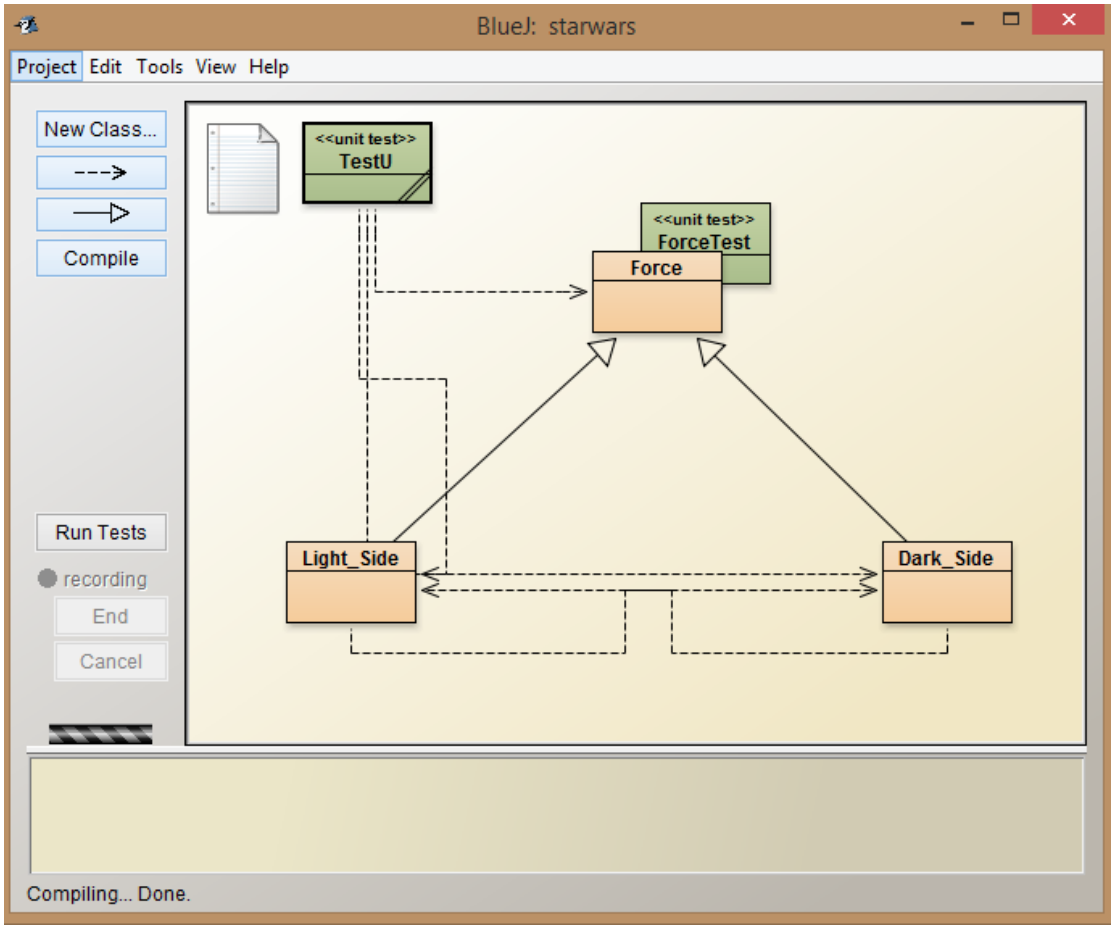
Le test est alors en train d'être enregistrer comme le montre le point rouge à côté de recording. Maintenant nous allons faire évoluer nos personnages. Pour cela effectuer 4 fois la méthode entrainementJedi() sur light_Si1 et 4 fois la méthode entrainementSith() sur dark_Sid1.
Vous devrez obtenir ceci :

```
anonymereçoit un saber laser de couleur verte et passe padawan
anonymepasse au rang de maitre jedi
anonymereçoit un saber laser de couleur rouge et passe apprenti Sith
anonymepasse au rang de Sith
```

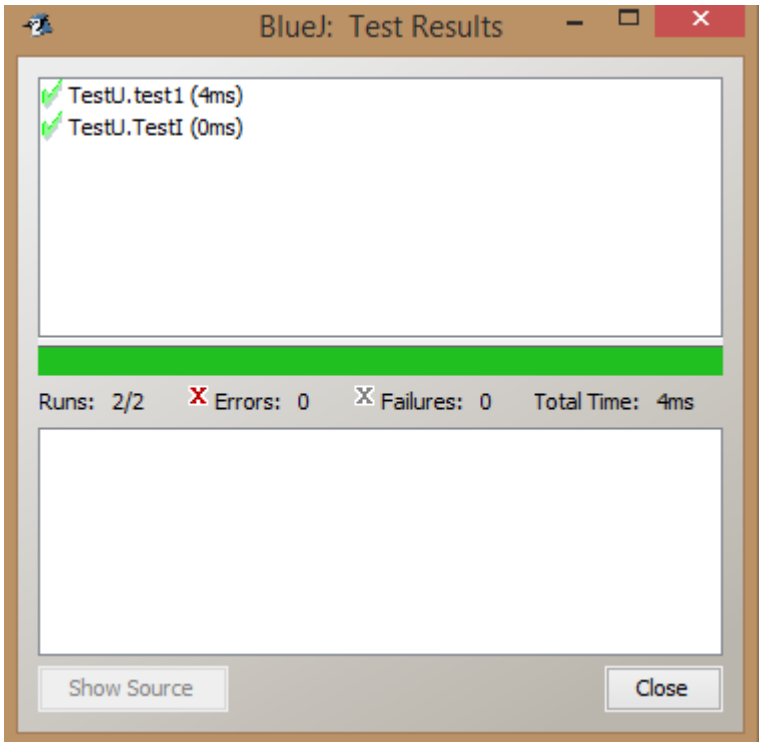
Ayant utilisé les constructeurs par défaut nos objets ont pour attribut nom = « anonyme »

Maintenant presser le bouton End





Puis Run Tests.



Lors de l'établissement de ce tutoriel, deux types de test ont été effectué, d'où la présence du TestU. qui ne devrait pas apparaître sur votre machine.

Nous avons alors pu créer un test interactif et le sauvegarder sous le nom de TestI.

Conclusion Première étape

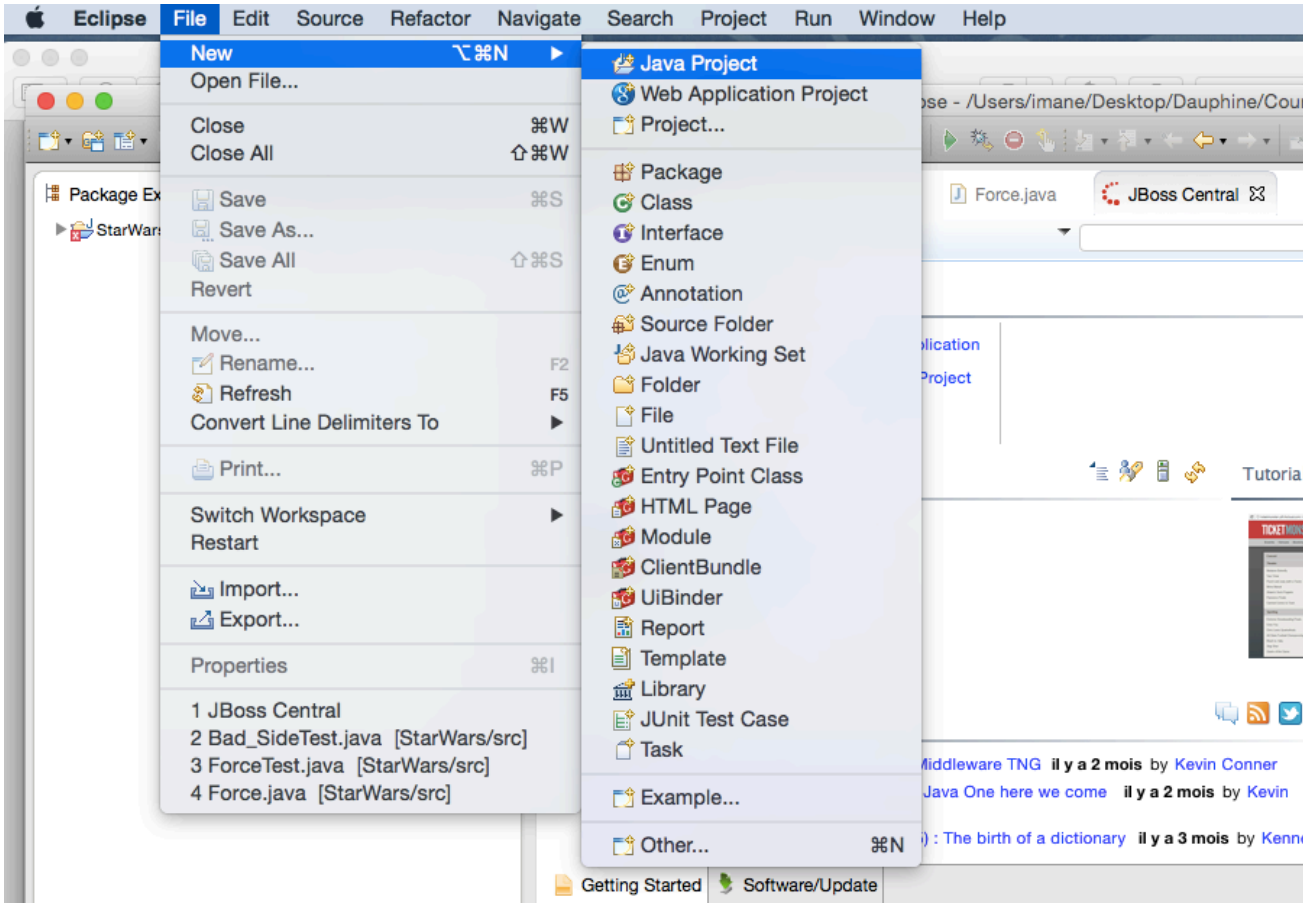
Vous avez appris à utiliser BlueJ et avez maintenant une connaissance solide en test unitaire ainsi qu'une base ludique pour vous exercer : le vaste monde de StarWars. Vous pouvez alors faire évoluer ce monde de StarWars en toute tranquillité.
Maintenant vous êtes prêt à pouvoir créer des armés de Jedi ou de Sith, quelle voie sera la vôtre ?

III. Deuxième Partie : Eclipse et jUnit

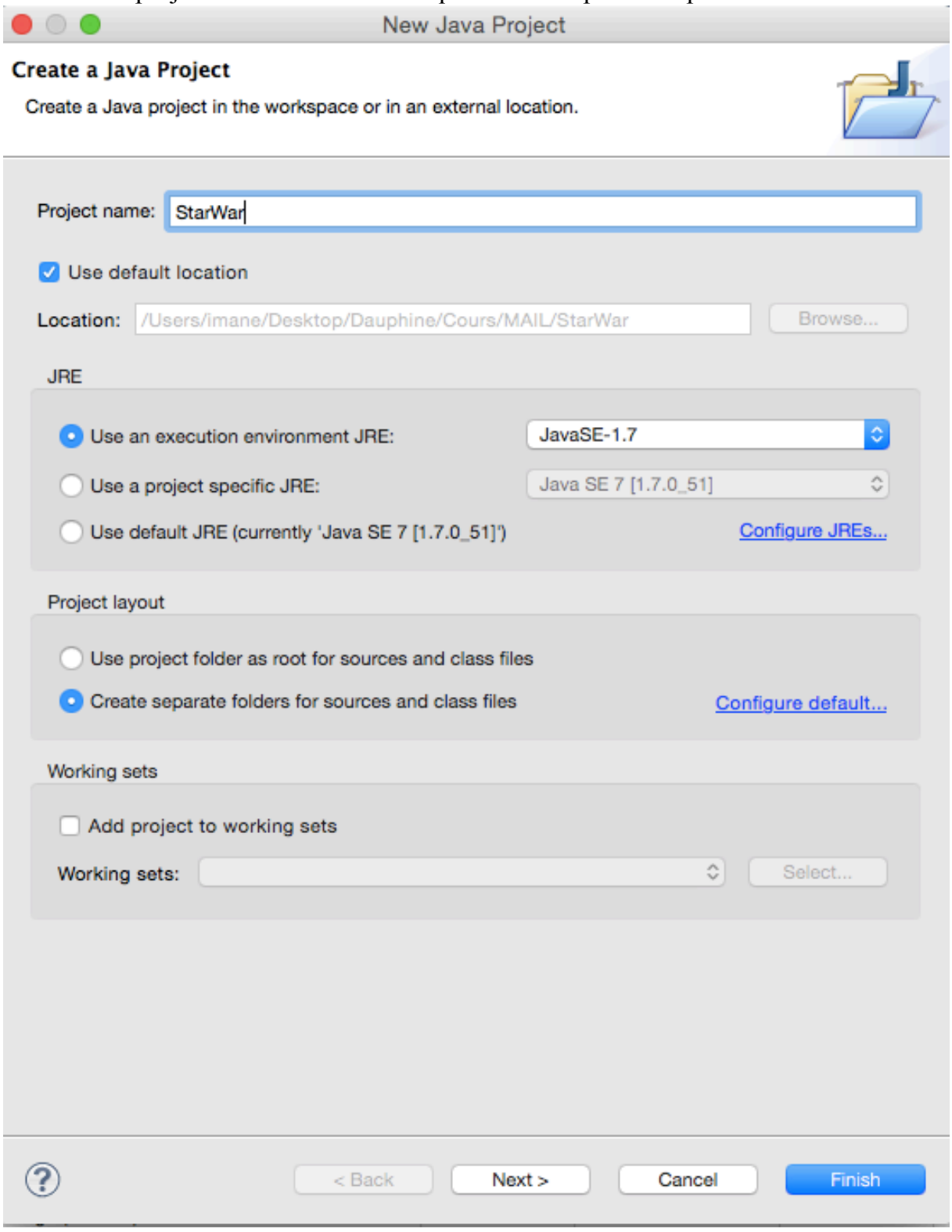
Dans cette deuxième partie, le projet est exporter sur Mac et les classes Light_Side et Dark_Side deviennent Good_Side et Bad_Side

14. Créez un projet éclipse

Ouvrer votre version d'éclipse puis suivez comme suit dans les captures :

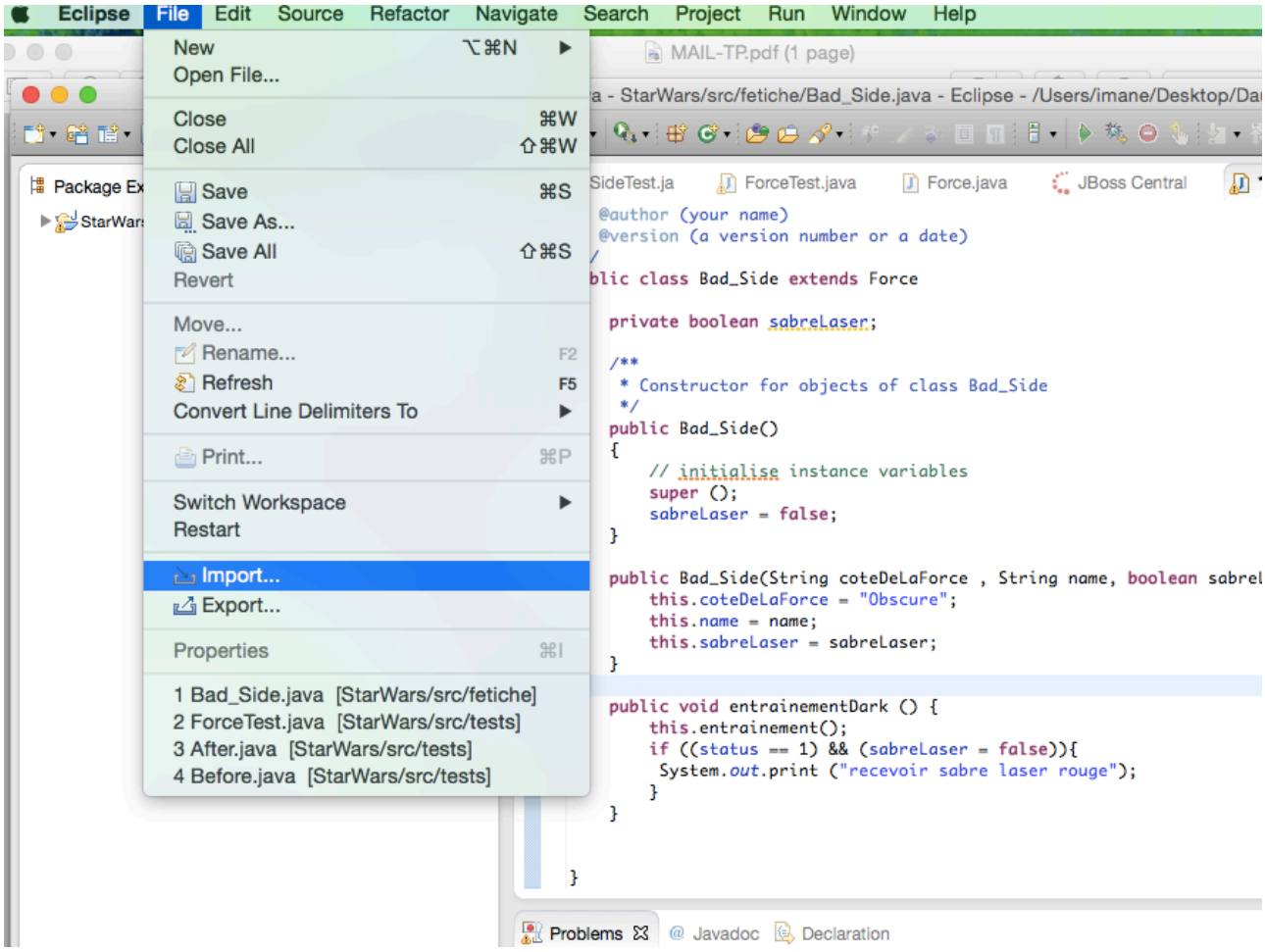


Nommez le projet de la même manière que lors de la première partie :

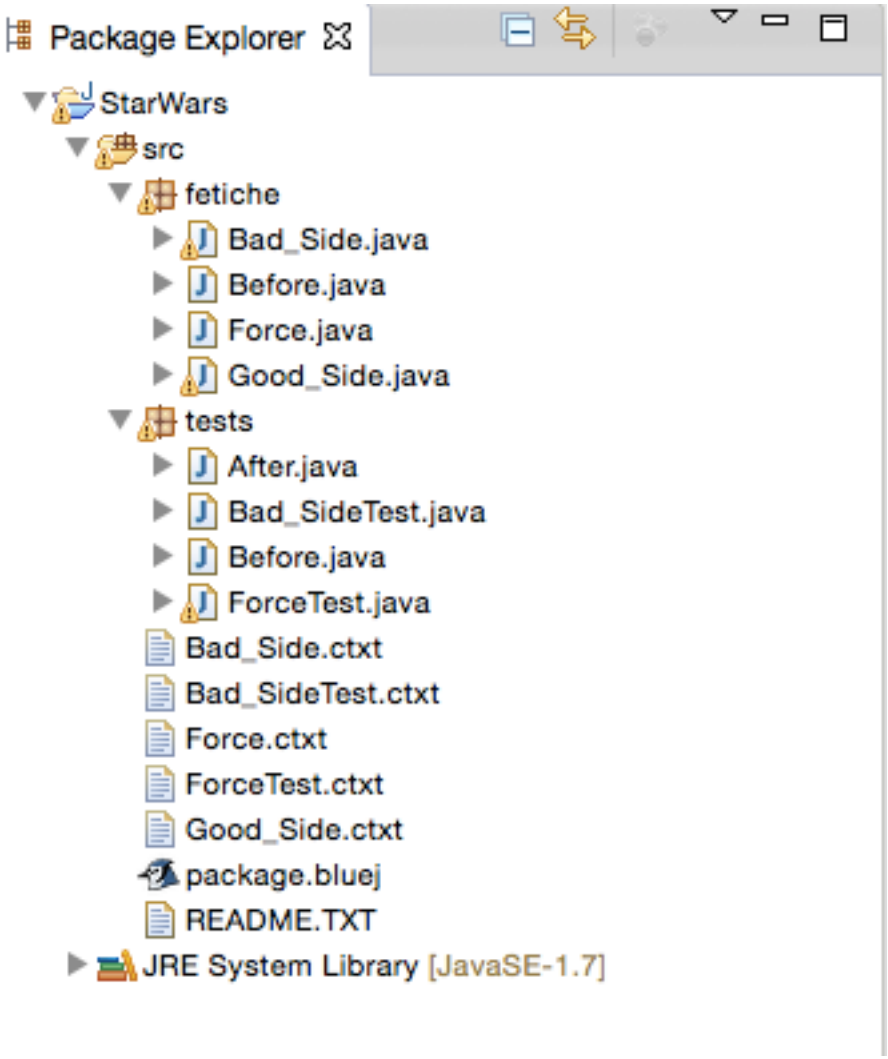


15. **Importez les classes élaborées en BlueJ et organisez-les dans un package dédié**

Importer maintenant les classes crées lors de la première étape dans votre projet éclipse.



Organisation en Package :
Voici l'organisation conseillé lors de ce tutoriel :



16. Implementer une association bidirectionnelle « 0..1 à *» en l'encapsulant bien et testez unitairement sa robustesse.

Cette operation conciste en l'instanciation des classes « Good_Side » et/ou « Bad_Side » ce qui crée une association bidirectionnelle de type « 0..1 à * » avec la classe Force qui est la classe mère de Good_Side et de Bad_Side, ceci dû à l'inexistence d'héritage multiple en Java.

17. **Illustrez l'usage de deux techniques de refactoring (ex : rename et extractMethod)**

Rename Method

Renommer les une propriété, les attributs, les méthode ou les objets peut réduire la nécessité de rajouter des commentaires de code et presque toujours contribue à promouvoir une plus grande clarté.

Cette technique repose sur les éléments donnant un nom descriptif ce qui facilite la lecture et la compréhension du code.

Exemple :

```
public class Force
{
    // instance variables
    protected String coteDeLaForce;
    protected String name;
    protected int status;
    /**
```

Les attributs de notre classe « fétiche » Force portent des noms descriptifs qui n'ont pas besoin d'être accompagnés de commentaires supplémentaires pour connaître leurs désignations. Ce qui respecte la technique de refactoring « rename ».

Extract Method

Cette technique de refatoring repose sur une mise en œuvre très simple. Elle consiste en la rupture de longues méthodes en déplaçant des morceaux trop complexes de code dans de nouvelles méthodes qui ont des identifiants plus descriptifs.

Dans notre cas, on ne dispose pas de méthodes très complexes ou longues, donc l'utilisation de cette technique de refactoring.

18. **Trouvez et parcourez le site officiel de jUnit. Lisez l'article « Test infected » et proposez une amélioration équivalente adaptée à votre exemple.**

Dans l'article « Test infected » du site officiel de jUnit, il est conseillé de créer une classe de tests dès le début du projet et de rajouter des sous classes de test pour chaque fonctionnalité du projet. Dans notre exemple on peut prévoir alors des sous classes de tests ne concernant que la classe Good_Side et d'autres ne concernant que la classe Bad_Side. De plus notre projet n'est qu'une base d'un projet plus grand, nous pourrions très bien créer un système de combat entre jedi et sith, ainsi toutes les fonctionnalités liées à cet addon serait testées par une sous classe de test dédiée uniquement aux méthodes de « combat ».

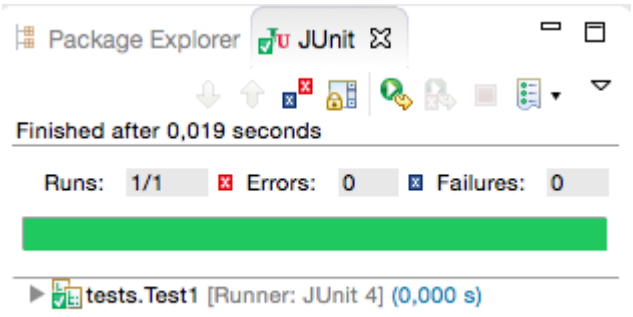
19. **Exécutez les tests en ligne de commande**

Ligne de commande pour compiler la classe de test:

javac -cp .;/Users/imane/Downloads/junit-4.12.jar Test1.java
généralement javac -cp .; "path vers junit.jar" "nom de la classe de test"

Ligne de commande pour lancer le test:

java -cp.;/Users/imane/Downloads/junit-4.12.jar org.junit.runner.JUnitCore Test1.java
généralement javac -cp .; "path vers junit.jar org.junit.runner.JUnitCore" "nom de la classe de test"



20. **Citez une loi de Murphy associez-là à une situation que vous avez rencontré dans ce périple.**

> « Tout ce qui est susceptible de mal tourner, tournera nécessairement mal»
>
> — Edward A. Murphy Jr.

Dans notre projet, nous avons créé une première version de la méthode « entrainementJedi » et son inverse « corruptionEmpereur » qui avaient pour but respectif de transformer des instances de la classe en « Bad_Side » en des instances de la classe « Good_Side » et inversement. Nous avons voulu supprimer les instances et en recréer une autre de l'autre classe, cependant la destruction d'instance nous ayant paru compliquer pour ce tutoriel, nous avons décidé de se baser sur les méthodes entrainements spécifiques à chaque classe.