TD Java + Tests unitaires

Les sources sont à récupérer sur

http://mass-cara2.univ-tlse2.fr/~brahim/test-logiciel/partie0/

1. Tester des instances (sur des exemples significatifs)

Il s'agit de réaliser le test de la classe `JaugeNaturel`.

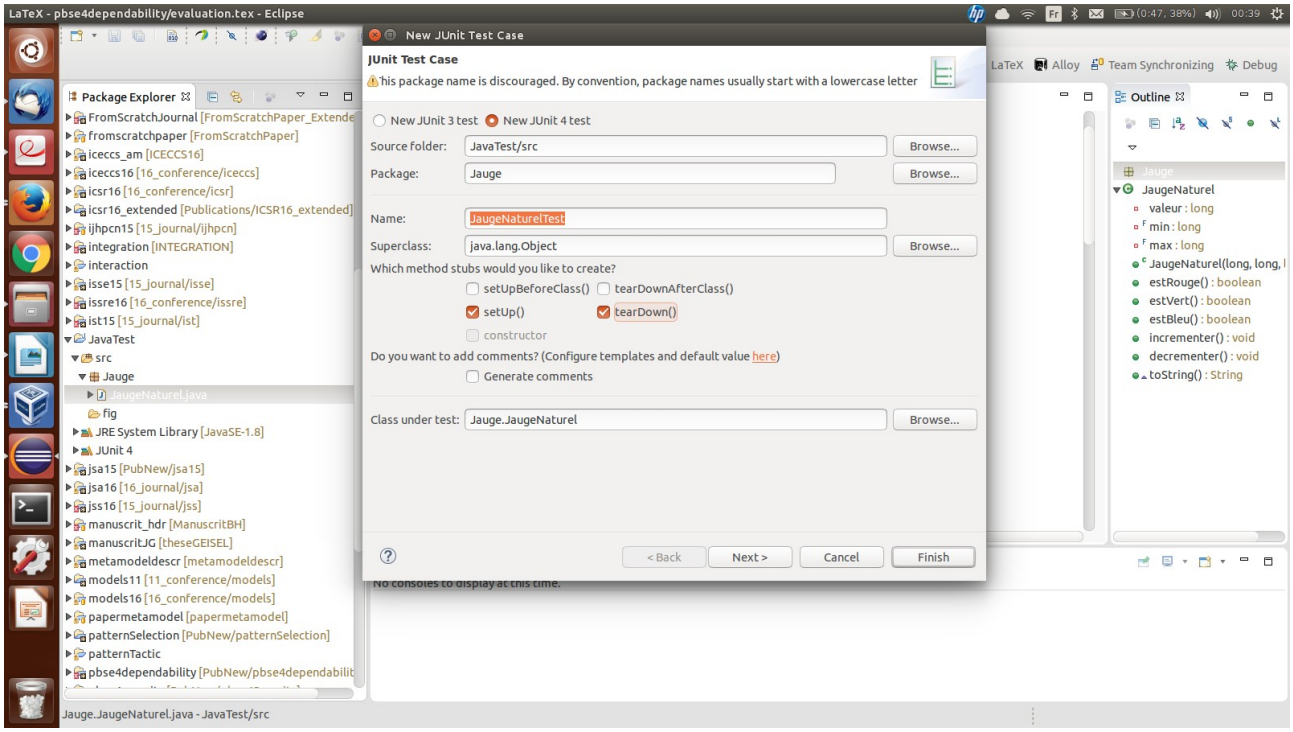
Le premier réflexe est de compiler le code fourni. Si vous rencontrez une erreur, vous devez faire un rapport d'erreur à l'encadrant avec description de l'erreur.

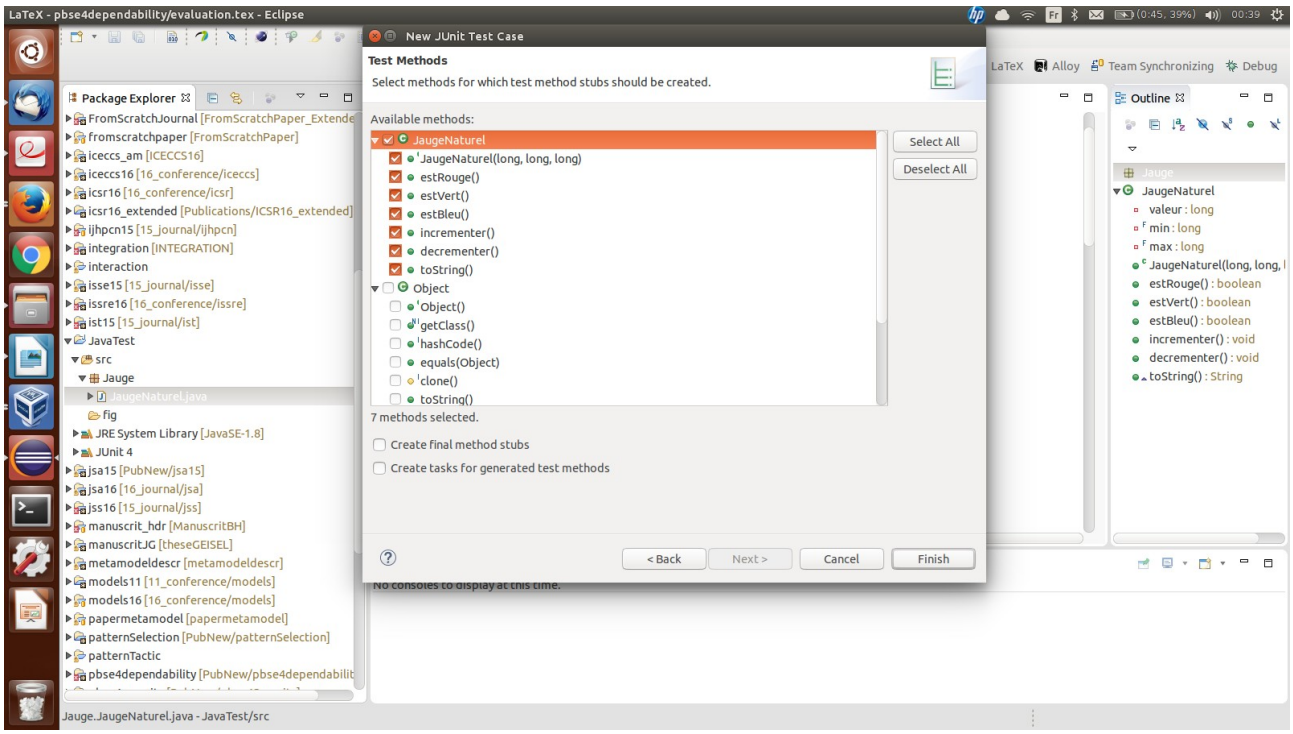
Le travail consiste à écrire ces tests et à les faire passer au code fourni. Pour s'assurer que le code (syntaxiquement juste) correspond bien à la spécification (aux fonctionnalités).

1.1 Mise en place de la classe de tests

1. définir la classe de tests dans un fichier source,
2. définir la première méthode de test de cette classe,
3. écrire le code de votre test dans cette méthode en utilisant le mécanisme des assertions inclus dans java.

On va utiliser le framework Junit pour générer la suite des tests. On peut faire un clique droit sur la classe (exemple JaugeNaturel.java) , sélectionner New.... Dans le menu proposé, il y a l'item Junit Test Case. Le sélectionner, puis suivre les deux étapes montrées ci-dessous :





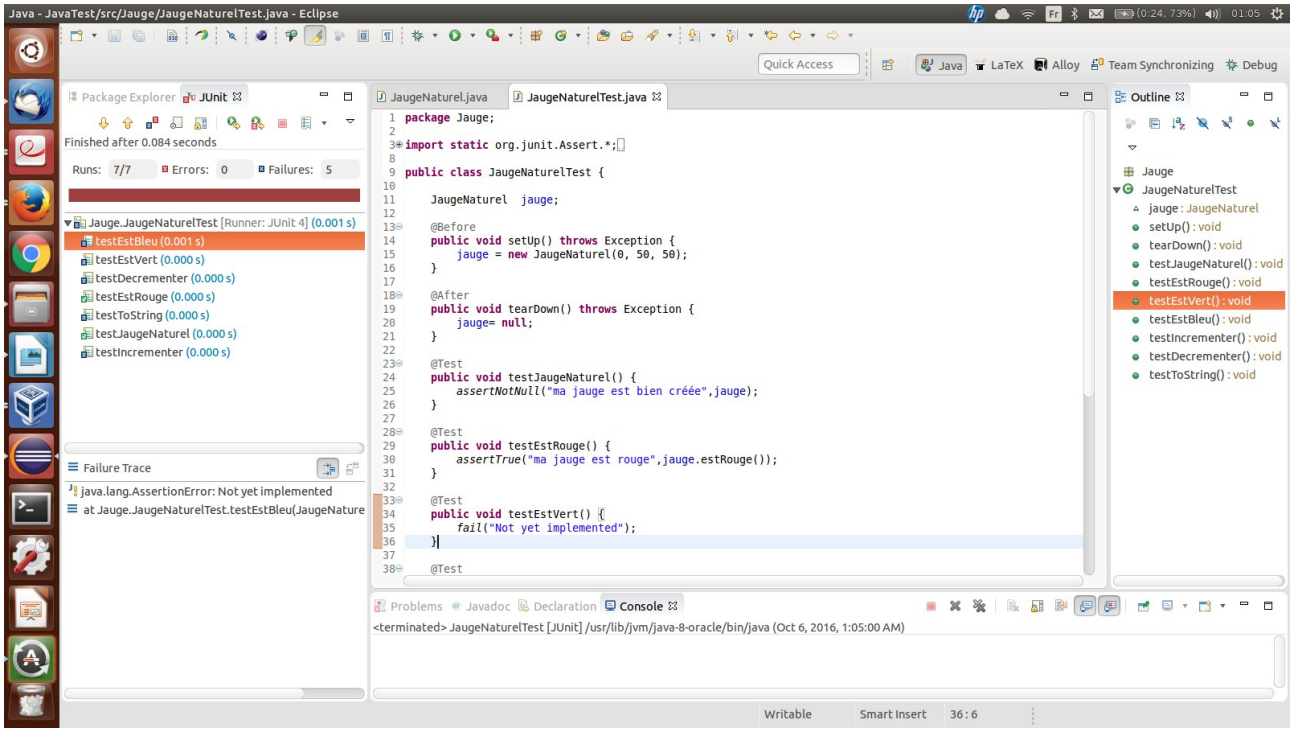
1.2 Réaliser la suite des tests générées : Mécanisme des assertions

Avant de continuer les tests, observons le traitement des assertions (voir le cours) .

Que se passe-t-il si une ou plusieurs assertions sont invalides ?

Modifier les deux premières assertions de votre test pour avoir un résultat faux.

Compiler et exécuter. Combien y-a-t-il d'erreurs à l'exécution ? Pourquoi ? Indiquer les informations données par le message d'erreur.



Les premiers tests à écrire concernent toujours l'état après instanciation.
Pour écrire le code du test, il faut répondre aux questions suivantes :

1. *Comment déclarer une variable pour contenir une telle instance ?*
2. *Comment s'écrit l'instanciation de la classe ?*
3. *Comment s'écrit un envoi de messages à cette instance ? (Comment faire appel aux méthodes estBleu())*

Réaliser itérativement le reste des tests, en apportant les modifications nécessaire au code de la classe `JaugeNaturel`. Suivez la même procédure : écrire un test, compiler, exécuter.

1.3 Réaliser une suite de tests spécifiées

Le reste des tests du développeur pour la classe `JaugeNaturel` sont décrits par la documentation de la classe `JaugeNaturelTest`

Réaliser itérativement le reste des tests dans l'ordre indiqué par la documentation . Suivez la même procédure : écrire un test, compiler, exécuter.

D'après la documentation, le premier test à écrire est :

- ***testDansIntervalle*** : test sur une instance avec une valeur de départ comprise dans l'intervalle de vigie pour la classe `JaugeNaturel` . Pour le test, il nous faut des valeurs ; prenons : depart vaut 100, vigieMin vaut -345 et VigieMax vaut 67899.

  Sur l'état de cette instance, nous avons les trois assertions suivantes :

  - `estBleu()` est faux,
  - `estVert()` est vraie,
  - `estRouge()` est faux.

pour finir votre suite de test

- La méthode ***testLimiteVigieMaxInferieurVigieMin()*** aborde le test des cas limites ici le passage d'un paramètre invalide.

  Pour l'instant, il s'agit de montrer le comportement si le cas limite n'est pas traité. Le traitement des paramètres invalides sera abordé plus tard avec l'utilisation des exceptions qui permettent d'indiquer une erreur.

  Envisagez-vous d'autres cas limites ?

2. Quelques remarques pour l'écriture des tests :

- Pour les repérer, le nom des méthodes de test commence par le mot "test".
- Les tests doivent être indépendant pour éviter les effets de bord. Pour chaque méthode de test, il est préférable de manipuler des variables locales et d'utiliser de nouvelles instances.
- A chaque exécution de la classe de test, toutes les méthodes de test doivent être appelées. Il est inutile de vérifier certaines assertions à chaque test par exemple les assertions après l'instanciation (une seule fois suffit).
- Prenez des valeurs différentes pour l'initialisation des objets à tester.
- Vous pouvez rencontrer des incohérences dans l'implémentation de la classe `JaugeNaturel`. Cela provient souvent d'une spécification imprécise, incomplète ou mal interprétée. Demandez à l'encadrant avant de modifier la classe. (soit l'implémentation est incorrecte, soit l'écriture du test est incorrecte).
- Les tests servent aussi de débogage. Eviter de tester trop de code à la fois. Dans notre exemple, pour tester le comportement des trois méthodes `estBleu()`, `estVert()`, `estRouge()`, nous n'avons pas utiliser les deux méthodes `incremente()` et `decremente()` mais uniquement l'instanciation (impossible de faire moins).
- Pour déboguer pendant un test, utilisez la méthode `toString()` pour afficher l'état (elle est là pour cela). Après débogage ne conserver aucun affichage dans le test.