JUnit

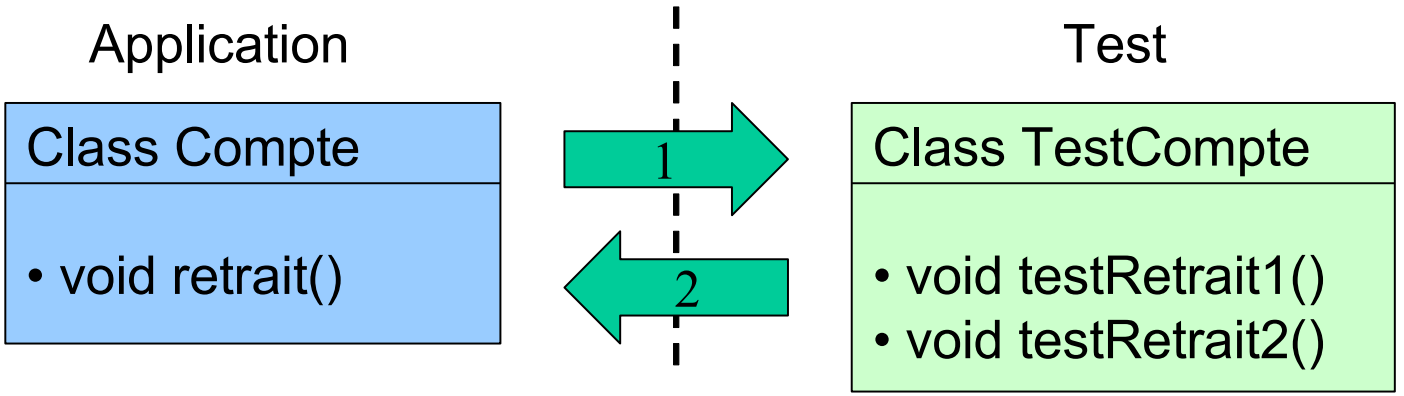
JUnit

- Framework de test Java
  - Cadre pour tester
  - Lancement automatique et rapide
  - Affichage élaboré

- Objectif: tester toute une application
  - Tracer les bugs
  - Éviter regression de code

- Reconnu sous Eclipse - Junit v4

Mecanique



- Principe
  - Séparer les tests de l'application
    - Une classe applicative – une classe de test
    - Un scénario d'une méthode = une méthode de test
  - Réunir et lancer toutes les classes de test

Démarche de test

- Méthode par Méthode
  - Ecrire le squelette de la classe applicative
  - Etudier les cas possibles
  - Ecrire une méthode de test par cas
  - Ecrire la méthode à executer
  - Tester la méthode
  - Jusque validation
    - modifier la méthode
  - Et seulement aprés passer à la méthode suivante

Une classe de Test (Junit 4)

- Classe de test
  - Nommée *TestClasse*
  - Débute par des import gérés par eclipse
- Chaque méthode de test
  - Par convention, commence par *testMethode*
  - Prealable par @Test
  - Structurée en trois morceaux
    - Préparation des données
    - Execution de ce qu'il y a a tester
    - Vérification des résultats

Vérification des résultats

- Objectif: vérifier que le resultat est celui attendu

- Méthodes fournies par Junit
  - assertEquals (String mess, int attendu, int obtenu)
  - assertEquals (String, Object, Object)
    - Utilise equals
  - assertEquals (String, Object[] , Object[])
    - Parcours tableaux
  - assertTrue (String, boolean)
  - fail (String)

Message et utilisation

- Message écrit au conditionnel,
  - "a devrait etre égal à 2"
  - "le solde devrait etre négatif"
- Permet de séparer
  - ce qui est attendu De ce qui est obtenu

- Lien avec debogger:
  - Test : detecter une erreur
  - Deboggeur: comprendre erreur
  - Mettre point arret dans test échoué

Contraintes java

- Conventions prog1
  - Javadoc
  - Nom de classe Majuscules
  - Nom de méthode minuscules
  - Nome de méthode = verbe infinitif

Exemple

- Classe compte
  - Polycopié
- Plein d'autres choses
  - Exceptions, ...

```java
class Compte {
  // peu importe le fonctionnement interne de la classe

  /**
   * constructeur à ecrire
   *
   * @param solde
   *            solde initial du compte
   */
  public Compte(int solde) {
  }

  /**
   * methode d'acces à completer retourne le solde
   *
   * @return solde du compte
   */
  public int getSolde() {
    return (0);
  }

  /**
   * methode qui retire un montant
   *
   * @param valeur
   *            du retrait souhaite
   * @return true si le retrait a été effectué
   */
  public boolean retirerMontant(int retrait) {
    return (false);
  }
}
```

```java
import static org.junit.Assert.*;
import org.junit.Test;

public class CompteTest {

  @Test
  public void testRetrait1() {
    // jeux de données
    Compte c = new Compte(500);
    // instruction de test
    boolean test = c.retirerMontant(200);
    // assertions
    assertTrue("le retrait devrait etre ok",test);
    assertEquals("le montant devrait etre débité", c.getSolde(), 300);
  }

  @Test
  public void testRetrait2() {
    // jeux de données
    Compte c = new Compte(500);
    // instruction de test
    boolean test = c.retirerMontant(600);
    // assertion
    assertTrue("le retrait devrait etre refusé, "test);
    assertEquals("le solde devrait etre inchangé", c.getSolde(), 500);
  }

}
```

Présentation sous Eclipse

- Classe Addition