Test Driven Development appliqué au test de classes auto-testables

–

Énoncé

Bruno Mermet

2011

Classes auto-testables & mise en oeuvre proposée

- Les classes auto-testables sont des classes qui incluent leurs méthodes de test unitaire.
- Ici
  - Classe auto-testable = classe implantant l'interface AutoTest (interface marqueur) ;
  - Méthode de test d'une méthode
    - Méthode renvoyant un booléen (true = test réussi, false = test raté)
    - Dont le nom est de la forme testMéthode ou méthode est une autre méthode de la classe
    - Méthode de classe ne prenant aucun argument
  - Méthode de test général
    - Méthode static boolean test()

Énoncé

- Écrire un programme qui demande le nom d'une classe. Le programme détermine alors si la classe est auto-testable et, dans ce cas, va :
  - Exécuter tous les tests
  - Afficher le résultat de chaque test sous la forme : méthode : réussi ou méthode : raté
  - Afficher le résumé ainsi :
    - % tests réussis
    - % méthodes testées

I. une classe est-elle auto-testable ?

Interface AutoTest

```
package autotestabletdd;
public interface AutoTest {
}
```

Cas de test 1

Une classe qui n'implante pas l'interface AutoTest n'est pas auto-testable

Cas de test 1

- Classe

```
package classesDeTest;
public class NonAutoTestable {
}
```

- Code de test

```
public void testIsAutoTestable1() {
AutoTestableTDD instance = new
AutoTestableTDD(classesDeTest.NonAutoTes
table.class);
boolean expResult = false;
boolean result = instance.isAutoTestable();
assertEquals(expResult, result);
}
```

- Code initial

```
public class AutoTestableTDD {
    public boolean isAutoTestable() {
        return false;
    }
}
```

Cas de test 2

Une classe qui implante directement l'interface est pas auto-testable

Cas de test 2

- Classe

```
package classesDeTest;
public class AutoTestableVide implements AutoTest {}
```

- Code de test

```
public void testIsAutoTestable1() {
AutoTestableTDD instance = new AutoTestableTDD(classesDeTest.AutoTestableVide.class);
boolean expResult = true;
boolean result = instance.isAutoTestable();
assertEquals(expResult, result);
}
```

Cas de test 3

Une classe qui implante l'interface AutoTest par héritage d'une classe implantant AutoTest est auto-testable

Cas de test 3

- Classe

```
public class AutoTestableParHeritage extends AutoTestableVide {
}
```

- Code de test

```
public void testIsAutoTestable1() {
    AutoTestableTDD instance = new
        AutoTestableTDD(classesDeTest.AutoTestableParHeritage.class);
    boolean expResult = true;
    boolean result = instance.isAutoTestable();
    assertEquals(expResult, result);
}
```

Cas de test 4

Une classe qui implante l'interface AutoTest par héritage indirect d'une classe implantant AutoTest est auto-testable

Cas de test 4

- Classe

```
public class AutoTestablePar2Heritages extends AutoTestableParHeritage {
}
```

- Code de test

```
public void testIsAutoTestable1() {
AutoTestableTDD instance = new
AutoTestableTDD(classesDeTest.AutoTestablePar2Heritages.class);
boolean expResult = true;
boolean result = instance.isAutoTestable();
assertEquals(expResult, result);
}
```

Cas de test 5

Une classe qui implante l'interface AutoTest par implantation d'une interface héritant d'AutoTest est auto-testable

Cas de test 5

- Interface

```
public interface SousInterfaceAutoTest extends AutoTest {
}
```

- Classe

```
public class AutoTestableParInterfaceIndirecte implements SousInterfaceAutoTest {
}
```

- Code de test

```
public void testIsAutoTestable1() {
AutoTestableTDD instance = new
AutoTestableTDD(classesDeTest.AutoTestableParInterfaceIndirecte.class);
boolean expResult = true;
boolean result = instance.isAutoTestable();
assertEquals(expResult, result);
}
```

II. une méthode est-elle une méthode de test ?

Cas de test 1

Une méthode statique de la forme boolean testMethode() ou methode est une méthode de la classe est une méthode de test

Cas de test 1

- Classe

```
package classesDeTest;
public class Methodes {
    public void methode() {}
    public static boolean testMethode() {return true;}
}
```

- Code de test

```
public void testIsMethodeDeTest1() throws
NoSuchMethodException {
    System.out.println("retourn non booleen");
    AutoTestableTDD instance = new
AutoTestableTDD(classesDeTest.Methodes.cl
ass);
    boolean expResult = false;
    boolean result =
instance.isMethodeDeTest(Methodes.class.get
Method("testMethode", null));
    assertEquals(expResult, result);
}
```

- Code initial

```
boolean isMethodeDeTest(Method methode)
{
    return true;
}
```

Cas de test 2

Une méthode qui ne retourne pas un booléen n'est pas une méthode de test.

Cas de test 2

- Classe

```
package classesDeTest;
public class Methodes {
    public void methode() {}
    public void methode1() {}
    public static boolean testMethode() {return true;}
    public static void testMethode1() {}
}
```

- Code de test

```
public void testIsMethodeDeTest2() throws NoSuchMethodException {
    System.out.println("retour non booléen");
    AutoTestableTDD instance = new AutoTestableTDD(classesDeTest.Methodes.class);
    boolean expResult = false;
    boolean result =
        instance.isMethodeStandard(classesDeTest.Methodes.class.getMethod("testMethode1", null));
    assertEquals(expResult, result);
}
```

Cas de test 2bis

Une méthode prenant des paramètres n'est pas une méthode de test

Cas de test 2bis

- Classe

```
package classesDeTest;
public class Methodes {
   [...]
   public static boolean testMethode(int i) {return true;}
}
```

- Code de test

```
   public void testIsMethodeDeTest2bis() throws NoSuchMethodException {
      System.out.println("paramètres");
      AutoTestableTDD instance = new AutoTestableTDD(classesDeTest.Methodes.class);
      boolean expResult = false;
      Class[] params = {int.class};
      boolean result =
instance.isMethodeDeTest(classesDeTest.Methodes.class.getMethod("testMethode", params));
      assertEquals(expResult, result);
   }
```

Cas de test 2ter

Une méthode d'instance n'est pas une méthode de test

Cas de test 2ter

- Classe

```
package classesDeTest;
public class Methodes {
    [...]
    public void methode2() {}
    public boolean testMethode2() {return true;}
}
```

- Code de test

```
public void testIsMethodeDeTest2ter() throws NoSuchMethodException {
    System.out.println("pas static");
    AutoTestableTDD instance = new AutoTestableTDD(classesDeTest.Methodes.class);
    boolean expResult = false;
    boolean result = instance.isMethodeDeTest(classesDeTest.Methodes.class.getMethod("testMethode2",
null));
    assertEquals(expResult, result);
}
```

Cas de test 3

Une méthode dont le nom ne commence pas par « test » n'est pas une méthode de test

Cas de test 3

- Classe

```
package classesDeTest;
public class Methodes {
    […]
    public static boolean methodeTest() {return true;}
}
```

- Code de test

```
public void testIsMethodeDeTest2() throws
NoSuchMethodException {
    AutoTestableTDD instance = new
AutoTestableTDD(classesDeTest.Methodes.class);
    boolean expResult = false;
    boolean result =
instance.isMethodeStandard(classesDeTest.Methode
s.class.getMethod("methodeTest", null));
    assertEquals(expResult, result);
}
```

Cas de test 4

Une méthode dont le caractère suivant le mot « test » n'est pas une majuscule n'est pas une méthode de test

Cas de test 4

- Classe

```
package classesDeTest;
public class Methodes {
    […]
    public static boolean testmethode() {return true;}
}
```

- Code de test

```
public void testIsMethodeDeTest2() throws NoSuchMethodException {
    AutoTestableTDD instance = new AutoTestableTDD(classesDeTest.Methodes.class);
    boolean expResult = false;
    boolean result = instance.isMethodeStandard(classesDeTest.Methodes.class.getMethod("testmethode",
null));
    assertEquals(expResult, result);
}
```

Cas de test 5

Et si on n'a rien derrière « test » ? Ça n'est pas une méthode de test (c'est la méthode de test général en l'occurrence).

Cas de test 5

- Classe

```
package classesDeTest;
public class Methodes {
    […]
    public static boolean test() {return true;}
}
```

- Code de test

```
public void testIsMethodeDeTest2() throws NoSuchMethodException {
    AutoTestableTDD instance = new AutoTestableTDD(classesDeTest.Methodes.class);
    boolean expResult = false;
    boolean result = instance.isMethodeStandard(classesDeTest.Methodes.class.getMethod("testmethode",
null));
    assertEquals(expResult, result);
}
```

Plantage !

Cas de test 6

La fin du nom de la méthode doit correspondre à une méthode existant dans la classe, avec le nom « majusculé ».

=> On a donc besoin de la liste des noms des méthodes de la classe avec majuscule au début. Pour accélérer la recherche, on classera ces noms de méthodes par ordre alphabétique. En cas de plusieurs méthodes de même nom, on ne stockera qu'une occurrence.

II.1 Liste des noms de méthodes « majusculés »

II.1.a « Majusculisation » d'un chaîne

Cas de test 1

La « majusculisation » de « AbCdE » donne « AbCdE »

Cas de test 1

- Code de test

```
  public void testMajusculiseInitiale1() {
      System.out.println("Majusculisation initiale
majuscule");
      String donnee = "AbCdE";
      String expResult = "AbCdE";
      String result =
AutoTestableTDD.majusculiseInitialeChaineNonVi
de(donnee);
      assertEquals(expResult,result);
  }
```

- Code initial

```
static String
majusculiseInitialeChaineNonVide(String s)
{
   return s;
}
```

Cas de test 2

La « majusculisation » de « abCdE » donne « AbCdE »

Cas de test 2

- Code de test

```
public void testMajusculiseInitiale2() {
    System.out.println("Majusculisation initiale majuscule");
    String donnee = "abCdE";
    String expResult = "AbCdE";
    String result = AutoTestableTDD.majusculiseInitialeChaineNonVide(donnee);
    assertEquals(expResult,result);
}
```

Fin II.1.a « Majusculisation » d'un chaîne

Suite II.1. Liste alphabétique des noms de méthodes « majusculés »

Cas de test 1

La liste des noms de méthodes de la classe Methodes, majusculés et par ordre alphabétique, est :

- Methode
- Methode1
- Methode2
- MethodeTest
- TestMethode
- TestMethode1
- TestMethode2
- Test

Cas de test 1

- Code de test

```
public void testListeNomsMethodes1() {
    System.out.println("liste des noms des
méthodes");
    AutoTestableTDD instance = new
AutoTestableTDD(classesDeTest.Methodes.class);
    String[] expResult =
{"Methode","Methode1","MethodeTest","Test","Te
stMethode","TestMethode1","Testmethode"};
    String[] result = instance.listeNomsMethodes();
    assertArrayEquals(expResult,result);
}
```

- Code initial

```
String[] listeNomsMethodesAlphaMaj() {
    Method[] methodes =
classeCible.getMethods();
    Set<String> listeNoms = new
TreeSet<String>();
    for (Method m : methodes) {
        ListeNoms.
add(majusculiseInitialeChaineNonVide(m.ge
tName()));
    }
    return listeNoms.toArray(new String[0]);
}
```

On a oublié les méthodes provenant de l'héritage

Modification du cas de test

- Code de test

```
public void testListeNomsMethodes1() {
    System.out.println("liste des noms des méthodes");
    AutoTestableTDD instance = new AutoTestableTDD(classesDeTest.Methodes.class);
    String[] expResult =
{"Equals","GetClass","HashCode","Methode","Methode1", "Methode2","MethodeTest","Notify","Notify
All","Test","TestMethode","TestMethode1","TestMethode2","Testmethode","ToString","Wait"};
    String[] result = instance.listeNomsMethodes();
    assertArrayEquals(expResult,result);
}
```

Fin II.1. Liste alphabétique des noms de méthodes « majusculés »

Suite « cas de test 6 du II.1 » :

La fin du nom de la méthode doit correspondre à une méthode existant dans la classe, avec le nom « majusculé ».

Cas de test 6

- Classe Methode

Retour à la version précédente

- Classe Methode2

```
package classesDeTest;
public class Methodes2 extends Methodes {
    public static boolean testNomAutre() {return true;}
}
```

- Code de test

```
public void testIsMethodeDeTest2() throws NoSuchMethodException {
    AutoTestableTDD instance = new
AutoTestableTDD(classesDeTest.Methodes2.class);
    boolean expResult = false;
    boolean result =
instance.isMethodeStandard(classesDeTest.Methodes.class.getMethod("testA
utreNom", null));
    assertEquals(expResult, result);
}
```

III. une méthode est-elle la méthode de test général ?

Cas de test 1

Une méthode quelconque n'est en général pas une méthode de test général. Il en est de même d'une méthode de test standard.

Cas de test 1

- Code de test

```
public void testIsTestGeneral1() throws
NoSuchMethodException {
    AutoTestableTDD instance = new
AutoTestableTDD(classesDeTest.Methodes.class);
    boolean expResult = false;
    boolean result =
instance.isMethodeStandard(classesDeTest.Methode
s.class.getMethod("testMethode", null));
    assertEquals(expResult, result);
        result =
instance.isTestGeneral(classesDeTest.Methodes.clas
s.getMethod("methode", null));
        assertEquals(expResult, result);

}
```

- Code initial

```
public boolean isTestGeneral(Method m) {
    return false;
}
```

Cas de test 2

Une méthode appelée *test*, statique, renvoyant un booléen, et ne prenant aucun paramètre, est une méthode de test général.

Cas de test 2

- Code de test

```
public void testIsTestGeneral2() throws NoSuchMethodException {
    System.out.println("méthode de test générale");
    AutoTestableTDD instance = new AutoTestableTDD(classesDeTest.Methodes.class);
    boolean expResult = true;
    boolean result = instance.isTestGeneral(classesDeTest.Methodes.class.getMethod("test", null));
    assertEquals(expResult, result);
}
```

IV. Tri des méthodes

Cas de test 1 (raccourci)

- On va disposer d'une classe (interne à AutoTestableTDD) qui va stocker le partitionnement des différentes méthodes
- On va tester cela sur la classe Methodes2

Cas de test 1 : code de test

```
public void testPartitionMethode() throws NoSuchMethodException {
    AutoTestableTDD instance = new AutoTestableTDD(classesDeTest.Methodes2.class);
    PartitionMethodes partition = instance.getPartition();

    Method expResult1 = classesDeTest.Methodes2.class.getMethod("test",null);
    assertEquals(expResult1, partition.getTestGeneral());

    Method m1 = classesDeTest.Methodes2.class.getMethod("testMethode", null);
    Method[] expResult2 = new Method[1];
    expResult2[0] = m1;
    assertArrayEquals(expResult2,partition. getMethodesDeTestAlpha());

    String[] expResult3 = {"equals", "getClass", "hashCode", "methode", "methode1",
"methode2", "methodeTest", "notify", "notifyAll", "testAutreNom", "testMethode",
"testMethode1", "testMethode2", "testmethode", "toString", "wait"};
    assertArrayEquals(expResult3,partition.getMethodesStandardsAlpha());
}
```

V. Statistiques

V.1. aTestGeneral

Cas de test 1

Une classe avec une méthode static boolean void test(), directement ou par héritage, a une méthode de test général

Cas de test 1

- Code de test

```
public void testATestGeneral1() throws
NoSuchMethodException {
    System.out.println("méthode de test générale");
    AutoTestableTDD instance = new
AutoTestableTDD(classesDeTest.Methodes.class);
    boolean expResult = true;
    boolean result = instance.aTestGeneral();
    assertEquals(expResult, result);
}
```

- Code

```
public boolean aTestGeneral() {
    return partition.getTestGeneral() != null;
}
```

Cas de test 2

Une classe sans méthode static boolean void test(), n'a pas une méthode de test général

Cas de test 2

- Classe de test

```
public class Methodes3 implements AutoTest {
    public void methode() {}
    public static boolean testMethode() {return true;}
}
```

- Code de test

```
public void testATestGeneral2() throws NoSuchMethodException {
    System.out.println("n'a pas méthode de test générale");
    AutoTestableTDD instance = new AutoTestableTDD(classesDeTest.Methodes3.class);
    boolean expResult = false;
    boolean result = instance.aTestGeneral();
    assertEquals(expResult, result);
}
```

V.2. Pourcentage méthodes testées

Cas de test 1

On teste une partie des méthodes :

Sur « Methodes », on a une méthode de test pour 15 méthodes standards (ne pas oublier celles dont on hérite), soit 7%

Cas de test 1

- Code de test

```
public void testPourcentageTests() throws
NoSuchMethodException {
    System.out.println("une partie testée");
    AutoTestableTDD instance = new
AutoTestableTDD(classesDeTest.Methodes.class);
    int expResult = 7;
    int result = instance.pourcentageTests();
    assertEquals(expResult, result);
}
```

- Code initial

```
public int pourcentageTests() {
    return 7;
}
```

Cas de test 2

On teste toutes les méthodes :

Créer une classe avec des méthodes de test pour toutes les méthode de Object.

Cas de test 2

- Classe de test

```
public class Methodes4 implements AutoTest {
    public boolean testEquals() {return true;}
    public boolean testGetClass() {return true;}
    public boolean testHashCode() {return true;}
    public boolean testNotify() {return true;}
    public boolean testNotifyAll() {return true;}
    public boolean testToString() {return true;}
    public boolean testWait() {return true;}
}
```

- Code de test

```
public void testPourcentageTests2() throws
NoSuchMethodException {
    System.out.println("tout est testée");
    AutoTestableTDD instance = new
AutoTestableTDD(classesDeTest.Methodes4.class);
    int expResult = 100;
    int result = instance.pourcentageTests();
    assertEquals(expResult, result);
}
```

Cas de test 3

On ne teste aucune méthode

Cas de test 3

- Classe de test

```
public class Methodes5 implements AutoTest {
    public void methode() {}

}
```

- Code de test

```
public void testPourcentageTests2() throws NoSuchMethodException {
    System.out.println("tout est testée");
    AutoTestableTDD instance = new AutoTestableTDD(classesDeTest.Methodes4.class);
    int expResult = 0;
    int result = instance.pourcentageTests();
    assertEquals(expResult, result);
}
```

Cas de test 4

Il n'y a aucune méthode à tester

=> ne peut se produire car on hérite forcément directement ou indirectement de *Object*, qui définit des méthodes standards

VI. Execution des méthodes de test

Cas de test 1

Exécution d'une méthode de test. Si le résultat est vrai, on renvoie vrai, et inversement

Cas de test 1

- Classe de test

```
public class Methodes6 implements AutoTest {
    public void methode1() {}
    public static boolean testMethode1() {return true;}
    public void methode2() {}
    public static boolean testMethode2() {return false;}
}
```

- Code de test

```
public void testExecuteMethode1() throws
NoSuchMethodException {
    AutoTestableTDD instance = new
AutoTestableTDD(classesDeTest.Methodes6.class);
    boolean expResult = true;
    boolean result =
instance.executeTest(classesDeTest.Methodes6.class.getMetho
d("testMethode1", null));
    assertEquals(expResult, result);
    expResult = false;
    result =
instance.executeTest(classesDeTest.Methodes6.class.getMetho
d("testMethode2", null));
    assertEquals(expResult, result);
}
```

- Code

```
public boolean executeTest(Method m) {
    try {
        return (boolean) m.invoke(null);
    } catch(Exception iae) {
        System.err.println("problème à
l'appel d'une méthode de test");
        System.exit(1);
    }
    return false;
}
```

Cas de test 2

Exécution de toutes les méthodes de test

=> Utiliser une classe avec 2 méthodes de test, l'une réussissant, l'autre échouant

Cas de test 2

- Code de test

```
public void testExecuteTout() {
    System.out.println("execution de toutes les
méthodes");
    AutoTestableTDD instance = new
AutoTestableTDD(classesDeTest.Methodes6.
class);
    AutoTestableTDD.RapportDeTest rapport
= instance.getRapport();
    int pourcentageAttendu = 50;
    int pourcentageReel =
rapport.getPourcentageIndividuel();

assertEquals(pourcentageAttendu,pourcentage
Reel);
    String sortieAttendue = "testMethode1
réussi\n"+
            "testMethode2 raté\n";
    String sortieReelle =
rapport.getRapportIndividuel();
    assertEquals(sortieAttendue, sortieReelle);
}
```

- Code

```
private RapportDeTest rapport;
public AutoTestableTDD(Class c) {
    classeCible = c; partitionne(); rapport = new RapportDeTest();
    executeTousLesTestsIndividuels();}
public class RapportDeTest {
    private int nbTestsReussis;private int nbTests;
    private ArrayList<String> rapportsIndividuels;
    public RapportDeTest() {NbTestsReussis = 0; nbTests = 0;
        rapportsIndividuels = new ArrayList<String>();}
    public void ajoutTestIndividuel(String nomMethode, boolean
reussi) {nbTests++;
        if (reussi) {nbTestsReussis++;
            rapportsIndividuels.add(nomMethode + " réussi");}
        else {apportsIndividuels.add(nomMethode + " raté");}
    }
    public int getPourcentageIndividuel() {
        return (int) Math.round(nbTestsReussis/(double) nbTests*100);}
    public String getRapportIndividuel() {
        StringBuilder sortie = new StringBuilder();
        for (String s : rapportsIndividuels) {sortie.append(s+"\n");}
        return sortie.toString();
}}
```

Refactorisation
Classe RapportDeTest sortie

- Code de test

```
public void testExecuteTout() {
    System.out.println("execution de toutes les méthodes");
    AutoTestableTDD instance = new AutoTestableTDD(classesDeTest.Methodes6.class);
    RapportDeTest rapport = instance.getRapport();
    int pourcentageAttendu = 50;
    int pourcentageReel = rapport.getPourcentageIndividuel();

assertEquals(pourcentageAttendu,pourcentageReel);
    String sortieAttendue = "testMethode1 réussi\n"+
            "testMethode2 raté\n";
    String sortieReelle = rapport.getRapportIndividuel();
    assertEquals(sortieAttendue, sortieReelle);
}
```

- Code

```
public class RapportDeTest {
private int nbTestsReussis;private int nbTests;
private ArrayList<String> rapportsIndividuels;
public RapportDeTest() {
    nbTestsReussis = 0;nbTests = 0;
    rapportsIndividuels = new ArrayList<String>();
}
public void ajoutTestIndividuel(String nomMethode, boolean reussi) {
    nbTests++;
    if (reussi) {nbTestsReussis++;
        rapportsIndividuels.add(nomMethode + " réussi");
    } else {rapportsIndividuels.add(nomMethode + " raté");}
}
public int getPourcentageIndividuel() {
    return (int) Math.round(nbTestsReussis / (double) nbTests * 100);
}
public String getRapportIndividuel() {
    StringBuilder sortie = new StringBuilder();
    for (String s : rapportsIndividuels) {sortie.append(s + "\n");}
        return sortie.toString();
}}
```

Cas de test 3

Cas d'une classe non auto-Testable

Cas de test 3

- Code de test

```
public void testExecuteTout() {
    System.out.println("execution de toutes les
méthodes");
    AutoTestableTDD instance = new
AutoTestableTDD(classesDeTest.Methodes6.
class);
    AutoTestableTDD.RapportDeTest rapport
= instance.getRapport();
    int pourcentageAttendu = 50;
    int pourcentageReel =
rapport.getPourcentageIndividuel();

assertEquals(pourcentageAttendu,pourcentage
Reel);
    String sortieAttendue = "testMethode1
réussi\n"+
            "testMethode2 raté\n";
    String sortieReelle =
rapport.getRapportIndividuel();
    assertEquals(sortieAttendue, sortieReelle);
}
```

- Code

```
private RapportDeTest rapport;
public AutoTestableTDD(Class c) {
    classeCible = c; partitionne(); rapport = new RapportDeTest();
    executeTousLesTestsIndividuels();}
public class RapportDeTest {
    private int nbTestsReussis;private int nbTests;
    private ArrayList<String> rapportsIndividuels;
    public RapportDeTest() {NbTestsReussis = 0; nbTests = 0;
        rapportsIndividuels = new ArrayList<String>();}
    public void ajoutTestIndividuel(String nomMethode, boolean
reussi) {nbTests++;
        if (reussi) {nbTestsReussis++;
            rapportsIndividuels.add(nomMethode + " réussi");}
        else {apportsIndividuels.add(nomMethode + " raté");}
    }
    public int getPourcentageIndividuel() {
        return (int) Math.round(nbTestsReussis/(double) nbTests*100);}
    public String getRapportIndividuel() {
        StringBuilder sortie = new StringBuilder();
        for (String s : rapportsIndividuels) {sortie.append(s+"\n");}
        return sortie.toString();
}}
```

VII. Bilan

Cas de test 1

Une classe non auto-testable doit avoir un bilan « Classe non auto-testable ».

Cas de test 1

- Code de test

```
public void testBilan1() {
    System.out.println("Bilan d'une classe non auto-testable");
    AutoTestableTDD instance = new AutoTestableTDD(classesDeTest.NonAutoTestable.class);
    RapportDeTest rapport = instance.getRapport();
    String sortieAttendue = "Classe non auto-testable";
    String sortieReelle = rapport.getBilan();
    assertEquals(sortieAttendue, sortieReelle);
}
```

- Code initial

```
public class RapportDeTest {
    [...]
    public String getBilan() {
        return getRapportIndividuel();
    }
}
```

Cas de test 2

Une classe auto-testable a un bilan autre que « Classe non auto-testable ».

Cas de test 2

- Code de test

```
public void testBilan2() {
    System.out.println("Bilan d'une classe auto-testable");
    AutoTestableTDD instance = new AutoTestableTDD(classesDeTest.Methodes.class);
    RapportDeTest rapport = instance.getRapport();
    String sortieNonAttendue = "Classe non auto-testable";
    String sortieReelle = rapport.getBilan();
    assertFalse(sortieNonAttendue.equals(sortieReelle));
}
```

Cas de test 3

Une classe auto-testable sans méthode de test individuelle doit être considérée comme ayant 100% des tests réussis.

Cas de test 3

- Code de test

```
public void testBilan3() {
    System.out.println("Bilan d'une classe auto-testable");
    AutoTestableTDD instance = new AutoTestableTDD(classesDeTest.Methodes5.class);
    RapportDeTest rapport = instance.getRapport();
    int pourcentageAttendu = 100;
    int pourcentageReel = rapport.getPourcentageIndividuel();
    assertEquals(pourcentageAttendu,pourcentageReel);
}
```

Cas de test 4

Le rapport doit contenir le pourcentage des méthodes testées

Cas de test 4

- Code de test

```
public void testBilan4() {
    System.out.println("Bilan d'une classe auto-testable :
bilan méthodes testées");
    AutoTestableTDD instance = new
AutoTestableTDD(classesDeTest.Methodes.class);
    RapportDeTest rapport = instance.getRapport();
    int pourcentageAttendu = 7;
    int pourcentageReel = rapport.getPourcentageTeste();
        assertEquals(pourcentageAttendu,pourcentageReel);
}
```

- RapportIndividuelNonAutoTestable

```
public RapportIndividuelNonAutoTestable() {
    super(0);
}
```

- RapportDeTest

```
private int pourcentageTeste;
public RapportDeTest(int teste) {
    nbTestsReussis = 0;
    nbTests = 0;
    pourcentageTeste = teste;
    rapportsIndividuels = new ArrayList<String>();
}
public int getPourcentageTeste() {
    return pourcentageTeste;
}
```

- AutoTestableTDD

```
public AutoTestableTDD(Class c) {
    classeCible = c;
    if (isAutoTestable()) {partitionne();
        rapport = new RapportDeTest(pourcentageTests());
        executeTousLesTestsIndividuels();
    } else {
        rapport = new RapportIndividuelNonAutoTestable();
}}
```

Cas de test 5

Le rapport doit contenir le résultat de la méthode de test général

- 5.1 La méthode existe et renvoie vrai
- 5.2 La méthode existe et renvoie faux
- 5.3 La méthode n'existe pas

Cas de test 5

- Code de test

```
public void testBilan5() {
        System.out.println("Bilan test général");
        AutoTestableTDD instance = new
AutoTestableTDD(classesDeTest.Methodes.class);
        RapportDeTest rapport = instance.getRapport();
        String resultatAttendu = "Test général réussi";
        String resultatReel =
rapport.getResultatTestGeneral();
        assertEquals(resultatAttendu,resultatReel);
        instance = new
AutoTestableTDD(classesDeTest.Methodes3.class);
        rapport = instance.getRapport();
        resultatAttendu = "Pas de test général";
        resultatReel = rapport.getResultatTestGeneral();
        assertEquals(resultatAttendu,resultatReel);
        instance = new
AutoTestableTDD(classesDeTest.Methodes7.class);
        rapport = instance.getRapport();
        resultatAttendu = "Test général raté";
        resultatReel = rapport.getResultatTestGeneral();
        assertEquals(resultatAttendu,resultatReel);
    }
```

- RapportDeTest

```
private int pourcentageTeste;
public void setTestGeneral(boolean reussite) {
    if (reussite) {testGeneral = "Test général réussi";}
    else {testGeneral = "Test général raté";}
}
public String getResultatTestGeneral() {return testGeneral;}
```

- AutoTestableTDD

```
public AutoTestableTDD(Class c) {
    classeCible = c;
    if (isAutoTestable()) {
        partitionne();
        rapport = new RapportDeTest(pourcentageTests());
        if (aTestGeneral()) {
rapport.setTestGeneral(executeTest(partition.getTestGeneral()))
;
        }
        executeTousLesTestsIndividuels();
    }
    else {rapport = new RapportIndividuelNonAutoTestable();}
}
```