Aix*Marseille université
Initiative d'excellence
Année 2018-19

Domaine Sciences et Technologies
Licence informatique

Programmation 2 : TD 5
Code UE : SIN3U02

Tests unitaires

1 Rappels sur les tests

Pour faire des tests en Java, on utilise les assertions de Junit :

- `assertTrue(boolean condition)` : vérifie que `condition` est vraie.
- `assertFalse(boolean condition)` : vérifie que `condition` est faux.
- `assertEquals(expected, actual)` : vérifie que `expected` est à `actual` (égal : `equals` pour les objets et `==` pour les types primitifs).
- `assertNotEquals(expected, actual)` : vérifie que `expected` **n'**est **pas** à `actual` (égal : `equals` pour les objets et `==` pour les types primitifs).
- `assertEquals(double expected, double actual, double delta)` : vérifie que $|expected - actual| \leq delta$
- `assertNull(Object object)` : vérifie que la référence est `null`
- `assertNotNull(Object object)` : vérifie que la référence **n'**est **pas** `null`
- `assertSame(Object expected, Object actual)` : vérifie que les deux objets sont les mêmes (même référence).
- `assertArrayEquals(Object[] expected, Object[] actual)` : vérifie si les deux tableaux contiennent les même éléments dans le même ordre.

Pour vérifier le bon fonctionnement de la méthode `equals` de `Point` (cf TD précédent), on peut écrire le code suivant :

```
class PointTest {
    @Test
    void testEqualsPoint(){
        int x = 10;
        int y = -15;
        Point p1 = new Point(x,y);
        Point p2 = new Point(x,y);
        Point p3 = new Point(x, -y);
        Point p4 = new Point(-x, y);
        String s = new String();
        assertEquals(p1, p2);
        assertNotEquals(p1, p3);
        assertNotEquals(p2, p4);
        assertNotEquals(p1, s);
    }
```

2 Test d'une classe d'entier

On nous fournit une classe `BigInteger` permettant de manipuler des entiers relatifs en précision arbitraire. Les `BigInteger` sont immutables : aucune méthode ne modifie `this`, ni ses arguments.

La classe `BigInteger` contient :

- des constantes :
  - `static BigInteger ONE` : The `BigInteger` constant one.
  - `static BigInteger TEN` : The `BigInteger` constant ten.
  - `static BigInteger TWO` : The `BigInteger` constant two.

    - `static BigInteger ZERO` : The `BigInteger` constant zero.
- un constructeur :
    - `public BigInteger(String val)` : Translates the decimal `String` representation of a `BigInteger` into a `BigInteger`.
- des méthodes :
    - `boolean equals(Object o)` : Compares this `BigInteger` with the specified `Object` for equality.
    - `BigInteger add(BigInteger val)` : Returns a `BigInteger` whose value is `(this + val)`.
    - `BigInteger negate()` : Returns a `BigInteger` whose value is `(-this)`.
    - `BigInteger subtract(BigInteger val)` : Returns a `BigInteger` whose value is `(this - val)`.
    - `BigInteger multiply(BigInteger val)` : Returns a `BigInteger` whose value is `(this * val)`.
    - `BigInteger divide(BigInt val)` : Returns a `BigInteger` whose value is `(this / val)`.
    - `BigInteger[] divideAndRemainder(BigInteger val)` : Returns an array of two `BigIntegers` containing `(this / val)` followed by `(this % val)`.

Écrire des tests unitaires pour chacune des méthodes.

3 Test d'une classe de file

Les files (ou *queue*) implémentent des séquences linéaires d'objets. On peut insérer (`offer`) ou retirer (`poll`) des éléments de la file. L'élément retiré est toujours le plus ancien élément de la file toujours présent dans la file (appelé *head of the queue*), c'est-à-dire celui qu'on a inséré en premier parmi tous les éléments de la file. La classe `Queue` contient :

- des constructeurs :
    - `Queue()` Creates an empty `Queue`
    - `Queue(int[] array)` : Creates a `Queue` initially containing the elements of the given `array`, added in traversal order of the array.
- des méthodes :
    - `boolean isEmpty()` : Returns `true` if this queue contains no elements.
    - `void offer(int element)` : Inserts the specified `element` into this queue.
    - `int poll()` : Retrieves and removes the head of this queue, or returns `-1` if this queue is empty.
    - `int length()` : Returns the number of elements in this queue.

Écrire des tests unitaires pour chacun des constructeurs et des méthodes.