Tests unitaires

JUnit

`list.size()` ***devrait*** renvoyer la taille de la liste.

```java
/** MyList represents a list of integers. */
public class MyList { … }

public class Main {

    public void main(String[] args) {
        MyList list = new MyList();
        list.add(5);
        list.add(23);
        System.out.println(list.size());
        // Will probably print "2" (?)
    }
}
```

L'étape **Arrange** configure le scénario de test.

```java
@Test
public void testMyListSize() {
    // Arrange
    MyList list = new MyList();
    list.add(1);




}
```

Knight, Andy. « Arrange-Act-Assert: A Pattern for Writing Good Tests ». *Automation Panda* (blog), 7 juillet 2020.

L'étape **Act** couvre la chose principale à tester.

```java
@Test
public void testMyListSize() {
    // Arrange
    MyList list = new MyList();
    list.add(1);

    // Act
    int size = list.size();



}
```

Knight, Andy. « Arrange-Act-Assert: A Pattern for Writing Good Tests ». *Automation Panda* (blog), 7 juillet 2020.

L'étape **Assert** vérifie si le résultat est correct ou non.

```java
@Test
public void testMyListSize() {
    // Arrange
    MyList list = new MyList();
    list.add(1);

    // Act
    int size = list.size();

    // Assert
    assertTrue(size == 1);
}
```

Knight, Andy. « Arrange-Act-Assert: A Pattern for Writing Good Tests ». *Automation Panda* (blog), 7 juillet 2020.

JUnit

```
public class Test2048 {

    @Test
    public void testSomething() {
        …
    }

    @Test
    public void testSomethingElse() {
        …
    }
}
```

```
assertTrue(text.startsWith("bonjour"));
```

```
assertTrue(a == b);
```

```
assertTrue(list1.equals(list2));
```

```
assertTrue(Arrays.equals(array1, array2));
```

```
public class Value {

    private int val;

    public Value(int initialValue) {
        val = initialValue;
    }

    public int getValue() {
        return val;
    }

    public void add(Value that) {
        that.val += val;
    }
}
```

```java
public class TestValue {

    @Test
    public void testAdd() {
        // Arrange
        Value v1 = new Value(3);
        Value v2 = new Value(10);

        // Act
        v1.add(v2);

        // Assert
        assertTrue(v1.getValue() == 13);
        assertTrue(v2.getValue() == 10);
    }
}
```

```java
public class Value {

    private int val;

    public Value(int initialValue) {
        val = initialValue;
    }

    public int getValue() {
        return val;
    }

    public void add(Value that) {
        that.val += val;
    }
}
```

```java
public class TestValue {

    @Test
    public void testAdd() {
        // Arrange
        Value v1 = new Value(3);
        Value v2 = new Value(10);

        // Act
        v1.add(v2);

        // Assert
        assertTrue(v1.getValue() == 13);
        assertTrue(v2.getValue() == 10);
    }
}
```

```java
public class Value {

    private int val;

    public Value(int initialValue) {
        val = initialValue;
    }

    public int getValue() {
        return val;
    }

    public void add(Value that) {
        that.val += this.val;
    }
}
```

```java
public class TestValue {

    @Test
    public void testAdd() {
        // Arrange
        Value v1 = new Value(3);
        Value v2 = new Value(10);

        // Act
        v1.add(v2);

        // Assert
        assertTrue(v1.getValue() == 13);
        assertTrue(v2.getValue() == 10);
    }
}
```

```java
public class Value {

    private int val;

    public Value(int initialValue) {
        val = initialValue;
    }

    public int getValue() {
        return val;
    }

    public void add(Value that) {
        this.val += that.val;
    }
}
```

```java
public class TestValue {

    @Test
    public void testAdd() {
        // Arrange
        Value v1 = new Value(3);
        Value v2 = new Value(10);

        // Act
        v1.add(v2);

        // Assert
        assertTrue(v1.getValue() == 13);
        assertTrue(v2.getValue() == 10);
    }
}
```