Les tests unitaires

http://alexandre-mesle.com

8 novembre 2022

```
package testsUnitaires;

public interface Puissance
{
        /**
         * Retourne x + 1. x quelconque.
         */

        public int succ(int x);

        /**
         * Retourne x − 1. x quelconque.
         */

        public int pred(int x);
```

```
package testsUnitaires;

public class ImplementationPuissance implements Puissance
{
        public int succ(int x)
        {
                return x + 1;
        }

        public int pred(int x)
        {
                return -succ(-x);
        }

        public int somme(int a, int b)
```

```
public int produit(int a, int b)
{
        if (b == 0)
                return 0;
        if (b > 0)
                return somme(produit(a, pred(b)), a);
        else
                return somme(produit(a, succ(b)), -a)
                    ;
}

public int puissance(int base, int exp)
{
        if (exp == 0)
                return 1;
```

Comment s'assurer que ce code fonctionne correctement ?

Il existe en Java (et en programmation en général) plusieurs façons d'effectuer des tests.

- Exécuter le programme et vérifier s'il se comporte conformément à ce qui est attendu.
- Tester les fonctions une par une en affichant les valeurs qu'elles retournent.
- Appeler les fonctions en comparant automatiquement les résultats retournés aux résultats attendus.
- Utiliser JUnit.

Il existe en Java (et en programmation en général) plusieurs façons d'effectuer des tests.

- Exécuter le programme et vérifier s'il se comporte conformément à ce qui est attendu.
- Tester les fonctions une par une en affichant les valeurs qu'elles retournent.
- Appeler les fonctions en comparant automatiquement les résultats retournés aux résultats attendus.
- Utiliser JUnit.

Il existe en Java (et en programmation en général) plusieurs façons d'effectuer des tests.

- Exécuter le programme et vérifier s'il se comporte conformément à ce qui est attendu.
- Tester les fonctions une par une en affichant les valeurs qu'elles retournent.
- Appeler les fonctions en comparant automatiquement les résultats retournés aux résultats attendus.
- Utiliser JUnit.

Il existe en Java (et en programmation en général) plusieurs façons d'effectuer des tests.

- Exécuter le programme et vérifier s'il se comporte conformément à ce qui est attendu.
- Tester les fonctions une par une en affichant les valeurs qu'elles retournent.
- Appeler les fonctions en comparant automatiquement les résultats retournés aux résultats attendus.
- Utiliser JUnit.

```
package testsUnitaires;

public class TestBourrin
{
        public static void main(String[] args)
        {
                Puissance p = new
                    ImplementationPuissance();
                System.out.println(p.puissance(2, 10));
        }
}
```

```
package testsUnitaires;

public class TestFonctions
{
        public static void main(String[] args)
        {
                Puissance p = new
                    ImplementationPuissance();
                System.out.println("2 = " + p.succ(1));
                System.out.println("5 = " + p.pred(6));
                System.out.println("2 = " + p.somme(2, 0));
                System.out.println("9 = " + p.somme(3, 6));
                System.out.println("10 = " + p.somme(13,
                    -3));
                System.out.println("2 = " + p.produit(2, 1))
```

```
package testsUnitaires;

public class TestFonctionsAutomatique
{
        private boolean ok = true;

        private void teste(String fonction, int obtenu, int
            attendu)
        {
                if (attendu == obtenu)
                        System.out.println("test OK");
                else
                {
                        System.out.println("test " +
                            fonction + " échoué : " + obtenu
```

```
public boolean teste(Puissance p)
{
        ok = true;
        try
        {
                teste("succ", p.succ(1), 2);
                teste("pred", p.pred(6), 5);
                teste("somme", p.somme(2, 0), 2);
                teste("somme", p.somme(3, 6), 9);
                teste("somme", p.somme(13, -3), 10);
                teste("produit", p.produit(2, 1), 2);
                teste("produit", p.produit(3, 6), 18);
                teste("produit", p.produit(-3, 6),
                    -18);
                teste("produit", p.produit(3, -6),
```

```
        public static void main(String[] args)
        {
                Puissance p = new
                    ImplementationPuissance();
                TestFonctionsAutomatique t = new
                    TestFonctionsAutomatique();
                t.teste(p);
        }
}
```

```
package testsUnitaires;

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

public class TestsJUnit
{
        private Puissance p = new ImplementationPuissance
            ();

        @Test
        public void testSucc()
        {
                assertEquals("test de la fonction successeur",
                    2, p.succ(1));
```

```
        assertEquals("somme", 2, p.somme(0, 2));
        assertEquals("somme", 9, p.somme(3, 6));
        assertEquals("somme", 10, p.somme(13, -3));
        assertEquals("somme", -10, p.somme(-13, 3))
            ;
        assertEquals("somme", -16, p.somme(-13,
            -3));
}

@Test
public void testProduit()
{
        assertEquals("produit", 0, p.produit(2, 0));
        assertEquals("produit", 0, p.produit(0, 2));
        assertEquals("produit", 2, p.produit(2, 1));
```

```
package testsUnitaires;

import java.io.IOException;
import java.util.logging.FileHandler;
import java.util.logging.Logger;
import java.util.logging.SimpleFormatter;

public class PuissanceAvecLog extends
    ImplementationPuissance
{
        private final static Logger LOGGER = Logger.
            getLogger(PuissanceAvecLog.class
                    .getName());
        static
        {
```

```
public int puissance(int base, int exp)
{
        String str = "Call to puissance : " + base + "
            ^" + exp + " = ";
        if (exp == 0)
                str += 1;
        else if ((exp & 1) == 0)
                str += "" + base * base + "^" + (
                    exp >> 1);
        else
                str += "" + base + "*(" + base + "
                    ^" + (exp - 1) + ")";
        LOGGER.info(str);
        return super.puissance(base, exp);
}
```