Java Essentials

Hoofdstuk 4:

Unittesten met JUnit

DE HOGESCHOOL
MET HET NETWERK

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook

Wat is unittesten?

- Unittesten is een manier om softwaremodulen of stukjes broncode (**units**) afzonderlijk te **testen**. Bij unittesten zal voor iedere **unit** een of meerdere **tests** ontwikkeld worden.
- Een **unit** is voor ons een methode in een klasse.

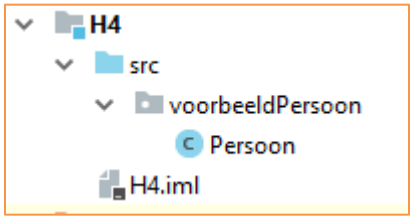
Waarom schrijf ik unit testen?

- Minder fouten in je code
- Je durft je code aan te passen en beter leesbaar te maken.
- Je denkt meer na over je klasse en de implementatie van de methoden.
- Bron: http://www.onjava.com/pub/a/onjava/2003/04/02/javaxpckbk.html

Wat is JUnit?

- JUnit is een open source framework ontwikkeld om unit testen te schrijven en uit te voeren in Java.

Een voorbeeld



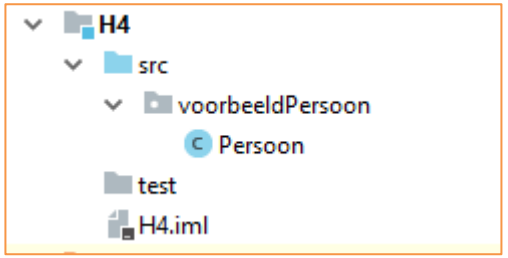

```
public class Persoon {
    private String naam;
    private String voornaam;

    public Persoon(String naam, String voornaam) {
        this.naam = naam;
        this.voornaam = voornaam;
    }

    public String getVolledigeNaam() {
        return null;
    }
}
```

Een test

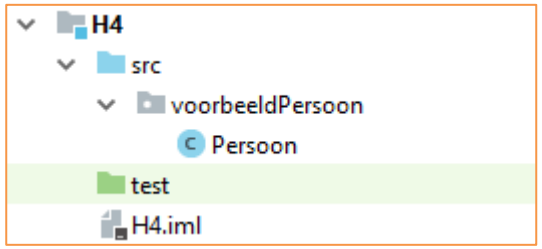
```
import …

public class PersoonTest {

    @Test
    public void testGetVolledigeNaam( ) {
        Persoon p = new Persoon("Flater", "Guust");
        assertEquals("Guust Flater", p.getVolledigeNaam( ));
    }

    @Test
    public void testGetVolledigeNaamIndienNaamNull() {
        Persoon p = new Persoon(null, "Guust");
        assertEquals("Guust ?", p.getVolledigeNaam( ));
    }

    @Test
    public void testGetVolledigeNaamIndienVoornaamNull() {
        Persoon p = new Persoon("Flater", null);
        assertEquals("? Flater", p.getVolledigeNaam( ));
    }

}
```

Vooraleer testen uit te voeren in IntelliJ

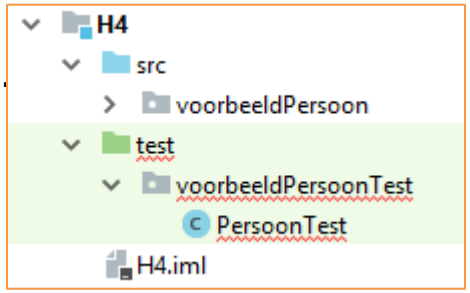
1. Maak een directory test aan onder de module H4



2. Deze directory moet gemarkeerd worden als een test directory. RMK / Mark directory as / Test Sources Root. Het icoon voor de test is dan gewijzigd.

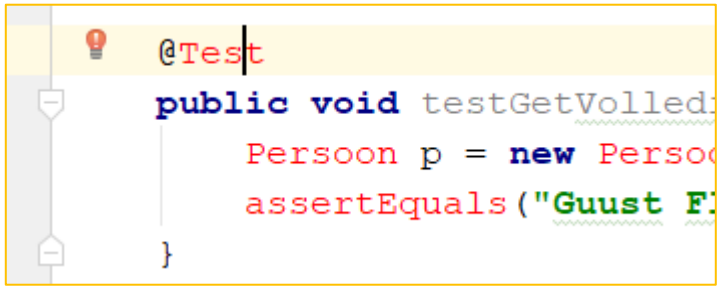


3. Maak in de test directory een package voorbeeldPersoonTest. Hieronder komt de test te staan.



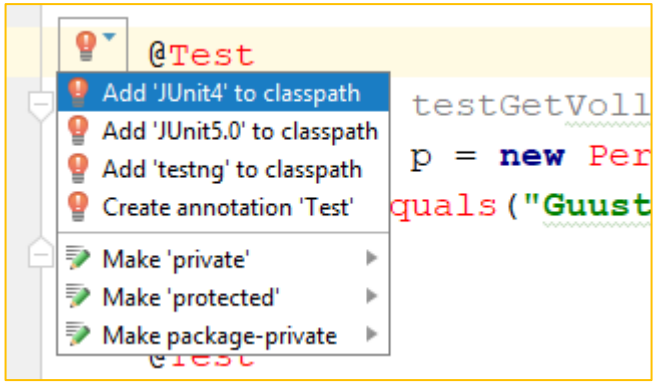
Vooraleer testen uit te voeren in IntelliJ

4. De test geeft een foutmelding omdat JUnit4 nog niet toegevoegd is aan het classpath.
   Hoe doe je dit?
   - In de code van de test klik je in een rode "@Test" .
     Er verschijnt een rood lampje.

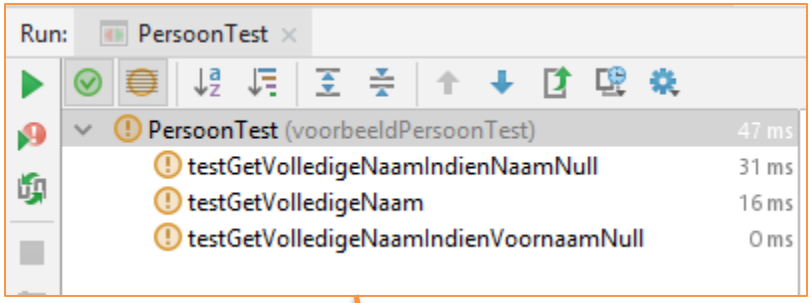


   - Opmerking: eventueel moet je eerst de klasse Persoon importeren .

   - Je kan nu gemakkelijk JUnit4 toevoegen aan het classpath.



Testen uitvoeren



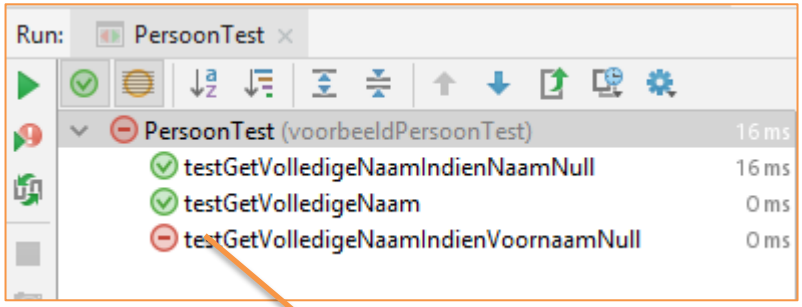
Tests failed: 3 of 3 tests – 47 ms

```
java.lang.AssertionError:
Expected :Guust Flater
Actual   :null
 <Click to see difference>

<1 internal call>
    at org.junit.Assert.failNotEquals(Assert.java:834) <2 internal calls>
    at voorbeeldPersoonTest.PersoonTest.testGetVolledigeNaam(PersoonTest.java:14) <22 internal calls>
```

Testen uitvoeren

```
package voorbeeldPersoon;

public class Persoon {
    private String naam;
    private String voornaam;

    public Persoon(String naam, String voornaam) {
        this.naam = naam;
        this.voornaam = voornaam;
    }

    public String getVolledigeNaam() {
        StringBuilder volledigeNaam = new StringBuilder(voornaam);
        if (naam == null) {
            volledigeNaam.append(" ?");
        } else {
            volledigeNaam.append(" ").append(naam);
        }
        return volledigeNaam.toString();
    }
}
```

Testen uitvoeren



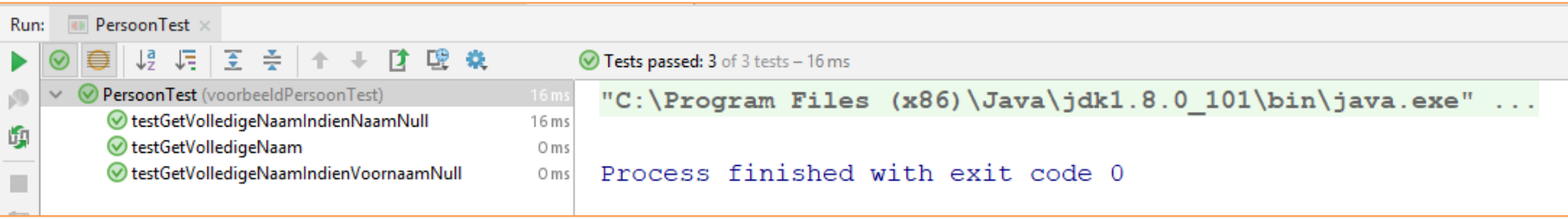
Tests failed: 1, passed: 2 of 3 tests – 16 ms

```
"C:\Program Files (x86)\Java\jdk1.8.0_101\bin\java.exe" ...

java.lang.NullPointerException
    at java.lang.StringBuilder.<init>(StringBuilder.java:112)
    at voorbeeldPersoon.Persoon.getVolledigeNaam(Persoon.java:13)
    at voorbeeldPersoonTest.PersoonTest.testGetVolledigeNaamIndienVoornaamNull(PersoonTest.java:26) <22 internal calls>

Process finished with exit code -1
```

Testen uitvoeren

```
public String getVolledigeNaam() {
    StringBuilder volledigeNaam = new StringBuilder();
    if (voornaam == null) {
        volledigeNaam.append("?");
    } else {
        volledigeNaam.append(voornaam);
    }
    if (naam == null) {
        volledigeNaam.append(" ?");
    } else {
        volledigeNaam.append(" ").append(naam);
    }
    return volledigeNaam.toString();

}
```

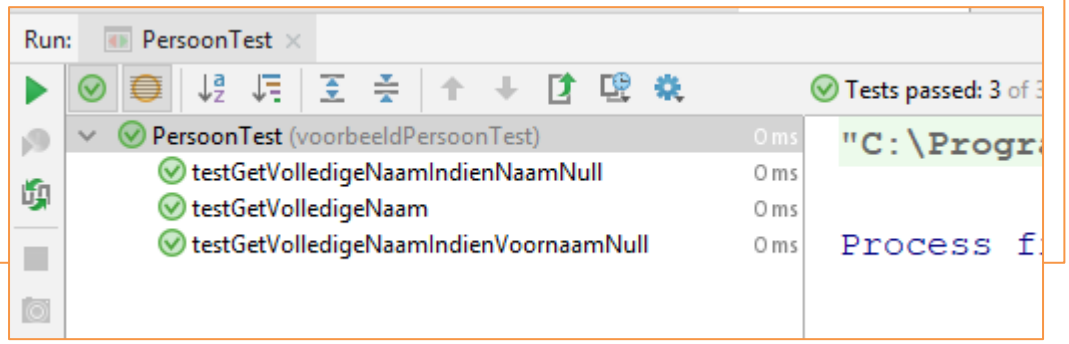
Testen uitvoeren



Refactor Kan je de leesbaarheid verbeteren?

Kan je dubbele code vermijden?

```
public String getVolledigeNaam() {
    StringBuilder volledigeNaam = new StringBuilder();
    volledigeNaam.append(vraagtekenIndienNull(voornaam));
    volledigeNaam.append(" ");
    volledigeNaam.append(vraagtekenIndienNull(naam));
    return volledigeNaam.toString();
}

private String vraagtekenIndienNull(String naam) {
    if (naam == null) {
        return "?";
    }
    return naam;

}
```



Methoden in org.junit.Assert

Methode	Betekenis
assertEquals()	Evalueert de gelijkheid van 2 waarden. De test slaagt als beide waarden gelijk (equal) zijn.
assertFalse()	Evaluatie van een booleaanse uitdrukking. De test slaagt indien de uitdrukking false is.
assertTrue()	Evaluatie van een booleaanse uitdrukking. De test slaagt indien de uitdrukking true is.
assertNotNull( )	Vergelijkt een object referentie met null. De test slaagt indien de object referentie niet null is.
assertNull( )	Vergelijkt een object referentie met null. De test slaagt indien de object referentie null is.
assertSame( )	Vergelijkt het geheugenadres van twee object referenties (gebruik maken van == operator). De test slaagt indien beide object referenties naar hetzelfde object verwijzen.
fail()	Zorgt ervoor de de huidige test zal fallen. Wordt regelmatig gebruikt bij het testen van exception handling.