Testplan

March 12, 2017

Versie historie

03/11/2017 Initiële versie

Introductie

Dit project calculeert het aantal inwoners voor een wijk voor het gegeven jaar. De tests die zijn uitgewerkt zijn een aantal unit tests, en een integratietest. Deze tests zijn geschreven om de juiste werking van het systeem te garanderen.

1 Testomgeving

Tijdens deze tests wordt de Wijk implementatie getest. Hiervoor wordt JUnit gebruikt.

2 Smoketest

Om de kwaliteit van het systeem te waarborgen zijn een aantal tests geschreven. Deze testen de code in een normale situatie, een situatie waarin de naam van de wijk leeg is, een situatie waarin het jaar 0 is, en een situatie waarin het jaar een negatief getal betreft. Ook wordt er gekeken of er een fout wordt gegeven als de wijknaam null is.

3 Integratietest

De integratietest doet hetzelfde als de Unit tests, alleen dan van buiten. Hiervoor is de WebServiceTester geschreven. Hieronder is een testgeval beschreven:

Input: Wijk: Amsterdam Bijlmer Arena
Jaar: 2017

Verwachte output: 6470

Daadwerkelijke output: 6470

4 Unit tests

Er zijn een aantal unit tests geschreven die testen of het programma de juiste output geeft en zich gedraagt zoals de bedoeling is. Het framework dat hiervoor gebruikt wordt is JUnit. Deze tests worden automatisch uitgevoerd op het moment dat het programma gecompileerd wordt.

```java
@Test
public void calcInwonersVoorWijkNormal() throws Exception {
    assertEquals(wijk.calcInwonersVoorWijk("test", 2015), 6450);
    assertEquals(wijk.calcInwonersVoorWijk("Nieuwegein Zuid", 2015), 6458);
    assertEquals(wijk.calcInwonersVoorWijk("Amsterdam Zuid", 2015), 6457);
    assertEquals(wijk.calcInwonersVoorWijk("Amsterdam Bijlmer Arena", 2015), 6464);

    assertEquals(wijk.calcInwonersVoorWijk("test", 2016), 6454);
    assertEquals(wijk.calcInwonersVoorWijk("Nieuwegein Zuid", 2016), 6461);
    assertEquals(wijk.calcInwonersVoorWijk("Amsterdam Zuid", 2016), 6461);
    assertEquals(wijk.calcInwonersVoorWijk("Amsterdam Bijlmer Arena", 2016), 6467);

    assertEquals(wijk.calcInwonersVoorWijk("test", 2017), 6457);
    assertEquals(wijk.calcInwonersVoorWijk("Nieuwegein Zuid", 2017), 6464);
    assertEquals(wijk.calcInwonersVoorWijk("Amsterdam Zuid", 2017), 6464);
    assertEquals(wijk.calcInwonersVoorWijk("Amsterdam Bijlmer Arena", 2017), 6470);
}

@Test
public void calcInwonersVoorWijkEmptyName() throws Exception {
    assertEquals(wijk.calcInwonersVoorWijk("", 0), 0);
    assertEquals(wijk.calcInwonersVoorWijk("", 1), 3);
    assertEquals(wijk.calcInwonersVoorWijk("", 2), 6);
    assertEquals(wijk.calcInwonersVoorWijk("", 3), 9);
}

@Test
public void calcInwonersVoorWijkYearZero() throws Exception {
    assertEquals(wijk.calcInwonersVoorWijk("test", 0), 2);
    assertEquals(wijk.calcInwonersVoorWijk("Nieuwegein Zuid", 0), 10);
    assertEquals(wijk.calcInwonersVoorWijk("Amsterdam Zuid", 0), 9);
    assertEquals(wijk.calcInwonersVoorWijk("Amsterdam Bijlmer Arena", 0), 16);
}

@Test
public void calcInwonersVoorWijkYearNegative() throws Exception {
    assertEquals(wijk.calcInwonersVoorWijk("test", -1), 0);
    assertEquals(wijk.calcInwonersVoorWijk("Nieuwegein Zuid", -1), 7);
    assertEquals(wijk.calcInwonersVoorWijk("Amsterdam Zuid", -1), 6);
    assertEquals(wijk.calcInwonersVoorWijk("Amsterdam Bijlmer Arena", -1), 12);
}

@Test(expected = NullPointerException.class)
public void calcInwonersVoorWijkYearNull() throws Exception {
    wijk.calcInwonersVoorWijk(null, 2017);
    wijk.calcInwonersVoorWijk(null, 2017);
    wijk.calcInwonersVoorWijk(null, 2017);
    wijk.calcInwonersVoorWijk(null, 2017);
}
```

Figure 1: Unit tests