**Yksikkötestaus ja Test Driven Development, JUnit**

**TK0802 -teeman tehtävät**

**TRTKNU15**

Kari Haavisto

SISÄLLYS

SANASTO 1

1 Taso 1 1

1.1 Tee JUnit Getting Started tutoriaali 1

1.2 Aja testitapaukset kehitysympäristössäsi 1

1.3 Luo useampi testitapaus käyttäen erilaisia assert-metodeja. 1

2 Taso 2 1

2.1 Kuvaile omin samoin, minkälaisia piirteistä on hyvällä yksikkötestillä? 1

2.2 Mitä hyötyjä yksikkötesteistä on? 1

2.3 Mitä haittaa yksikkötesteistä on tai missä tilanteessa et käyttäisi niitä? 1

2.4 Kuvaile omin sanoin mitä asioita yksikkötestauksessa varsinaisesti testataan? 2

2.5 Mikä ero on yksikkötestauksella ja ns. test driven development (TDD) -menetelmällä? 2

2.6 Mikä ero edellisiin on ns. behavior driven development (BDD) -menetelmällä? 2

2.7 Mitkä ovat TDD -menetelmän säännöt, joiden perusteella testitapaukset tulisi kirjoitta? 2

3 TASO 3 2

3.1 Test driven development -käytännössä 2

SANASTO

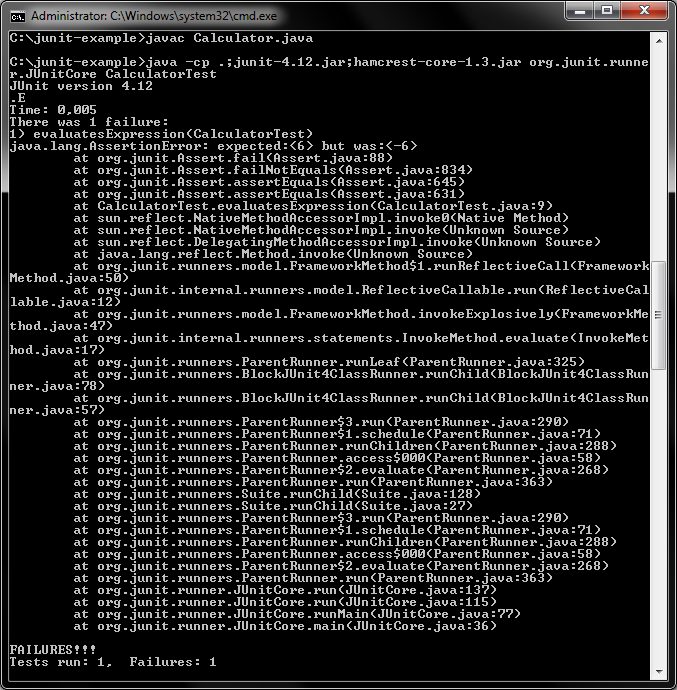
Tutustuessasi tehtäviin sekä tekemällä niitä, kerää tähän käytettyjä termejä, joista sinulla ei aluksi ole tietoa. Kuvaile löytämäsi termit siten, että ymmärrät mitä niillä tarkoitetaan.

1. Taso 1
   1. Tee JUnit Getting Started tutoriaali

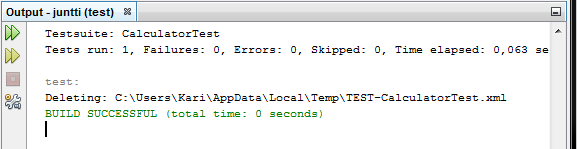
**OK testi**

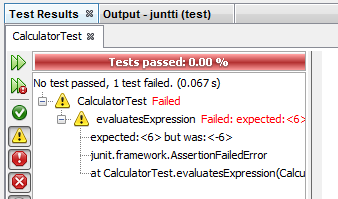


**Failed**



* 1. Aja testitapaukset kehitysympäristössäsi





* 1. Luo useampi testitapaus käyttäen erilaisia assert-metodeja.

assertEquals

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class CalculatorTest {

@Test

public void evaluatesExpression() {

Testi testi = new Testi();

String nimi = "Pekka";

assertEquals(nimi,testi.haeNimi());

}

}

public class Testi {

private String nimi;

public Testi(){

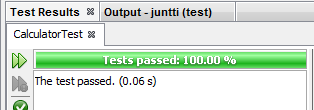
}

public String haeNimi(){

return "Pekka";

}

}



assertTrue

import static org.junit.Assert.assertTrue;

import org.junit.Test;

public class CalculatorTest {

@Test

public void Testi() {

Testi test = new Testi();

assertTrue(test.onkoParillinenLuku(12));

}

}

public class Testi {

public boolean onkoParillinenLuku(int numero) {

boolean tulos = false;

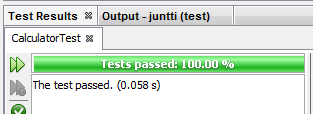
if (numero % 2 == 0) {

tulos = true;

}

return tulos;

}



assertFalse

import static org.junit.Assert.assertFalse;

import org.junit.Test;

public class CalculatorTest {

@Test

public void Testi() {

Testi test = new Testi();

assertFalse(test.onkoParillinenLuku(13));

}

}

public class Testi {

public boolean onkoParillinenLuku(int numero) {

boolean tulos = false;

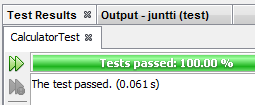
if (numero % 2 == 0) {

tulos = true;

}

return tulos;

}



1. Taso 2
   1. Kuvaile omin samoin, minkälaisia piirteistä on hyvällä yksikkötestillä?

Kerro omin sanoin. Saat käyttää erilaisia lähteitä.

* 1. Mitä hyötyjä yksikkötesteistä on?

Kerro omin sanoin. Saat käyttää erilaisia lähteitä.

* 1. Mitä haittaa yksikkötesteistä on tai missä tilanteessa et käyttäisi niitä?

Kerro omin sanoin. Saat käyttää erilaisia lähteitä.

* 1. Kuvaile omin sanoin mitä asioita yksikkötestauksessa varsinaisesti testataan?

Kerro omin sanoin. Saat käyttää erilaisia lähteitä.

* 1. Mikä ero on yksikkötestauksella ja ns. test driven development (TDD) -menetelmällä?

Kerro omin sanoin. Saat käyttää erilaisia lähteitä.

* 1. Mikä ero edellisiin on ns. behavior driven development (BDD) -menetelmällä?

Kerro omin sanoin. Saat käyttää erilaisia lähteitä.

* 1. Mitkä ovat TDD -menetelmän säännöt, joiden perusteella testitapaukset tulisi kirjoitta?

Kerro omin sanoin. Saat käyttää erilaisia lähteitä.

1. TASO 3
   1. Test driven development -käytännössä

Nauhoita kuvaruutuvideo. Videossa, yritä ratkaista alla annettu tehtävänanto TDD -menetelmää käyttäen. Eli tee testi aina ennen uuden vaiheen ohjelmointia. Videon aikana, sinun ei tarvitse päästä loppuun asti vaan tee niin pitkälle kuin osaat. Voit harjoitella kyllä etukäteen. Ratkaisun lopputuloksella ei ole väliä vaan tehtävän tarkoitus on pystyä havainnoimaan, kuinka luot ohjelmat. Ohjaaja katsoo videon ja antaa sen perusteella palautetta mitä voisit omassa työskentelyssä parantaa. Videon pituudella ei ole väliä. Tee joko niin pitkä kuin toteuttamiseen menee aikaa tai nauhoita alusta noin 15min. Tärkeintä on nähdä toteutus alusta lähtien.

**Tehtävänanto**: Merkkijono laskin.

Tehtävänäsi on luoda ohjelma, joka laskee numeroiden summan. Numeroita ei anneta lukutyyppeinä vaan merkkijonona. Ohjelman pitää tukea seuraavia vaadittuja toimintoja.

1. Tyhjä merkkijono palauttaa nollan.
2. Yksittäinen luku palauttaa arvoksi annetun luvun.
3. Kaksi lukua, pilkulla erotettuna, palauttaa lukujen summan.
4. Kaksi lukua, rivinvaihdolla (newline, /n) erotettuna, palauttaa lukujen summan.
5. Kolme lukua, erotettuna kummalla tahansa tapaa, palauttaa lukujen summan.
6. Negatiiviset luvut aiheuttavat virheen.
7. Luvut, jotka ovat yhtäsuuria tai suurempia kuin 1000, ei huomioida.