Haaga-Helia

Yksikkötestaus

Haaga-Helia

Testaus

- Testauksen tavoitteena on löytää virheitä.
- Täydellisen kattava testaus ei ole mahdollista.
- Kompleksisuus ja virheet eivät jakaudu tasaisesti.
- Testausta voidaan tehdä joko automatisoidusti tai manuaalisesti.
- Testausta tehdään tyypillisesti useilla eri tasoilla:
  - yksikkötestaus
  - integraatiotestaus
  - järjestelmätestaus
  - hyväksymistestaus
- Testejä lisätään jatkuvasti kehitystyön yhteydessä ja niitä suoritetaan uudelleen
  - regressiotestaus

Haaga-Helia

Testaus ohjelmointikursseilla tähän asti

- Tähän mennessä olette varmasti testanneet ohjelmianne suorittamalla koodia ja antamalla sovellukselle erilaisia syötteitä.
- Vastaavasti Viope testaa sovelluksia suorittamalla niitä, antamalla erilaisia syötteitä ja vertailemalla ohjelmien tulosteita.
- Ohjelman käyttöliittymän kautta testaaminen on työlästä ja monimutkaisten sovellusten kanssa kattava testaus on hyvin haastavaa:
  - Jokaista testattavaa tapausta varten täytyy ohjelma esim. suorittaa "kokonaan"
  - Maksutapahtuman testaamiseksi sinun tulee kirjautua sisään palveluun, lisätä tuote ostoskoriin, valita maksu- ja toimitustapa, syöttää osoitetiedot jne.

- Olette mahdollisesti myös koodanneet erilaisia main-metodeja, joiden avulla testataan tiettyjä metodeja ohjelmallisesti
  - ➢ hyvä ratkaisu, mutta tulosteet täytyy silti käydä manuaalisesti läpi

Haaga-Helia

Testauksen tasot

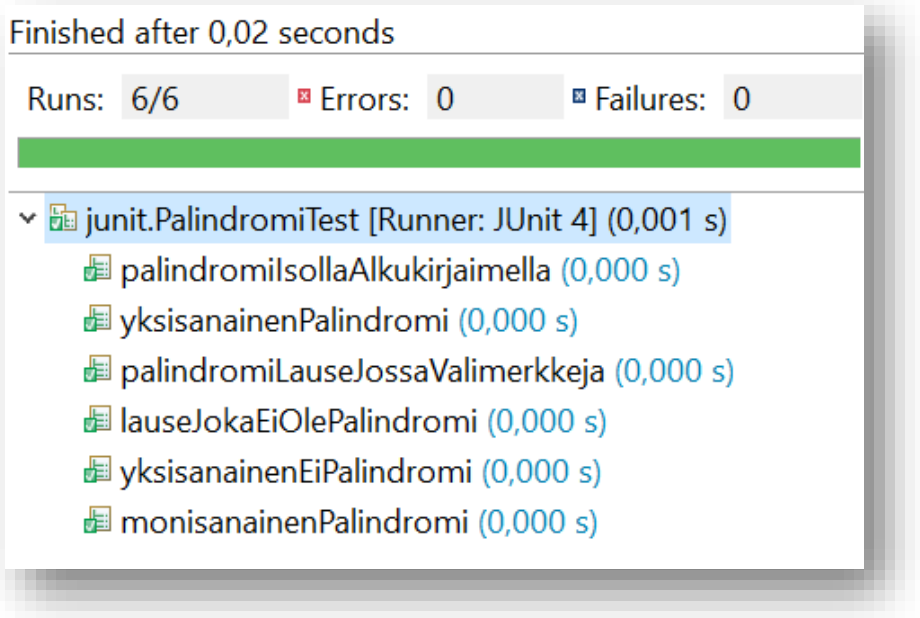
Miten testaamme jatkossa paremmin/laajemmin/helpommin/...?

Yksikkötestaus

Haaga-Helia

Yksikkötestaus

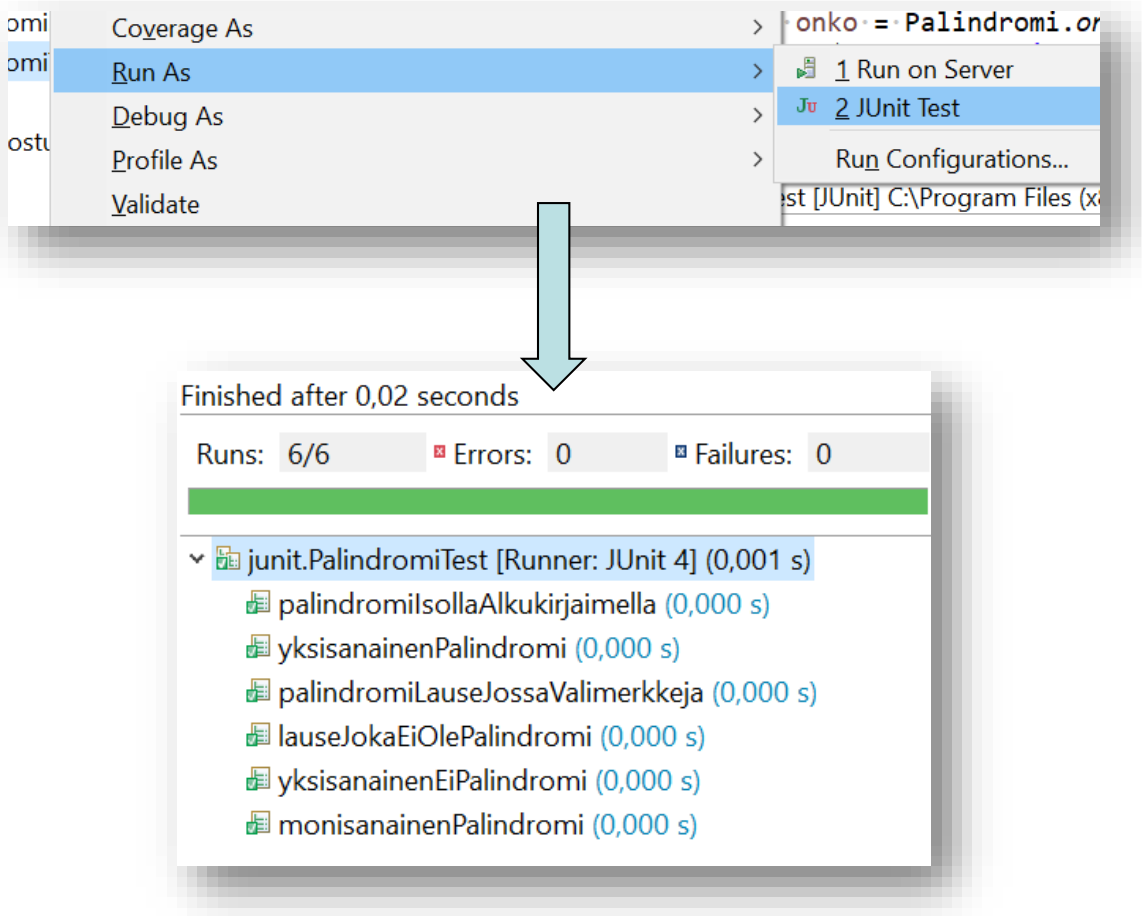
- Testaustapa, jossa yksittäisille, hyvin pienille ohjelman osille kirjoitetaan erillisiä testejä.
- Testit suoritetaan tyypillisesti automaattisesti ja testien suorituksesta muodostetaan helppolukuinen raportti.
- Samoja luokkia ja metodeja testataan tyypillisesti lukuisilla erilaisilla syötteillä.



Haaga-Helia

JUnit

- JUnit on yksikkötestauskirjasto, josta on muodostunut de facto –standardi Javan yksikkötestaukseen
- JUnit perustuu testiluokkiin, joihin kirjoitetaan testimetodeita:
  - yksi tai useita testiluokkia voidaan suorittaa kerralla
  - jokaisen testimetodin suorituksesta saadaan joko **onnistunut** tai **epäonnistunut** tulos
- JUnit-testejä voidaan ajaa monilla eri tavoilla, mm. Eclipsestä löytyy ominaisuudet JUnit-testien ajoon ja raportointiin



JUnit-kirjaston nykyinen versio on 5 (Jupiter). Vanhat versiot ovat samankaltaisia, mutta annotaatiot ovat erinimisiä ja eri paketeissa.

Haaga-Helia

Testiluokat

- Testiluokkien nimet alkavat tyypillisesti sen luokan nimellä, jota on tarkoitus testata ja päättyvät sanaan "Test"
- Testeissä suoritettaviin metodeihin lisätään JUnit-kirjaston annotaatiot:

@Test –yksittäinen testitapaus

@BeforeEach –suoritetaan ennen jokaista testitapausta

@AfterEach –suoritetaan jokaisen testitapauksen jälkeen

@BeforeClass –suoritetaan ennen luokan ensimmäistä testiä

@AfterClass –suoritetaan kun kaikki testitapaukset on käyty läpi

```
import org.junit.jupiter.api.Test;

public class MyClassTest {

    @Test
    public void testMethod1() {
        // Täällä suoritetaan testi 1
    }

    @Test
    public void testMethod2() {
        // Täällä suoritetaan testi 2
    }

    // ...
}
```

Haaga-Helia

Assertiot

- JUnit-kirjastossa on assert-metodeja, joiden avulla tarkastetaan, että testattava koodi toimii oikein

- `assertEquals` tarkastaa, että sille annetut arvot ovat samat.
  - Ensimmäisenä parametrina annetaan tunnettu "oikea vastaus"
  - Toisena parametrina annetaan testattava arvo

- Jos arvot eivät täsmää, kyseinen testi merkitään epäonnistuneeksi

```
@Test
public void testSumOfIntegers() {
    int summa = 3 + 2 + 1;
    assertEquals(6, summa);
}

@Test
public void testHelloLength() {
    assertEquals(5, "Hello".length());
}

@Test
public void testStringReplace() {
    String teksti = "Hello World!";
    String muutettu = teksti.replace("World", "JUnit");

    assertEquals("Hello JUnit!", muutettu);
}
```

Haaga-Helia

Demo

Testiluokan luominen, annotaatiot, testimetodit, assertiot ja testin suorittaminen

Katso video:
https://video.haaga-helia.fi/media/t/0_pl76xbuy

Haaga-Helia

Testien sijainti ja rakenne

- Testiluokat sijaitsevat usein `src/test/java` – lähdekoodihakemistossa
  - Tällöin tuotantokoodi sijaitsee hakemistossa `src/main/java`
- Yksikkötestiluokat sijaitsevat tyypillisesti samannimisissä paketeissa kuin niillä testattavat luokat
- Yksi testiluokka testaa tyypillisesti vain yhtä luokkaa
- Kukin testiluokka sisältää tyypillisesti lukuisia erilaisia testitapauksia (`@Test`-metodeita)
- Kukin testitapaus sisältää vähintään yhden `assertion`

Haaga-Helia

Assert-metodeita (org.junit.jupiter.api.Assertions.*)

- `assertEquals(a, b)`
  - a ja b ovat "samat" (equals-metodilla)
- `assertTrue(a)`
  - a on oltava true
- `assertFalse(a)`
  - a on oltava false
- `assertNull(a)`
  - a on oltava null-viittaus
- `assertNotNull(a)`
  - a ei saa olla null-viittaus

+ monia muita: https://junit.org/junit5/docs/5.0.1/api/org/junit/jupiter/api/Assertions.html

Haaga-Helia

Demo

Laskuri-luokan toteutus ja testaus

Video 1:
https://video.haaga-helia.fi/media/t/0_1gkcscbe

Video 2:
https://video.haaga-helia.fi/media/t/0_poklvdms

Toisen luokan testaaminen

Haaga-Helia

```
public class LaskuriTest {

    private Laskuri laskuri;

    @Before
    public void setUp() {
        laskuri = new Laskuri();
    }

    @Test
    public void yhdenLuvunSumma() {
        int[] numerot = new int[] { 12 };
        int summa = laskuri.summa(numerot);

        assertEquals(12, summa);
    }

    @Test
    public void useanLuvunSumma() {
        int[] numerot = new int[] { 12, 34 };
        int summa = laskuri.summa(numerot);

        assertEquals(46, summa);
    }

    @Test
    public void tyhjanTaulukonSumma() {
        int[] numerot = new int[] {};
        int summa = laskuri.summa(numerot);
        assertEquals(0, summa);
    }

    @Test
    public void negatiivisetLuvutSummassa() {
        int[] numerot = new int[] {1, 2, -3, -4};
        int summa = laskuri.summa(numerot);
        assertEquals(-4, summa);
    }
}
```

Testin alustus

Tuloksen tarkastus

```
public class Laskuri {
    public int summa(int[] luvut) {
        int yhteensa = 0;
        for (int luku : luvut) {
            yhteensa += luku;
        }
        return yhteensa;
    }
}
```

Testattava luokka

Haaga-Helia

Mitä on yleisesti hyvä testata?

- Odotetusti toimivat arvot
- Loogisesti virheelliset arvot
- Tyhjät arvot
  - Tyhjät listat, taulukot, merkkijonot...
- Raja-arvot
  - Sallittujen rajojen sisällä olevia arvoja
  - Sallitut maksimi- ja minimiarvot
  - Liian pienet, suuret, lyhyet tai pitkät arvot
- Null-arvot
  - Olioviittauksen sijasta annetaankin null
- Erikoistapaukset
  - Erikoismerkit + skandit (å, ä, ö)
  - Positiivinen ja negatiivinen "ääretön"

Haaga-Helia

Testivetoinen kehitys (test-driven development)

- Ohjelmistokehitysprosessi, jossa edetään hyvin lyhyissä testauksen ja toteutuksen sykleissä.
- Palaute kirjoitetun koodin toimivuudesta saadaan hyvin pian, minkä vuoksi virheiden löytäminen on usein hyvin helppoa ja nopeaa.
- Ensin koodataan testi, minkä jälkeen koodataan vain sen verran koodia, että testi menee läpi ja toistetaan, kunnes haluttu toiminnallisuus on saavutettu.
- Koodia voidaan refaktoroida ilman huolta siitä, että koodin muokkaaminen aiheuttaisi uusia bugeja.

Haaga-Helia

Edistynyttä sisältöä: testivetoisen kehityksen demo

Katso esimerkki videona osoitteessa:
https://video.haaga-helia.fi/media/t/0_m8y5zv8k