VDAB

samen sterk voor werk

Test Driven Development met JUnit takenbundel

Inhoudsopgave

1 TAKEN

1.1 Palindroom

Je maakt een class `Woord`

Je geeft aan de constructor een woord mee.

De method `isPalindroom` geeft enkel `true` terug
als dit woord een palindroom is: een woord dat hetzelfde is
als je het van voor naar achter leest
en als je het van achter naar voor leest (bvb. `lepel`).

Woord
+Woord(woord: String) +isPalindroom(): boolean

Je schrijft in een class `WoordTest` de nodige tests voor de class `Woord`

1.2 Veiling

Je maakt met de voorgeschreven stappen van TDD een class die
een veiling (verkoop) voorstelt en de bijbehorende unit test.

Je kan op een `Veiling` object meerdere keren
de method doeBod oproepen.
Je geeft als parameter het bedrag van het bod mee.
Je kan op ieder moment de method `getHoogsteBod` oproepen.
Deze geeft het hoogst geboden bedrag terug.

Veiling
+doeBod(bedrag: int) +getHoogsteBod(): int

```
Veiling veiling = new Veiling();
veiling.doeBod(1000);
int hoogsteBod = veiling.getHoogsteBod();  // hoogsteBod bevat 1000
veiling.doeBod(2000);
hoogsteBod = veiling.getHoogsteBod(); // hoogsteBod bevat 2000
```

Uit de analyse blijkt dat

- Als nog geen enkel bod werd uitgevoerd, het hoogste bod gelijk is aan 0.
- Als een eerste bod werd uitgevoerd, het hoogste bod gelijk is aan het bedrag van dit bod.
- Als meerdere keren een bod werd uitgevoerd,
  het hoogste bod gelijk is aan bedrag van het hoogste bod.

1.3 Test fixtures

Je gebruikt een `@Before` method in de unit test van de class `Veiling`.

1.4 ISBN

Je maakt met de voorgeschreven stappen van TDD
een class die een ISBN (Internationaal Standaard Boeknummer)
voorstelt en de bijbehorende unit test.

Isbn
+Isbn(nummer: long) +toString(): String

De regels van ISBN zijn als volgt

- Het bestaat uit 13 cijfers.
- De eerste 3 cijfers zijn 978 of 979.
- Een ISBN bevat een controle-mechanisme
  - Je maakt de som van de cijfers op de eerste 6 oneven posities
  - Je maakt de som van de cijfers op de eerste 6 even posities
    en je vermenigvuldigt deze som maal 3.
  - Je maakt de som van deze twee tussenresultaten.
  - Je maakt het verschil van deze som en het naastgelegen hoger gelegen tiental.
  - Het dertiende cijfer moet gelijk zijn aan dit verschil,
    tenzij het verschil 10 is. Dan moet het dertiende cijfer gelijk zijn aan 0.

Voorbeeld het ISBN 9789027439642

- de som van de cijfers op de eerste 6 oneven posities
  `9 + 8 + 0 + 7 + 3 + 6 = 33`
- de som van de cijfers op de eerste 6 even posities, maal 3
  `7 + 9 + 2 + 4 + 9 + 4 = 35 x 3 = 105`
- de som van deze twee tussenresultaten
  `33 + 105 = 138`
- het verschil van deze som en het naastgelegen hoger gelegen tiental
  2
- het dertiende cijfer is gelijk aan dit verschil

1.5 Stub

Je maakt een class `WinstService`.

Deze class heeft een dependency, uitgedrukt in een interface `OpbrengstRepository`.
Deze interface bevat één method declaratie: `BigDecimal findTotaleOpbrengst();`

De class `WinstService` heeft een tweede dependency, uitgedrukt in een interface `KostRepository`. Deze interface bevat één method declaratie: `BigDecimal findTotaleKost();`

De class `WinstService` bevat één method: `BigDecimal getWinst();`
De berekening van de winst is: totale opbrengst – totale kost

Je schrijft de class `WinstService` en de bijbehorende unit test.
In deze unit test gebruik je stubs voor `OpbrengstRepository` en `KostRepository`.

1.6 Mock

Je vervangt de stubs in `WinstService` door mocks, aangemaakt met Mockito.

Je doet ook verificaties op deze stubs.

1.7 WoordTeller

Je zoekt het aantal woorden in een zin. Woorden worden van mekaar gescheiden door één of meerdere spaties en/of komma's.

2 VOORBEELDOPLOSSINGEN

2.1 Palindroom

2.1.1 Woord

```
package be.vdab.util;
public class Woord {
  private final String woord;
  public Woord(String woord) {
    this.woord = woord;
  }
  public boolean isPalindroom() {
    String omgekeerdWoord = new StringBuilder(woord).reverse().toString();
    return woord.equals(omgekeerdWoord);
  }
}
```

2.1.2 WoordTest

```
package be.vdab.util;
import org.junit.Test;
import static org.junit.Assert.*;
public class WoordTest {
  @Test
  public void lepelIsEenPalindroom() {
    Woord woord = new Woord("lepel");
    assertTrue(woord.isPalindroom());
  }
  @Test
  public void vorkIsGeenPalindroom() {
    Woord woord = new Woord("vork");
    assertFalse(woord.isPalindroom());
  }
  @Test
  public void eenLegeStringIsEenPalindroom() {
    Woord woord = new Woord("");
    assertTrue(woord.isPalindroom());
  }
}
```

2.2 Veiling

2.2.1 De class Veiling zonder echte code in zijn methods

```
package be.vdab.valueobjects;
public class Veiling {
  public void doeBod(int bedrag) {
    throw new UnsupportedOperationException();
  }
  public int getHoogsteBod() {
    throw new UnsupportedOperationException();
  }
}
```

2.2.2 De unit test VeilingTest

```
package be.vdab.valueobjects;
import org.junit.Test;
import static org.junit.Assert.*;
public class VeilingTest {
  @Test
  public void hetHoogsteBodVanEenNieuweVeilingIsNul() {
    assertEquals(0, new Veiling().getHoogsteBod());
  }
  @Test
  public void naEenEersteBodIsHetHoogsteBodGelijkAanHetBedragVanDitBod() {
    Veiling veiling = new Veiling();
    veiling.doeBod(100);
    assertEquals(100, veiling.getHoogsteBod());
  }
  @Test
  public void hetHoogsteBodNaEerst100Dan200Dan150Is200() {
    Veiling veiling = new Veiling();
    veiling.doeBod(100);
    veiling.doeBod(200);
    veiling.doeBod(150);
    assertEquals(200, veiling.getHoogsteBod());
  }
}
```

2.2.3 De class Veiling met echte code in zijn methods

```
package be.vdab.valueobjects;
public class Veiling {
  private int hoogsteBod; // Java initialiseert een private int default op 0
  public void doeBod(int bedrag) {
    if (bedrag > hoogsteBod) {
      hoogsteBod = bedrag;
    }
  }
  public int getHoogsteBod() {
    return hoogsteBod;
  }
}
```

2.3 Test fixtures

```
package be.vdab.valueobjects;
import static org.junit.Assert.assertEquals;
import org.junit.Before;
import org.junit.Test;
public class VeilingTest {
  private Veiling veiling;
  @Before
  public void before() {
    veiling = new Veiling();
  }
  @Test
  public void hetHoogsteBodVanEenNieuweVeilingStaatOpNul() {
    assertEquals(0, veiling.getHoogsteBod());
  }
```

```
  @Test
  public void naEenEersteBodIsHetHoogsteBodGelijkAanHetBedragVanDitBod() {
    veiling.doeBod(100);
    assertEquals(100, veiling.getHoogsteBod());
  }
  @Test
  public void naMeerdereodIsHetHoogsteBodGelijkAanHetBedragVanDitBod() {
    veiling.doeBod(100);
    veiling.doeBod(200);
    veiling.doeBod(150);
    assertEquals(200, veiling.getHoogsteBod());
  }
}
```

2.4 ISBN

2.4.1 De class Isbn zonder echte code in zijn methods

```
package be.vdab.valueobjects;
public class Isbn {
  public Isbn(long nummer) {
    throw new UnsupportedOperationException();
  }
  @Override
  public String toString() {
    throw new UnsupportedOperationException();
  }
}
```

2.4.2 De unit test IsbnTest

```
package be.vdab.valueobjects;
import org.junit.Test;
import static org.junit.Assert.*;
public class IsbnTest {
  @Test(expected = IllegalArgumentException.class)
  public void hetNummer0IsVerkeerd() {
    new Isbn(0);
  }
  @Test(expected = IllegalArgumentException.class)
  public void eenNegatiefNummerIsVerkeerd() {
    new Isbn(-9789027439642L);
  }
  @Test(expected = IllegalArgumentException.class)
  public void eenNummerMet12CijfersIsVerkeerd() {
    new Isbn(978902743964L);
  }
  @Test(expected = IllegalArgumentException.class)
  public void eenNummerMet14CijfersIsVerkeerd() {
    new Isbn(97890274396421L);
  }
  @Test(expected=IllegalArgumentException.class)
  public void deEerste3CijfersMoeten978of979Zijn() {
    new Isbn(9779227439643L);
  }
  @Test(expected = IllegalArgumentException.class)
  public void eenNummerMet13CijfersMetVerkeerdControleGetal2() {
    new Isbn(8789027439642L);
  }
```

```
  @Test
  public void eenNummerMet13CijfersMetCorrectControleGetal2() {
    new Isbn(9789027439642L);
  }
  @Test(expected = IllegalArgumentException.class)
  public void eenNummerMet13CijfersMetVerkeerdControleGetal3() {
    new Isbn(9789027439643L);
  }
  @Test
  public void eenNummerMet13CijfersMetCorrectControleGetal0() {
    new Isbn(9789227439640L);
  }
  @Test
  public void toStringMoetHetNummerTeruggeven() {
    long nummer = 9789027439642L;
    assertEquals(nummer, Long.parseLong(new Isbn(nummer).toString()));
  }
}
```

2.4.3 De class Isbn met echte code in zijn methods

```
package be.vdab.valueobjects;
public class Isbn {
  private static final long KLEINSTE_GETAL_MET_13_CIJFERS = 1000_000_000_000L;
  private static final long GROOTSTE_GETAL_MET_13_CIJFERS = 9999_999_999_999L;
  private static final Set<Short> MOGELIJKE_EERSTE_3_CIJFERS = new HashSet<>();
  private final long nummer;
  static {
    // deze static initializer wordt één keer uitgevoerd in het programma,
    // als je de eerste keer de class Isbnr aanspreekt.
    // je kan in deze static initializer enkel static variabelen manipuleren
    MOGELIJKE_EERSTE_3_CIJFERS.add((short) 978);
    MOGELIJKE_EERSTE_3_CIJFERS.add((short) 979);
  }
  public Isbn(long nummer) {
    if (nummer < KLEINSTE_GETAL_MET_13_CIJFERS
     || nummer > GROOTSTE_GETAL_MET_13_CIJFERS) {
      throw new IllegalArgumentException("Bevat geen 13 cijfers");
    }
    short eerste3Cijfers = (short) (nummer / 10_000_000_000L);
      if (!MOGELIJKE_EERSTE_3_CIJFERS.contains(eerste3Cijfers)) {
      throw new IllegalArgumentException(
        "Begint niet met " + MOGELIJKE_EERSTE_3_CIJFERS);
      }
    long somEvenCijfers = 0;
    long somOnEvenCijfers = 0;
    long teVerwerkenCijfers = nummer / 10;
    for (int teller = 0; teller != 6; teller++) {
      somEvenCijfers += teVerwerkenCijfers % 10;
      teVerwerkenCijfers /= 10;
      somOnEvenCijfers += teVerwerkenCijfers % 10;
      teVerwerkenCijfers /= 10;
    }
    long controleGetal = somEvenCijfers * 3 + somOnEvenCijfers;
    long naastGelegenHoger10Tal = controleGetal - controleGetal % 10 + 10;
    long verschil = naastGelegenHoger10Tal - controleGetal;
    long laatsteCijfer = nummer % 10;
    if (verschil == 10) {
      if (laatsteCijfer != 0) {
        throw new IllegalArgumentException("Verkeerd controlegetal");
```

```
      }
    } else {
      if (laatsteCijfer != verschil) {
        throw new IllegalArgumentException("Verkeerd controlegetal");
      }
    }
    this.nummer = nummer;
  }
  @Override
  public String toString() {
    return String.valueOf(nummer);
  }
}
```

2.5 Stub

2.5.1 OpbrengstRepository

```
package be.vdab.repositories;
import java.math.BigDecimal;
public interface OpbrengstRepository {
  BigDecimal findTotaleOpbrengst();
}
```

2.5.2 KostRepository

```
package be.vdab.repositories;
import java.math.BigDecimal;
public interface KostRepository {
  BigDecimal findTotaleKost();
}
```

2.5.3 WinstService

```
package be.vdab.services;
import java.math.BigDecimal;
import be.vdab.repositories.KostRepository;
import be.vdab.repositories.OpbrengstRepository;
public class WinstService {
  private final OpbrengstRepository opbrengstRepository;
  private final KostRepository kostRepository;
  public WinstService(OpbrengstRepository opbrengstRepository,
    KostRepository kostRepository) {
    this.opbrengstRepository = opbrengstRepository;
    this.kostRepository = kostRepository;
  }
  public BigDecimal getWinst() {
    throw new UnsupportedOperationException();
  }
}
```

2.5.4 OpbrengstRepositoryStub

```
package be.vdab.repositories;
import java.math.BigDecimal;
public class OpbrengstRepositoryStub implements OpbrengstRepository {
  @Override
  public BigDecimal findTotaleOpbrengst() {
    return BigDecimal.valueOf(200);
  }
}
```

2.5.5 KostRepositoryStub

```
package be.vdab.repositories;
import java.math.BigDecimal;
public class KostRepositoryStub implements KostRepository {
  @Override
  public BigDecimal findTotaleKost() {
    return BigDecimal.valueOf(169);
  }
}
```

2.5.6 WinstServiceTest

```
package be.vdab.services;
import static org.junit.Assert.assertEquals;
import java.math.BigDecimal;
import org.junit.Before;
import org.junit.Test;
import be.vdab.repositories.KostRepository;
import be.vdab.repositories.KostRepositoryStub;
import be.vdab.repositories.OpbrengstRepository;
import be.vdab.repositories.OpbrengstRepositoryStub;
public class WinstServiceTest {
  private WinstService winstService;
  private OpbrengstRepository opbrengstRepository;
  private KostRepository kostRepository;
  @Before
  public void before() {
  opbrengstRepository = new OpbrengstRepositoryStub();
  kostRepository = new KostRepositoryStub();
  winstService = new WinstService(opbrengstRepository, kostRepository);
  }
  @Test
  public void winstIsOpbrengstMinKost() {
    assertEquals(0, BigDecimal.valueOf(31).compareTo(winstService.getWinst()));
  }
}
```

2.5.7 Echte code in de method getWinst van WinstService

```
public BigDecimal getWinst() {
  return opbrengstRepository.findTotaleOpbrengst()
    .subtract(kostRepository.findTotaleKost());
}
```

2.6 Mock

2.6.1 Juist voor de class WinstServiceTest

```
@RunWith(MockitoJUnitRunner.class)
```

2.6.2 Juist voor de private variabele opbrengstRepository en voor de variabele kostRepository

```
@Mock
```

2.6.3 Gewijzigde method before in WinstServiceTest

```
@Before
public void before() {
  when(opbrengstRepository.findTotaleOpbrengst())
    .thenReturn(BigDecimal.valueOf(200));
  when(kostRepository.findTotaleKost()).thenReturn(BigDecimal.valueOf(169));
  winstService = new WinstService(opbrengstRepository, kostRepository);
}
```

2.6.4 Extra opdrachten onder in @Test method in WinstServiceTest

```
verify(opbrengstRepository).findTotaleOpbrengst();
verify(kostRepository).findTotaleKost();
```

2.7 WoordTeller

2.7.1 De class WoordTeller zonder echte code

```
package be.vdab.util;
public class WoordTeller {
  public int telWoorden(String zin) {
    throw new UnsupportedOperationException();
  }
}
```

2.7.2 De unit test

```
package be.vdab.util;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class WoordTellerTest {
  private WoordTeller woordTeller;
  @Before
  public void before() {
    woordTeller = new WoordTeller();
  }
  @Test(expected=NullPointerException.class)
  public void eenNullWaarde() {
    woordTeller.telWoorden(null);
  }
  @Test
  public void eenLegeString() {
    assertEquals(0, woordTeller.telWoorden(""));
  }
  @Test
  public void enkelEenSpatie() {
    assertEquals(0, woordTeller.telWoorden(" "));
  }
  @Test
  public void enkelSpaties() {
    assertEquals(0, woordTeller.telWoorden("   "));
  }
  @Test
  public void enkelEenWoord() {
    assertEquals(1, woordTeller.telWoorden("woord"));
  }
  @Test
  public void eenWoordMetEenVoorafgaandeSpatie() {
    assertEquals(1, woordTeller.telWoorden(" woord"));
  }
  @Test
  public void eenWoordMetDaaropvolgendeSpatie() {
    assertEquals(1, woordTeller.telWoorden("woord "));
  }
  @Test
  public void tweeWoorden() {
    assertEquals(2, woordTeller.telWoorden("ik ook"));
  }
  @Test
  public void tweeWoordenMetDaartussenMeerdereSpaties() {
```

```
    assertEquals(2, woordTeller.telWoorden("ik  ook"));
  }
  @Test
  public void tweeWoordenMetEenKommaErTussen() {
    assertEquals(2, woordTeller.telWoorden("ik,ook"));
  }
  @Test
  public void enkelEenKomma() {
    assertEquals(0, woordTeller.telWoorden(","));
  }
  @Test
  public void enkelKommas() {
    assertEquals(0, woordTeller.telWoorden(",,,"));
  }
  @Test
  public void enkeKommasEnSpaties() {
    assertEquals(0, woordTeller.telWoorden(" , ,, "));
  }
  @Test
  public void tweeWoordenMetEenKommaEnEenSpatieErTussen() {
    assertEquals(2, woordTeller.telWoorden("ik, ook"));
  }
  @Test
  public void tweeWoordenMetEenSpatieEnEenKommaErTussen() {
    assertEquals(2, woordTeller.telWoorden("ik ,ook"));
  }
}
```

2.7.3 De class WoordTeller met echte code

```
package be.vdab.util;
public class WoordTeller {
  public int telWoorden(String zin) {
    int aantalWoorden = 0;
    int index = 0;
    while (index != zin.length()) {
      while (index != zin.length() &&
        (zin.charAt(index) == ' ' || zin.charAt(index) == ',')) {
        index++;
      }
      if (index != zin.length()) {
        aantalWoorden++;
        while (index != zin.length() && zin.charAt(index) != ' ' &&
          zin.charAt(index) != ',') {
          index++;
        }
      }
    }
    return aantalWoorden;
  }
}
```

2.7.4 De gerefactorde class WoordTeller

```
package be.vdab.util;
import java.util.StringTokenizer;
public class WoordTeller {
  public int telWoorden(String zin) {
    return new StringTokenizer(zin, " ,").countTokens();
  }
}
```

3 COLOFON

Domeinexpertisemanager: Jean Smits

Moduleverantwoordelijke: Jean Smits

Medewerkers: Hans Desmet

Versie: 12/4/2018