**TESTONTWERP & -REALISATIE**

|  |  |
| --- | --- |
| **Naam** | Vinay Mahadew |
| **Studentnummer** | 21022208 |
| **Klas** | 6 |

**OPDRACHT**

Geef hieronder een korte beschrijving van je opdracht (een samenvatting van je productvisie) zodat je docent OPT3 ook begrijpt met welke opdracht jij bezig bent.

Zie testCasus

**GITHUB EN TDD**

Maak je GitHub-repo Private en geef je docent toegang tot je repo:

https://github.com/VIMAH/OPT3

Om aan te tonen dat je volgens TDD hebt gewerkt voor deze opdracht neem je hieronder een kopie van je netwerk over (zie de toelichting op het template voor instructie):

<Kopie van je GitHub netwerkdiagram>

**GESLAAGDE TESTS**

Toon met een relevante screenshot aan dat alle onderstaande tests inderdaad zijn geslaagd:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

**SAMENGESTELDE DECISION**

In mijn software neem ik een besluit op basis van de volgende voorwaarden (maximaal 3 te testen):

Voorwaarde A: Heeft toestemming van toren

Voorwaarde B: gewicht < 1000kg

Voorwaarde C: snelheid > 50 km/uur

Het besluit is gebaseerd op de volgende combinatie van deze voorwaarden (bijv. D = A && (B || C)):

D = A && (B || C)

**CONDITION COVERAGE**

Als ik mijn software voor deze Decision zou testen, dan zou ik met Condition Coverage de volgende test cases testen (als er teveel regels in de tabel staan, kun je de waarden vervangen door een spatie):

|  |  |  |  |
| --- | --- | --- | --- |
| **Voorwaarde A** | **Voorwaarde B** | **Voorwaarde C** | **D = <Decision**> |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
|  | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |

**DECISION COVERAGE**

Als ik mijn software voor deze Decision zou testen, dan zou ik met Decision Coverage de volgende test cases testen (als er teveel regels in de tabel staan, kun je de waarden vervangen door een spatie):

|  |  |  |  |
| --- | --- | --- | --- |
| **Voorwaarde A** | **Voorwaarde B** | **Voorwaarde C** | **D = <Decision**> |
| 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |

**CONDITION/DECISION COVERAGE**

Als ik mijn software voor deze Decision zou testen, dan zou ik met Condition/Decision Coverage de volgende test cases testen (als er teveel regels in de tabel staan, kun je de waarden vervangen door een spatie):

|  |  |  |  |
| --- | --- | --- | --- |
| **Voorwaarde A** | **Voorwaarde B** | **Voorwaarde C** | **D = <Decision**> |
| 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |

**MODIFIED CONDITION/DECISION COVERAGE**

Als ik mijn software voor deze Decision zou testen, dan zou ik met Modified Condition/Decision Coverage de volgende test cases testen (als er teveel regels in de tabel staan, kun je de waarden vervangen door een spatie). Geef met <l>/<f> aan met welke logische en fysieke test cases je werkt:

|  |  |  |  |
| --- | --- | --- | --- |
| **Voorwaarde A** | **Voorwaarde B** | **Voorwaarde C** | **D = <Decision**> |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> |

**MULTIPLE CONDITION COVERAGE**

Als ik mijn software voor deze Decision zou testen, dan zou ik met Multiple Condition Coverage de volgende test cases testen (als er teveel regels in de tabel staan, kun je de waarden vervangen door een spatie):

|  |  |  |  |
| --- | --- | --- | --- |
| **Voorwaarde A** | **Voorwaarde B** | **Voorwaarde C** | **D = <Decision**> |
| **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **0** |
| **0** | **1** | **0** | **0** |
| **0** | **1** | **1** | **0** |
| **1** | **0** | **0** | **0** |
| **1** | **0** | **1** | **1** |
| **1** | **1** | **0** | **1** |
| 1 | 1 | 1 | 1 |
| <waarde> | <waarde> | <waarde> | <waarde> |

**MODIFIED CONDITION/DECISION COVERAGE (JUNIT)**

Ik heb de testcases voor Modified Condition/Decision Coverage omgezet naar de volgende JUnit-code om mijn software voor deze beslissing te testen.

import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class CessnaVliegtuigTest {  
  
 @Test  
 void TrueCheckCessnaVliegtuig () {  
 CessnaVliegtuig cessnaVliegtuig = new CessnaVliegtuig(1000,50.00,true);  
 *assertTrue*(cessnaVliegtuig.CheckCessnaVliegtuig(1050,60.00,true));  
 *assertTrue*(cessnaVliegtuig.CheckCessnaVliegtuig(500,80.00,true));  
 *assertFalse*(cessnaVliegtuig.CheckCessnaVliegtuig(1050,40.00,true));  
 }  
}

Hiermee test ik de volgende methode (van het type boolean, waarin de waarde van D = <Decision> wordt bepaald):

public class CessnaVliegtuig {  
  
 public static double *MaximaalGewicht*;  
 private Boolean Toestemmingtoren;  
 private double Snelheid;  
  
  
 public CessnaVliegtuig(double MaximaalGewicht, double Snelheid, Boolean Toestemmingtoren) {  
 this.*MaximaalGewicht* = MaximaalGewicht;  
 this.Snelheid = Snelheid;  
 this.Toestemmingtoren = Toestemmingtoren;  
 }  
  
 public boolean CheckCessnaVliegtuig(double Gewicht, double Snelheid, Boolean Toestemmingtoren) {  
 return Toestemmingtoren && Snelheid > 50 || Gewicht < 1000;  
 }  
}

**EQUIVALENTIEKLASSEN EN RANDWAARDEN**

Voor het nemen van een besluit in de software heb ik een domein dat bestaat uit de volgende equivalentieklassen (maximaal 3 te testen):

|  |  |  |
| --- | --- | --- |
|  | Logische weergave | Besluit |
| Equivalentieklasse 1 | Viegtuig < 1000kg | 100 euro |
| Equivalentieklasse 2 | 1000 <= vliegtuig < 5000kg | 500 euro |
| Equivalentieklasse 3 | Vliegtuig >= 5000kg | 2500 euro |

Geef nu alle relevante fysieke testgevallen (incl. de meest intensieve test van randwaarden) en geef voor elk testgeval aan voor welke equivalentieklasse dit een test is (als je de laatste regels niet meer nodig hebt, kun je deze velden vullen met een spatie):

|  |  |  |
| --- | --- | --- |
| Equivalentieklasse | Fysiek testgeval | Besluit |
| Viegtuig < 1000kg | 999 kg | 100 |
| 1000 <= vliegtuig < 5000kg | 1000 kg | 500 |
| 1000 <= vliegtuig < 5000kg | 1001 kg | 500 |
| 1000 <= vliegtuig < 5000kg | 4999 kg | 500 |
| Vliegtuig >= 5000kg | 5000 kg | 2500 |
| Vliegtuig >= 5000kg | 5001 kg | 2500 |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |
| <Equivalentieklasse> | <Testgeval> | <Besluit> |

**EQUIVALENTIEKLASEN EN RANDWAARDEN (JUNIT)**

Ik heb de testcases voor Equivalentieklassen en randwaarden omgezet naar de volgende JUnit-code om mijn software voor deze beslissing te testen.

import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class PrijsMetGewichtTest {  
  
 @Test  
 void getPrijsMetGewicht() {  
  
 double gewicht999 = 999;  
 double gewicht4500 = 4500;  
 double gewicht5001= 5001;  
  
 *assertEquals*(100.00, new PrijsMetGewicht(gewicht999).*GetPrijsMetGewicht*());  
 *assertEquals*(500.00, new PrijsMetGewicht(gewicht4500).*GetPrijsMetGewicht*());  
 *assertEquals*(2500.00, new PrijsMetGewicht(gewicht5001).*GetPrijsMetGewicht*());  
  
 }  
}

Hiermee test ik de volgende methode (waarin de besluiten op basis van de equivalentieklassen worden bepaald):

public class PrijsMetGewicht {  
  
 private static double *Gewicht*;  
  
 public PrijsMetGewicht(double Gewicht) {  
 this.*Gewicht* = Gewicht;  
 }  
  
 public static double GetPrijsMetGewicht() {  
  
 if (*Gewicht* < 1000) {  
 return 100.00;  
 } else if (*Gewicht* < 5000) {  
 return 500.00;  
 } else {  
 return 2500;  
 }  
 }  
}

**PAIRWISE TESTING**

Voor Pairwise Testing maak ik gebruik van de volgende parameters (op basis waarvan een besluit wordt genomen; vereenvoudig tot een set van 4 parameters die ieder 3, 2, 2 en 2 mogelijkheden hebben (anders wordt handmatig testen met Pairwise Testing veel te ingewikkeld; vul een spatie in, als je een optie niet wilt gebruiken):

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Optie 1** | **Optie 2** | **Optie 3** |
| MaximaalGewicht | <1000 | 1000 - < 5000 | >5000 |
| Passagiers | Minder dan 2 | Meer dan 2 | <Optie 3> |
| WelOfNietAfhandelen | Wel | Niet | <Optie 3> |
| NederlandsVliegtuig | Wel | Niet | <Optie 3> |

Na de tussenstappen in het proces heb ik de volgende testtabel met fysieke testgevallen opgesteld (vul spaties in, als er teveel testgevallen in de tabel staan voor jouw casus):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **<Parameter** 1> | **<Parameter** 2> | **<Parameter** 3> | **<Parameter** 4> | **Decision** |
| 999 | Minder dan 2 | Wel | Niet | 900 |
| 999 | Meer dan 2 | Niet | wel | 181,50 |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| 1001 | Minder dan 2 | Niet | Wel | 605 |
| 1001 | Meer dan 2 | Wel | Niet | 1550 |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| 5001 | Minder dan 2 | Niet | Niet | 2500 |
| 5001 | Meer dan 2 | Wel | Wel | 5505,50 |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |
| <waarde> | <waarde> | <waarde> | <waarde> | <waarde> |

**PAIRWISE TESTING (JUNIT)**

Ik heb de testcases voor Pairwise Testing omgezet naar de volgende JUnit-code om mijn software voor deze beslissing te testen.

Afbeelding met tekst, schermafbeelding, monitor

Automatisch gegenereerde beschrijving

Hiermee test ik de volgende methode (waarin het besluit op basis van de meegegeven parameters wordt bepaald):

Afbeelding met tekst, monitor, schermafbeelding, computer

Automatisch gegenereerde beschrijving