**ET.02.03 Segundo Problema**

In its new public transport plan of the JCCM, we have been asked to implement an application that determines the ticket price of the transport, depending on the health and age conditions of a person, and depending on the state of the pandemic. In this sense, the AI of Castilla-La Mancha will be monitored (we can assume it to be “constant” during the execution of the project, which will require an initialization of the environment), with a reduction of seats as follows: if it is less than 100 there will be no space restrictions (level 0), if it is between 100 and 200 (level 1) the capacity of the means of transport is reduced to 80%, if it is between 201 and 300 (level 2), it is reduced to 60%, if it is between 301 and 500 (level 3) the capacity is reduced to 40%, and if it is higher than 501 (level 4) to 30%. In order to avoid unnecessary movements, an increase in the ticket price will be established. For this purpose, the following rules will be used to calculate the ticket price:

• Regardless of AI status, a person who is ill, has recent contact within the last 10 days with infected persons, or has suspected symptoms of COVID will not be allowed to travel. A person with a COVID passport and not ill may travel if space is available, regardless of their occupation type.

• If there are no space restrictions (level 0), any person, regardless of age, may travel, and will receive a 60% discount if under 23 years of age, and an 80% discount if over 65 years of age.

• In Level 1, those under 23 years of age will have a 30% discount, and those over 65 will have a 50% discount. No transportation priorities are established, but places are reduced, so that a price will only be given (consider implement and throw some kind of exception) if there are places available.

• In Level 2, those under 23 years of age will have no discount, and those over 65 years of age will have an increase of 20%. At this level, of the possible reduced capacity, 60% of the places are reserved for professionals with essential professions.

• At Level 3, those under 23 years of age will have a 20% surcharge, and those over 65 years of age will have a 50% increase. At this capacity level, 80% of the available reduced space is reserved for essential professionals.

• At Level 4, those under 23 years of age will have a 50% surcharge, and those over 65 years of age will not be allowed to travel. A 90% space is reserved for people with essential professions.

**It is requested:**

1) Write, at least the pseudocode of the identified method.

2) Identify the variables that must be considered to test the method.

3) Identify the test values for each one of the variables previously identified, specifying the technique used to obtain each of those values).

4) Calculate the maximum possible number of test cases that could be generated from the test values.

5) Define some test suites using each use.

6) Define test suits to achieve pairwise coverage by using the proposed algorithm in Lectures. You can check the results by means of the software PICT1.

7) For code snippets that include decisions, propose a set of test cases to achieve coverage of decisions.

8) For code snippets that include decisions, propose test case sets to achieve MC/DC coverage.

9) Comment on the results of the number of test cases obtained in section 4, 5, and 6, as well as the execution of the oracles: what could be said about the coverage achieved?

**1) Write, at least the pseudocode of the identified method.**

**public** **double** calculateTicketCost(Person person) **throws** NoSeatsAvailableException, NotHealthyException {

**double** price = 0;

**if**(!person.isCOVIDPassport() || person.isIll()) {

**throw** **new** NotHealthyException("You are not allowed to travel in those conditions");

}

**if**(person.isEssentialProfession()) {

**if**(**this**.level == 2 || **this**.level == 3 || **this**.level == 4) {

**if**((**this**.essentialProfessionsSeats-1)>=0) {

**this**.essentialProfessionsSeats -= 1;

}**else** **if**((**this**.normalNumberSeats-1)>=0) {

**this**.normalNumberSeats -= 1;

}**else** {

**throw** **new** NoSeatsAvailableException("There are no seats available");

}

}

}**else** {

**if**((**this**.normalNumberSeats-1)<0) {

**throw** **new** NoSeatsAvailableException("There are no seats available");

}

**this**.normalNumberSeats -= 1;

}

**if**(**this**.level == 0) {

**if**(person.getAge() < 23) {

price = 0.4 \* **this**.ticketPrice;

}**else** **if**(person.getAge() > 65) {

price = 0.2 \* **this**.ticketPrice;

}**else** {

price = **this**.ticketPrice;

}

}**else** **if**(**this**.level == 1){

**if**(person.getAge() < 23) {

price = 0.7 \* **this**.ticketPrice;

}**else** **if** (person.getAge() > 65) {

price = 0.5 \* **this**.ticketPrice;

}**else** {

price = **this**.ticketPrice;

}

}**else** **if**(**this**.level == 2) {

**if** (person.getAge() > 65) {

price = 1.2 \* **this**.ticketPrice;

}**else** {

price = **this**.ticketPrice;

}

}**else** **if**(**this**.level == 3) {

**if**(person.getAge() < 23) {

price = 1.2 \* **this**.ticketPrice;

}**else** **if** (person.getAge() > 65) {

price = 1.5 \* **this**.ticketPrice;

}**else** {

price = **this**.ticketPrice;

}

}**else** **if**(**this**.level == 4) {

**if**(person.getAge() < 23) {

price = 1.5 \* **this**.ticketPrice;

}**else** **if** (person.getAge() > 65) {

**throw** **new** NoSeatsAvailableException("You are not allowed to travel");

}**else** {

price = **this**.ticketPrice;

}

}

**return** price;

}

**2) Identify the variables that must be considered to test the method.**

There are variables that correspond to different classes:

* ***Transport***

- level: According to the level we will have one specific discount.

- availableNumberSeats: Total number of seats after reduction.

- essentialProfessionSeats: Seats that are reserved for professionals.

* ***Person***
* age: Determine the discount applying for the group age.
* essentialProfession: Determine if it is an essential profession or not.
* COVIDpassport: Represents the COVID passport for travelling
* Ill: Represents if the person is ill or has symptoms

**3) Identify the test values for each one of the variables previously identified, specifying the technique used to obtain each of those values).**

Integer = [-2147483648, 2147483647]

**level** = (-Inf, -2147483648), [-2147483648, 0), [0], [1], [2], [3], [4], (4, 2147483647], (2147483647, +Inf)

**availableNumberSeats** = (-Inf, -2147483648), [-2147483648, 0), [0, 2147483647], (2147483647, +Inf)

**essentialProfessionSeats** = (-Inf, -2147483648), [-2147483648, 0), [0, 2147483647], (2147483647, +Inf)

**age** = (-Inf, -2147483648), [-2147483648, 23), [23, 65), [65, 2147483647], (2147483647, +Inf)

**profession** = [True], [False]

**COVIDpassport** = [True], [False]

**Ill** = [True], [False]

|  |  |  |  |
| --- | --- | --- | --- |
| **VARIABLES** | **ERROR GUESSING** | **BOUNDARY VALUES** | |
| **level** | -2247483648,0,2247483648 | -2147483648,0,1,2,3,4 2147483647 | -2147483649,  -2147483648, -2147483647,  -1,0,1,2,3,4,5  2147483646,  2147483647,  2147483648, |
| **normalNumberSeats** | -2247483648,0,2247483648 | -2147483648,0, 2147483647 | -2147483649,  -2147483648, -2147483647,  -1,0,1  2147483646,  2147483647,  2147483648, |
| **essentialProfessionSeats** | -2247483648,0,2247483648 | -2147483648,0, 2147483647 | -2147483649,  -2147483648, -2147483647,  -1,0,1  2147483646,  2147483647,  2147483648, |
| **age** | -2247483648,0,30,69 2247483648 | -2147483648,23,65, 2147483647 | -2147483649,  -2147483648, -2147483647,  22,23,24,64,65,66,  2147483646, 2147483647, 2147483648, |
| **profession** | True, False | True, False | True, False |
| **COVIDpassport** | True, False | True, False | True, False |
| **Ill** | True, False | True, False | True, False |

**4) Calculate the maximum possible number of test cases that could be generated from the test values.**

In this case, as we are dealing with integers that can get any value from -Inf to +Inf, we can have infinite number of test cases.

If we consider the test values previously generated:

**level: -**2247483648,-2147483649,-2147483648,-2147483647,-1,0,1,2,3,4,5,2147483646, 2147483647,2147483648. ***(There are 14 possible values).***

**normalNumberSeats: -**2247483648,-2147483649,-2147483648,-2147483647,-1,0,1, 2147483646,2147483647,2147483648. ***(There are 10 possible values).***

**essentialProfessionSeats: -**2247483648,-2147483649,-2147483648,-2147483647,-1,0,1, 2147483646,2147483647,2147483648. ***(There are 10 possible values).***

**age:-**2247483648,-2147483649,-2147483648,- 2147483647,22,23,24,64,65,66,2147483646, 2147483647,2147483648,0,30,69. ***(There are 16 possible values).***

**profession:** True,False. ***(There are 2 possible values).***

**COVIDpassport:**True,False. ***(There are 2 possible values).***

**ill:**True,False. ***(There are 2 possible values).***

Maximum number of combinations: 14\*10\*10\*16\*2\*2\*2 = 179000 test cases. This test cases will represent most of the scenarios.

**5) Define some test suites using each use.**

{level,normalSeats,essentialProfessionSeats,age,essentialProfession,COVIDpassport,ill}

Test suite 1: {0, 1, 1, 23, False, True, False}

Test suite 2: {1, 0, 1, 65, False, True, True}

Test suite 3: {2, 0, 0, 22, False, True, False}

Test suite 4: {3, 1, 0, 66, False, True, True}

Test suite 5: {4, -1, 0, 64, True, False, False}

**6) Define test suits to achieve pairwise coverage by using the proposed algorithm in Lectures. You can check the results by means of the software PICT1.**

Firstly, we have to consider the values each variable can take. Once this is done, we must achieve pairwise coverage. Due to the number of variables and the number of values they can take, we will make use software PICT1. In the file test-results there are a total of 224 test cases. The solution is in the file named “pairwise-result”.

**7) For code snippets that include decisions, propose a set of test cases to achieve coverage of decisions.**

**if**(!person.isCOVIDPassport() || person.isIll()) {

**throw** **new** NotHealthyException("You are not allowed to travel in those conditions");

}

|  |  |  |
| --- | --- | --- |
| **COVIDpassport** | **ill** | **DECISION** |
| False | False | NotHealthyException |

Test suite 1: {?, ?, ?, ?, ?, ?, False, False}

**if**(person.isEssentialProfession()) {

**if**(**this**.level == 2 || **this**.level == 3 || **this**.level == 4) {

**if**((**this**.essentialProfessionsSeats-1)>=0) {

**this**.essentialProfessionsSeats -= 1;

}**else** **if**((**this**.normalSeats-1)>=0) {

**this**.normalSeats -= 1;

}**else** {

**throw** **new** NoSeatsAvailableException("There are no seats available");

}

}**else** {

**if**((**this**.normalSeats-1)>=0) {

**this**.normalSeats -= 1;

}**else** {

**throw** **new** NoSeatsAvailableException("There are no seats available");

}

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **eP** | **level** | **ePS** | **nS** | **DECISION** |
| True | 2 | 1 | 0 | ePS = ePS-1 |
| True | 3 | 0 | 1 | nS = nS-1 |
| True | 4 | 0 | 0 | NoSeatsAvailableException |
| True | 0 | 0 | 1 | nS = nS-1 |
| True | 1 | 0 | 0 | NoSeatsAvailableException |

Test suite 2: {2, 0, 1, ?, True, True, False}

Test suite 3: {3, 1, 0, ?, True, True, False}

Test suite 4: {4, 0, 0, ?, True, True, False}

Test suite 5: {0, 1, 0, ?, True, True, False}

Test suite 6: {1, 0, 0, ?, True, True, False}

**else** {

**if**((**this**.normalSeats-1)<0) {

**throw** **new** NoSeatsAvailableException("There are no seats available");

}

**this**.normalSeats -= 1;

}

|  |  |  |
| --- | --- | --- |
| **level** | **nS** | **DECISION** |
| 1 | 1 | nS = nS-1 |
| 2 | 0 | NoSeatsAvailableException |

Test suite 7: {1, 1, ?, ?, False, True, False}

Test suite 8: {2, 0, ?, ?, False, True, False}

**if**(person.getAge() < 23) {

price = 0.4 \* **this**.ticketPrice;

}**else** **if**(person.getAge() > 65) {

price = 0.2 \* **this**.ticketPrice;

}**else** {

price = **this**.ticketPrice;

}

|  |  |  |
| --- | --- | --- |
| **age** | **DECISION** | **EXPECTED** |
| 23 | False, False, True | ticketPrice |
| 65 | False, False, True | ticketPrice |
| 66 | False, True, False | ticketPrice \* 0.2 |
| 22 | True, False, False | ticketPrice \* 0.4 |

Test suite 9: {0, ?, ?, 65, ?, ?, True, False}

Test suite 10: {0, ?, ?, 66, ?, ?, True, False}

Test suite 11: {0, ?, ?, 22, ?, ?, True, False}

**if**(person.getAge() < 23) {

price = 0.7 \* **this**.ticketPrice;

}**else** **if** (person.getAge() > 65) {

price = 0.5 \* **this**.ticketPrice;

}**else** {

price = **this**.ticketPrice;

}

|  |  |  |
| --- | --- | --- |
| **age** | **DECISION** | **EXPECTED** |
| 23 | False, False, True | ticketPrice |
| 65 | False, False, True | ticketPrice |
| 66 | False, True, False | ticketPrice \* 0.5 |
| 22 | True, False, False | ticketPrice \* 0.7 |

Test suite 12: {1, ?, ?, 65, ?, ?, True, False}

Test suite 13: {1, ?, ?, 66, ?, ?, True, False}

Test suite 14: {1, ?, ?, 22, ?, ?, True, False}

**if** (person.getAge() > 65) {

price = 1.2 \* **this**.ticketPrice;

}**else** {

price = **this**.ticketPrice;

}

|  |  |  |
| --- | --- | --- |
| **age** | **DECISION** | **EXPECTED** |
| 23 | False, True | ticketPrice |
| 65 | False, True | ticketPrice |
| 66 | True, False | ticketPrice \* 1.2 |
| 22 | False, True | ticketPrice |

Test suite 15: {2, ?, ?, 65, ?, ?, True, False}

Test suite 16: {2, ?, ?, 66, ?, ?, True, False}

**if**(person.getAge() < 23) {

price = 1.2 \* **this**.ticketPrice;

}**else** **if** (person.getAge() > 65) {

price = 1.5 \* **this**.ticketPrice;

}**else** {

price = **this**.ticketPrice;

}

|  |  |  |
| --- | --- | --- |
| **age** | **DECISION** | **EXPECTED** |
| 23 | False, False, True | ticketPrice |
| 65 | False, False, True | ticketPrice |
| 66 | False, True, False | ticketPrice \* 1.5 |
| 22 | True, False, False | ticketPrice \* 1.2 |

Test suite 17: {3, ?, ?, 65, ?, ?, True, False}

Test suite 18: {3, ?, ?, 66, ?, ?, True, False}

Test suite 19: {3, ?, ?, 22, ?, ?, True, False}

**if**(person.getAge() < 23) {

price = 1.5 \* **this**.ticketPrice;

}**else** **if** (person.getAge() > 65) {

**throw** **new** NoSeatsAvailableException("You are not allowed to travel");

}**else** {

price = **this**.ticketPrice;

}

|  |  |  |
| --- | --- | --- |
| **age** | **DECISION** | **EXPECTED** |
| 23 | False, False, True | ticketPrice |
| 65 | False, False, True | ticketPrice |
| 66 | False, True, False | NoSeatsAvailableException |
| 22 | True, False, False | ticketPrice \* 1.5 |

Test suite 20: {4, ?, ?, 65, ?, ?, True, False}

Test suite 21: {4, ?, ?, 66, ?, ?, True, False}

Test suite 22: {4, ?, ?, 22, ?, ?, True, False}

**8) For code snippets that include decisions, propose test case sets to achieve MC/DC coverage.**

**if**(!person.isCOVIDPassport() || person.isIll()) {

**throw** **new** NotHealthyException("You are not allowed to travel in those conditions");

}

|  |  |  |  |
| --- | --- | --- | --- |
| **!Person.isCOVIDPassport()** | **Person.isIll()** | **DECISION** | **DOMINANT CONDITION** |
| True | True | True | A,B |
| True | False | True | A |
| False | True | True | B |
| False | False | False | A,B |

Test suite 1: {?, ?, ?, ?, ?, ?, True, True}

Test suite 2: {?, ?, ?, ?, ?, ?, True, False}

Test suite 3: {?, ?, ?, ?, ?, ?, False, True}

**if**(person.isEssentialProfession()) {

**if**(**this**.level == 2 || **this**.level == 3 || **this**.level == 4) {

**if**((**this**.essentialProfessionsSeats-1)>=0) {

**this**.essentialProfessionsSeats -= 1;

}**else** **if**((**this**.normalSeats-1)>=0) {

**this**.normalSeats -= 1;

}**else** {

**throw** **new** NoSeatsAvailableException("There are no seats available");

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **person.isEssentialProfession** | **this.level == 2 or this.level == 3 or this.level == 4** | **ePS-1 >= 0** | **DECISION** | **DOMINANT CONDITION** |
| True | True | True | True | A,B,C |
| True | True | False | False | C |
| True | False | True | False | B |
| True | False | False | False | B |
| False | True | True | False | A |
| False | True | False | False | A |
| False | False | True | False | A |
| False | False | False | False | A |

Test suite 4: {2, ?, 1, ?, True, True, False}

Test suite 5: {2, ?, 0, ?, True, True, False}

Test suite 6: {3, ?, 1, ?, True, True, False}

Test suite 7: {4, ?, 1, ?, False, True, False}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **person.isEssentialProfession** | **this.level == 2 or this.level == 3 or this.level == 4** | **nS-1 >= 0** | **DECISION** | **DOMINANT CONDITION** |
| True | True | True | True | A,B,C |
| True | True | False | False | C |
| True | False | True | False | B |
| True | False | False | False | B |
| False | True | True | False | A |
| False | True | False | False | A |
| False | False | True | False | A |
| False | False | False | False | A |

Test suite 8: {2, 1, 0, ?, True, True, False}

Test suite 9: {2, 0, 0, ?, True, True, False}

Test suite 10: {3, 1, 0, ?, True, True, False}

Test suite 11: {4, 1, 0, ?, False, True, False}

**else** {

**if**((**this**.normalSeats-1)>=0) {

**this**.normalSeats -= 1;

}**else** {

**throw** **new** NoSeatsAvailableException("There are no seats available");

}

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **!person.isEssentialProfession** | **this.level != 2 and this.level != 3 and**  **this.level != 4** | **nS-1 >= 0** | **DECISION** | **DOMINANT CONDITION** |
| True | True | True | True | A,B,C |
| True | True | False | False | C |
| True | False | True | False | B |
| True | False | False | False | B |
| False | True | True | False | A |
| False | True | False | False | A |
| False | False | True | False | A |
| False | False | False | False | A |

Test suite 12: {0, 1, ?, ?, False, True, False}

Test suite 13: {0, 0, ?, ?, False, True, False}

Test suite 14: {2, 1, ?, ?, False, True, False}

Test suite 15: {1, 1, ?, ?, True, True, False}

**else** {

**if**((**this**.normalSeats-1)>=0) {

**this**.normalSeats -= 1;

}**else** {

**throw** **new** NoSeatsAvailableException("There are no seats available");

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| **!person.isEssentialProfession** | **nS-1 >= 0** | **DECISION** | **DOMINANT CONDITION** |
| True | True | True | A,B |
| True | False | False | B |
| False | True | False | A |
| False | False | False | A |

Test suite 16: {?, 1, ?, ?, True, True, False}

Test suite 17: {?, 0, ?, ?, True, True, False}

Test suite 18: {?, 1, ?, ?, False, True, False}

**if**(**this**.level == 0) {

**if**(person.getAge() < 23) {

price = 0.4 \* **this**.ticketPrice;

}**else** **if**(person.getAge() > 65) {

price = 0.2 \* **this**.ticketPrice;

}**else** {

price = **this**.ticketPrice;

}

|  |  |  |  |
| --- | --- | --- | --- |
| **this.level == 0** | **Person.getAge() < 23** | **DECISION** | **DOMINANT CONDITION** |
| True | True | True | A,B |
| True | False | False | B |
| False | True | False | A |
| False | False | False | A |

Test suite 19: {0, ?, ?, 22, ?, True, False}

Test suite 20: {0, ?, ?, 23, ?, True, False}

Test suite 21: {1, ?, ?, 22, ?, True, False}

|  |  |  |  |
| --- | --- | --- | --- |
| **this.level == 0** | **Person.getAge() > 65** | **DECISION** | **DOMINANT CONDITION** |
| True | True | True | A,B |
| True | False | False | B |
| False | True | False | A |
| False | False | False | A |

Test suite 22: {0, ?, ?, 66, ?, True, False}

Test suite 23: {0, ?, ?, 65, ?, True, False}

Test suite 24: {1, ?, ?, 66, ?, True, False}

**else** **if**(**this**.level == 1){

**if**(person.getAge() < 23) {

price = 0.7 \* **this**.ticketPrice;

}**else** **if** (person.getAge() > 65) {

price = 0.5 \* **this**.ticketPrice;

}**else** {

price = **this**.ticketPrice;

}

|  |  |  |  |
| --- | --- | --- | --- |
| **this.level == 1** | **Person.getAge() < 23** | **DECISION** | **DOMINANT CONDITION** |
| True | True | True | A,B |
| True | False | False | B |
| False | True | False | A |
| False | False | False | A |

Test suite 25: {1, ?, ?, 22, ?, True, False}

Test suite 26: {1, ?, ?, 23, ?, True, False}

Test suite 27: {2, ?, ?, 22, ?, True, False}

|  |  |  |  |
| --- | --- | --- | --- |
| **this.level == 1** | **Person.getAge() > 65** | **DECISION** | **DOMINANT CONDITION** |
| True | True | True | A,B |
| True | False | False | B |
| False | True | False | A |
| False | False | False | A |

Test suite 28: {1, ?, ?, 66, ?, True, False}

Test suite 29: {1, ?, ?, 65, ?, True, False}

Test suite 30: {2, ?, ?, 66, ?, True, False}

}**else** **if**(**this**.level == 2) {

**if** (person.getAge() > 65) {

price = 1.2 \* **this**.ticketPrice;

}**else** {

price = **this**.ticketPrice;

}

|  |  |  |  |
| --- | --- | --- | --- |
| **this.level == 2** | **Person.getAge() > 65** | **DECISION** | **DOMINANT CONDITION** |
| True | True | True | A,B |
| True | False | False | B |
| False | True | False | A |
| False | False | False | A |

Test suite 31: {2, ?, ?, 66, ?, True, False}

Test suite 32: {2, ?, ?, 65, ?, True, False}

Test suite 33: {3, ?, ?, 66, ?, True, False}

**else** **if**(**this**.level == 3) {

**if**(person.getAge() < 23) {

price = 1.2 \* **this**.ticketPrice;

}**else** **if** (person.getAge() > 65) {

price = 1.5 \* **this**.ticketPrice;

}**else** {

price = **this**.ticketPrice;

}

|  |  |  |  |
| --- | --- | --- | --- |
| **this.level == 3** | **Person.getAge() < 23** | **DECISION** | **DOMINANT CONDITION** |
| True | True | True | A,B |
| True | False | False | B |
| False | True | False | A |
| False | False | False | A |

Test suite 34: {3, ?, ?, 22, ?, True, False}

Test suite 35: {3, ?, ?, 23, ?, True, False}

Test suite 36: {4, ?, ?, 22, ?, True, False}

|  |  |  |  |
| --- | --- | --- | --- |
| **this.level == 3** | **Person.getAge() > 65** | **DECISION** | **DOMINANT CONDITION** |
| True | True | True | A,B |
| True | False | False | B |
| False | True | False | A |
| False | False | False | A |

Test suite 37: {3, ?, ?, 66, ?, True, False}

Test suite 38: {3, ?, ?, 65, ?, True, False}

Test suite 39: {4, ?, ?, 66, ?, True, False}

**else**{

**if**(person.getAge() < 23) {

price = 1.5 \* **this**.ticketPrice;

}**else** **if** (person.getAge() > 65) {

**throw** **new** NoSeatsAvailableException("You are not allowed to travel");

}**else** {

price = **this**.ticketPrice;

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| **else** | **Person.getAge() < 23** | **DECISION** | **DOMINANT CONDITION** |
| True | True | True | A,B |
| True | False | False | B |
| False | True | False | A |
| False | False | False | A |

Test suite 40: {4, ?, ?, 22, ?, True, False}

Test suite 41: {4, ?, ?, 23, ?, True, False}

Test suite 42: {0, ?, ?, 22, ?, True, False}

|  |  |  |  |
| --- | --- | --- | --- |
| **else** | **Person.getAge() > 65** | **DECISION** | **DOMINANT CONDITION** |
| True | True | True | A,B |
| True | False | False | B |
| False | True | False | A |
| False | False | False | A |

Test suite 43: {4, ?, ?, 66, ?, True, False}

Test suite 44: {4, ?, ?, 65, ?, True, False}

Test suite 45: {0, ?, ?, 66, ?, True, False}

**9) Comment on the results of the number of test cases obtained in section 4, 5, and 6, as well as the execution of the oracles: what could be said about the coverage achieved?**