# CS608
# Software Testing

Dr. Stephen Brown

Room Eolas 116

stephen.brown@mu.ie

# Tutorial: Lab 1

- Lab Contents:
  - Running the examples in chapter 1
  - Quiz
- Any questions?

# CS608

## Testing with Equivalence Partitions

(Essentials of Software Testing, Chapter 2)

# Introduction

- Today, we will consider the simplest black-box testing technique of “equivalence partitions”, using a worked unit test example
- Often referred to as EP

# Introduction

- Today, we will consider the simplest black-box testing technique of “equivalence partitions”, using a worked unit test example
- Often referred to as EP
- We will develop tests using the steps identified previously:
  1. Analysis
  2. Develop test coverage items (TCI)
  3. Develop test cases (TC)
  4. Test design verification
  5. Test implementation
  6. Test execution
  7. Examination of test results

# Testing with Equivalence Partitions

- Goal is to verify that the software works correctly:
  - For each different type of processing
  - By using at least one representative input **value** (for each type of processing)
  - And producing at least one representative output **value** (for each type of processing)
- To identify these **values**, equivalence partitions are used
- Today: a worked example
- Next week: examine the topic in more detail after the lab

# Testing with Equivalence Partitions

**Definition**:
an equivalence partition
is a range of discrete values
for an input, or an output,
for which the specification states
equivalent processing

# Example

- The program check as described previously uses a class **OnlineSales** to implement its core functionality
- This class contains a static method giveDiscount() which is defined below (Javadoc)
- **Note: you do not need the source code to develop black-box tests**
- We introduce test techniques using static methods that do not require an object to be instantiated through a constructor
- (Testing object-oriented software looks at testing instance methods)
- Explaining this example takes much longer than doing it in practice!

giveDiscount()

**Status giveDiscount(long bonusPoints, boolean goldCustomer)**

**Inputs**

**bonusPoints:** the number of bonusPoints the customer has accumulated

**goldCustomer:** true for a Gold Customer

**Outputs**

**return value:**

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

Status is defined as follows:

```
enum Status { FULLPRICE, DISCOUNT, ERROR };
```

# Error Handling

- For simplicity, this code uses **in-band** error handling
  - The errors are reported using the same mechanism as normal results (the return value)
- We will look at testing code that uses Java exceptions for error handling later in the module

# Step 1. Analysis

- Analyse the specification to identify the equivalence partitions
- Based on the principle of equivalent processing
- Two stages:
  - first identify the **natural ranges** for each parameter

# Step 1. Analysis

- Analyse the specification to identify the equivalence partitions
- Based on the principle of equivalent processing
- Two stages:
  - first identify the **natural ranges** for each parameter
  - then identify the **specification-based ranges** (or equivalence partitions)

# Natural Ranges

- Natural ranges are based on the **types** of the input parameters, and of the return value(s)

# Natural Ranges

- Natural ranges are based on the **types** of the input parameters, and of the return value(s)
- Use **value lines** for each input and output to help with the analysis

# Natural Ranges

- Natural ranges are based on the **types** of the input parameters, and of the return value(s)
- Use **value lines** for each input and output to help with the analysis
- **Value line:** graphical representation of a range of values
  - The minimum value is always placed to the left
  - And the maximum value to the right

# Natural Ranges

- Natural ranges are based on the **types** of the input parameters, and of the return value(s)
- Use **value lines** for each input and output to help with the analysis
- **Value line:** graphical representation of a range of values
  - The minimum value is always placed to the left
  - And the maximum value to the right
- These value lines assist in ensuring that there are no **gaps** or **overlaps** in the equivalence partitions once they have been identified

# Recap: signature & specification

**giveDiscount(long bonusPoints, boolean goldCustomer)**

FULLPRICE if bonusPoints$\leq$120 and not a goldCustomer
FULLPRICE if bonusPoints$\leq$80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

**giveDiscount(long bonusPoints, boolean goldCustomer)**

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

# bonusPoints

- The first input parameter, bonusPoints:

**giveDiscount(long bonusPoints, boolean goldCustomer)**

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

# bonusPoints

- The first input parameter, bonusPoints:

**SPECIAL NOTE**

In Java (and most languages)
case is important

So be very careful to get
the case right
during testing

**giveDiscount(long bonusPoints, boolean goldCustomer)**

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

# bonusPoints

- The first input parameter, bonusPoints:
  - Is a “long”

# bonusPoints

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

- The first input parameter, bonusPoints:
  - Is a “long”
  - Has one natural range with $2^{64}$ values

# bonusPoints

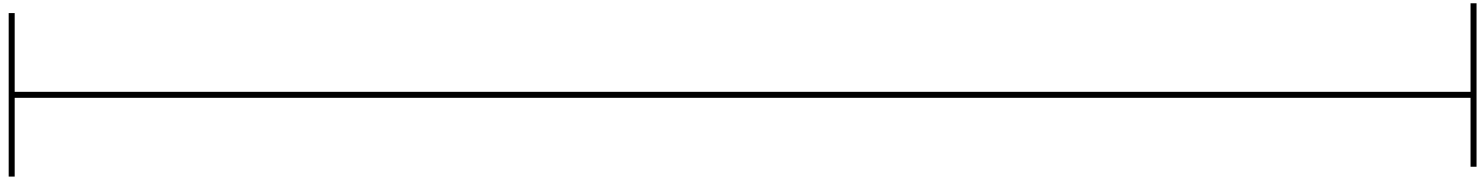
FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

- The first input parameter, bonusPoints:
  - Is a “long”
  - Has one natural range with $2^{64}$ values
- Value line:

# bonusPoints

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

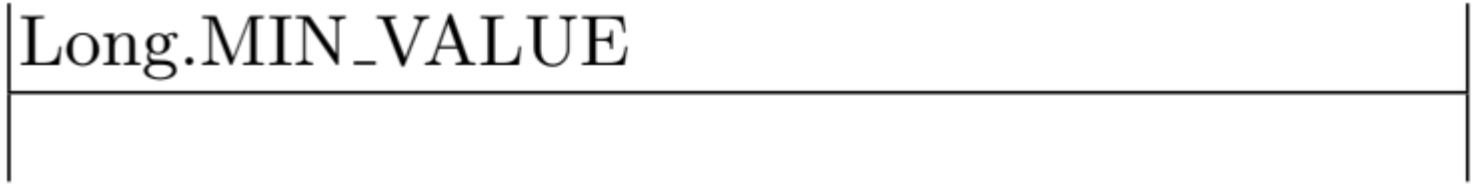
- The first input parameter, bonusPoints:
  - Is a “long”
  - Has one natural range with $2^{64}$ values
- Value line:



# bonusPoints

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

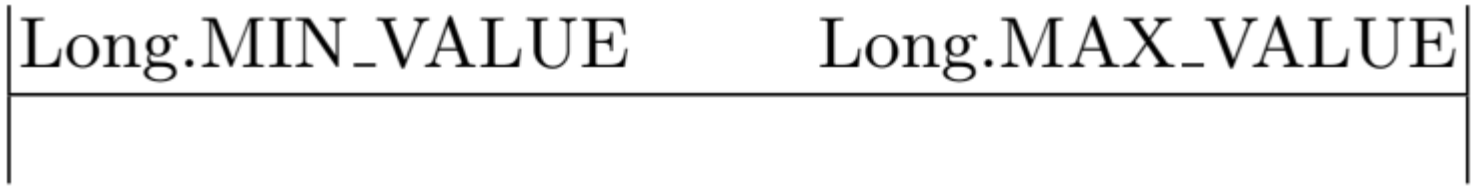
- The first input parameter, bonusPoints:
  - Is a “long”
  - Has one natural range with $2^{64}$ values
- Value line:



# bonusPoints

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

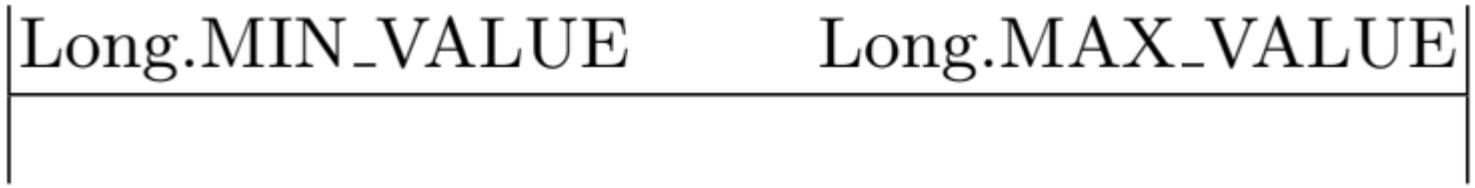
- The first input parameter, bonusPoints:
  - Is a “long”
  - Has one natural range with $2^{64}$ values
- Value line:



# bonusPoints

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

- The first input parameter, bonusPoints:
  - Is a “long”
  - Has one natural range with $2^{64}$ values
- Value line:

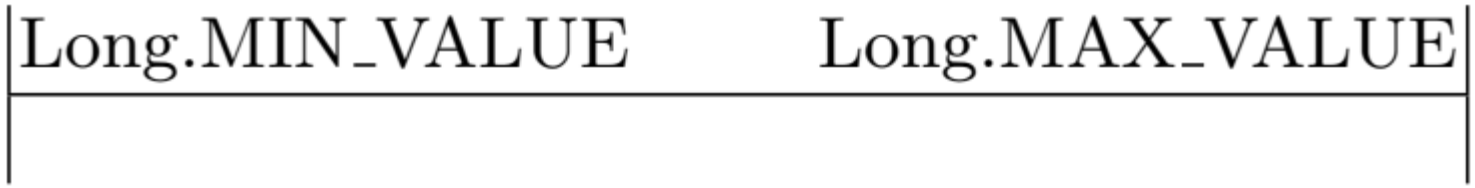



- bonusPoints may hold any value from Long.MIN_VALUE to Long.MAX_VALUE

# bonusPoints

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

- The first input parameter, bonusPoints:
  - Is a “long”
  - Has one natural range with $2^{64}$ values
- Value line:

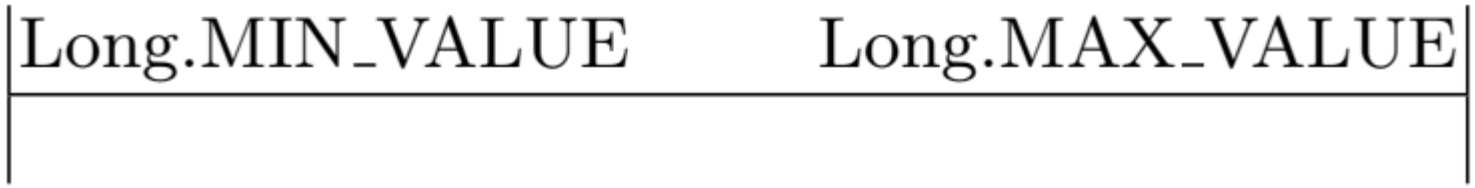



- bonusPoints may hold any value from Long.MIN_VALUE to Long.MAX_VALUE
- Alternative typed syntax: [Long.MIN_VALUE..Long.MAX_VALUE]

# bonusPoints

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

- The first input parameter, bonusPoints:
  - Is a “long”
  - Has one natural range with $2^{64}$ values
- Value line:




- bonusPoints may hold any value from Long.MIN_VALUE to Long.MAX_VALUE
- Alternative typed syntax: [Long.MIN_VALUE..Long.MAX_VALUE]
- Note: use symbolic constants where possible for the analysis

giveDiscount(long bonusPoints, boolean goldCustomer)

# goldCustomer

- The second input parameter, goldCustomer:

giveDiscount(long bonusPoints, boolean goldCustomer)

# goldCustomer

- The second input parameter, goldCustomer:
  - Is a “boolean”

giveDiscount(long bonusPoints, boolean goldCustomer)

# goldCustomer

- The second input parameter, goldCustomer:
  - Is a “boolean”
  - Values: **true** and **false**

giveDiscount(long bonusPoints, boolean goldCustomer)

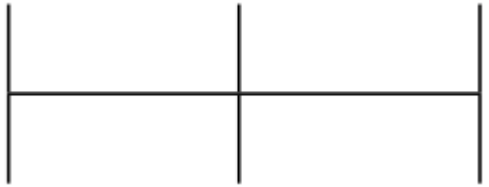
# goldCustomer

- The second input parameter, goldCustomer:
  - Is a “boolean”
  - Values: **true** and **false**
  - Boolean values are best treated as two separate ranges with one value each, as there is no natural ordering of the values true and false

giveDiscount(long bonusPoints, boolean goldCustomer)

# goldCustomer

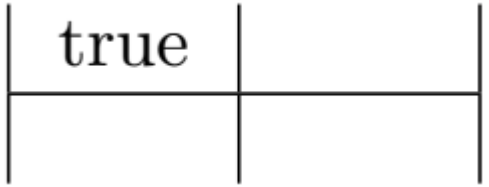
- The second input parameter, goldCustomer:
  - Is a “boolean”
  - Values: **true** and **false**
  - Boolean values are best treated as two separate ranges with one value each, as there is no natural ordering of the values true and false
- Value line:



giveDiscount(long bonusPoints, boolean goldCustomer)

# goldCustomer

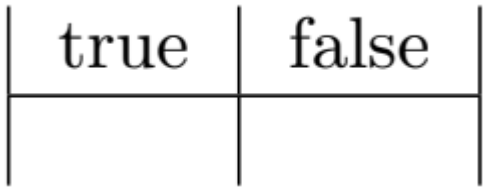
- The second input parameter, goldCustomer:
  - Is a “boolean”
  - Values: **true** and **false**
  - Boolean values are best treated as two separate ranges with one value each, as there is no natural ordering of the values true and false
- Value line:



giveDiscount(long bonusPoints, boolean goldCustomer)

# goldCustomer

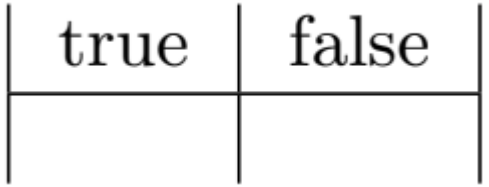
- The second input parameter, goldCustomer:
  - Is a “boolean”
  - Values: **true** and **false**
  - Boolean values are best treated as two separate ranges with one value each, as there is no natural ordering of the values true and false
- Value line:



giveDiscount(long bonusPoints, boolean goldCustomer)

# goldCustomer

- The second input parameter, goldCustomer:
  - Is a “boolean”
  - Values: **true** and **false**
  - Boolean values are best treated as two separate ranges with one value each, as there is no natural ordering of the values true and false
- Value line:

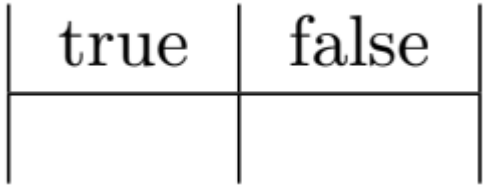



- goldCustomer has two natural ranges, each with one value

giveDiscount(long bonusPoints, boolean goldCustomer)

# goldCustomer

- The second input parameter, goldCustomer:
  - Is a “boolean”
  - Values: **true** and **false**
  - Boolean values are best treated as two separate ranges with one value each, as there is no natural ordering of the values true and false
- Value line:




- goldCustomer has two natural ranges, each with one value
- typed syntax: [true][false]

# Return Value

```
enum Status { FULLPRICE, DISCOUNT, ERROR };
```

```
Status giveDiscount(long bonusPoints, boolean goldCustomer)
```

# Return Value

```
enum Status { FULLPRICE, DISCOUNT, ERROR };
```

```
Status giveDiscount(long bonusPoints, boolean goldCustomer)
```

- Enumerated values are best treated in the same way as Boolean values, with multiple separate ranges and one value in each range

```
enum Status { FULLPRICE, DISCOUNT, ERROR };
```

```
Status giveDiscount(long bonusPoints, boolean goldCustomer)
```

# Return Value

- Enumerated values are best treated in the same way as Boolean values, with multiple separate ranges and one value in each range
- Java does define an ordering for enumerated values – ordinal() – but for testing purposes it is best to ignore this
  - For example: the implementation may change the ordering
  - Or insert extra `enum` values
  - (Unless the ordering is important)

```
enum Status { FULLPRICE, DISCOUNT, ERROR };
```

```
Status giveDiscount(long bonusPoints, boolean goldCustomer)
```

# Return Value

- Enumerated values are best treated in the same way as Boolean values, with multiple separate ranges and one value in each range
- Java does define an ordering for enumerated values – ordinal()
- But the different enumerated values often reflect different types of processing, so most effective to treat each value as a separate range

```
enum Status { FULLPRICE, DISCOUNT, ERROR };
```

```
Status giveDiscount(long bonusPoints, boolean goldCustomer)
```

# Return Value

- Enumerated values are best treated in the same way as Boolean values, with multiple separate ranges and one value in each range
- Java does define an ordering for enumerated values – ordinal()
- But the different enumerated values often reflect different types of processing, so most effective to treat each value as a separate range
- Value line:

```
enum Status { FULLPRICE, DISCOUNT, ERROR };
```

```
Status giveDiscount(long bonusPoints, boolean goldCustomer)
```

# Return Value

- Enumerated values are best treated in the same way as Boolean values, with multiple separate ranges and one value in each range
- Java does define an ordering for enumerated values – ordinal()
- But the different enumerated values often reflect different types of processing, so most effective to treat each value as a separate range
- Value line:

# Return Value

```
enum Status { FULLPRICE, DISCOUNT, ERROR };
```

```
Status giveDiscount(long bonusPoints, boolean goldCustomer)
```

- Enumerated values are best treated in the same way as Boolean values, with multiple separate ranges and one value in each range
- Java does define an ordering for enumerated values – ordinal()
- But the different enumerated values often reflect different types of processing, so most effective to treat each value as a separate range
- Value line:

```
enum Status { FULLPRICE, DISCOUNT, ERROR };
```

```
Status giveDiscount(long bonusPoints, boolean goldCustomer)
```

# Return Value

- Enumerated values are best treated in the same way as Boolean values, with multiple separate ranges and one value in each range
- Java does define an ordering for enumerated values – ordinal()
- But the different enumerated values often reflect different types of processing, so most effective to treat each value as a separate range
- Value line:

```
enum Status { FULLPRICE, DISCOUNT, ERROR };
```

```
Status giveDiscount(long bonusPoints, boolean goldCustomer)
```

# Return Value

- Enumerated values are best treated in the same way as Boolean values, with multiple separate ranges and one value in each range
- Java does define an ordering for enumerated values – ordinal()
- But the different enumerated values often reflect different types of processing, so most effective to treat each value as a separate range
- Value line:


- The return value has three natural ranges, each with one value

```
enum Status { FULLPRICE, DISCOUNT, ERROR };
```

```
Status giveDiscount(long bonusPoints, boolean goldCustomer)
```

# Return Value

- Enumerated values are best treated in the same way as Boolean values, with multiple separate ranges and one value in each range
- Java does define an ordering for enumerated values – ordinal()
- But the different enumerated values often reflect different types of processing, so most effective to treat each value as a separate range
- Value line:


- The return value has three natural ranges, each with one value
- Alternative typed syntax: [FULLPRICE][DISCOUNT][ERROR]

# Document Your Analysis

- We are going to use tables to document all our work
- They are the most concise and precise way to represent data
- Develop up a **guidebook** for yourself of all the tables we use

# Document Your Analysis

- Natural Ranges for giveDiscount()

| Parameter | Natural Range |
|---|---|

# Document Your Analysis

- Natural Ranges for giveDiscount()

| Parameter | Natural Range |
| --- | --- |
| | |

# Document Your Analysis

- Natural Ranges for giveDiscount()

| Parameter | Natural Range |
| --- | --- |
| bonusPoints | |

# Document Your Analysis

- Natural Ranges for giveDiscount()

| **Parameter** | **Natural Range** |
|---|---|
| bonusPoints | Long.MIN_VALUE..Long.MAX_VALUE |

# Document Your Analysis

- Natural Ranges for giveDiscount()

| **Parameter** | **Natural Range** |
|---|---|
| bonusPoints | Long.MIN_VALUE..Long.MAX_VALUE |
| | |

# Document Your Analysis

- Natural Ranges for giveDiscount()

| **Parameter** | **Natural Range** |
|---|---|
| bonusPoints | Long.MIN_VALUE..Long.MAX_VALUE |
| goldCustomer | |

# Document Your Analysis

- Natural Ranges for giveDiscount()

| **Parameter** | **Natural Range** |
|---|---|
| bonusPoints | Long.MIN_VALUE..Long.MAX_VALUE |
| goldCustomer | true |

# Document Your Analysis

- Natural Ranges for giveDiscount()

| **Parameter** | **Natural Range** |
|---|---|
| bonusPoints | Long.MIN_VALUE..Long.MAX_VALUE |
| goldCustomer | true<br>false |

# Document Your Analysis

- Natural Ranges for giveDiscount()

| **Parameter** | **Natural Range** |
|---|---|
| bonusPoints | Long.MIN_VALUE..Long.MAX_VALUE |
| goldCustomer | true<br>false |
| | |

# Document Your Analysis

- Natural Ranges for giveDiscount()

| **Parameter** | **Natural Range** |
|---|---|
| bonusPoints | Long.MIN_VALUE..Long.MAX_VALUE |
| goldCustomer | true<br>false |
| Return Value | |

# Document Your Analysis

- Natural Ranges for giveDiscount()

| **Parameter** | **Natural Range** |
|---|---|
| bonusPoints | Long.MIN_VALUE..Long.MAX_VALUE |
| goldCustomer | true<br>false |
| Return Value | FULLPRICE |

# Document Your Analysis

- Natural Ranges for giveDiscount()

| **Parameter** | **Natural Range** |
|---|---|
| bonusPoints | Long.MIN_VALUE..Long.MAX_VALUE |
| goldCustomer | true<br>false |
| Return Value | FULLPRICE<br>DISCOUNT |

# Document Your Analysis

- Natural Ranges for giveDiscount()

| **Parameter** | **Natural Range** |
|---|---|
| bonusPoints | Long.MIN_VALUE..Long.MAX_VALUE |
| goldCustomer | true<br>false |
| Return Value | FULLPRICE<br>DISCOUNT<br>ERROR |

# Document Your Analysis

- Natural Ranges for giveDiscount()

| **Parameter** | **Natural Range** |
|---|---|
| bonusPoints | Long.MIN_VALUE..Long.MAX_VALUE |
| goldCustomer | true<br>false |
| Return Value | FULLPRICE<br>DISCOUNT<br>ERROR |

# Specification-Based Ranges

- The natural ranges were developed based on the **data types**
- We now look at the ranges of values of interest in testing, based on the software **specification**

```
Status giveDiscount(long bonusPoints, boolean goldCustomer)
```

# Specification-Based Ranges

- The specification-based ranges are built up by “walking” the value lines from left to right

Status giveDiscount(long bonusPoints, boolean goldCustomer)

# Specification-Based Ranges

- The specification-based ranges are built up by “walking” the value lines from left to right
- And identifying the values at which a change of processing may take place

Status giveDiscount(long bonusPoints, boolean goldCustomer)

# Specification-Based Ranges

- The specification-based ranges are built up by “walking” the value lines from left to right
- And identifying the values at which a change of processing may take place
- Note: value lines need not be drawn to scale

# Specification-Based Ranges

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

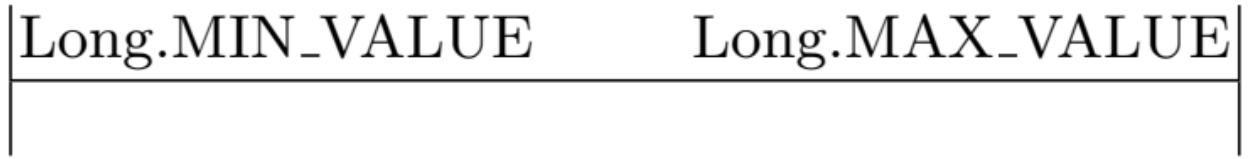
- Start with the natural range
- For bonusPoints, the first value at the left is Long.MIN_VALUE

| Long.MIN_VALUE | Long.MAX_VALUE |
|---|---|
| | |

# Specification-Based Ranges

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

- Start with the natural range
- For bonusPoints, the first value at the left is Long.MIN_VALUE

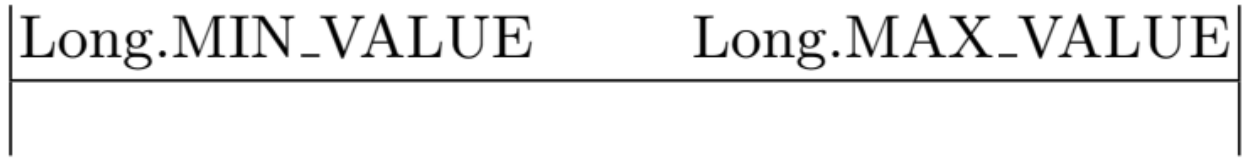



- Walking along the value line, according to the specification, all the subsequent values up to and including the value 0 are treated equivalently: they are all errors

# Specification-Based Ranges

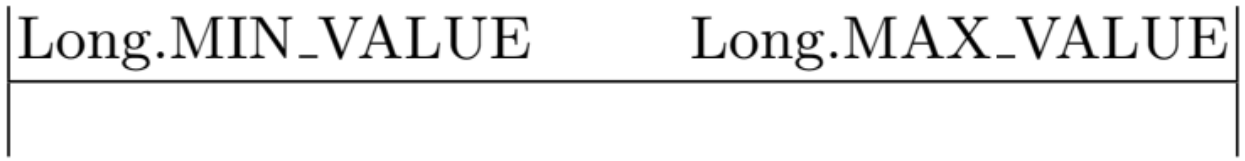
FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

- Start with the natural range
- For bonusPoints, the first value at the left is Long.MIN_VALUE

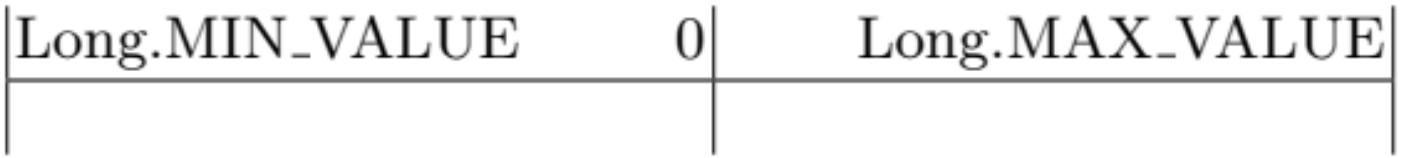


# Specification-Based Ranges

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

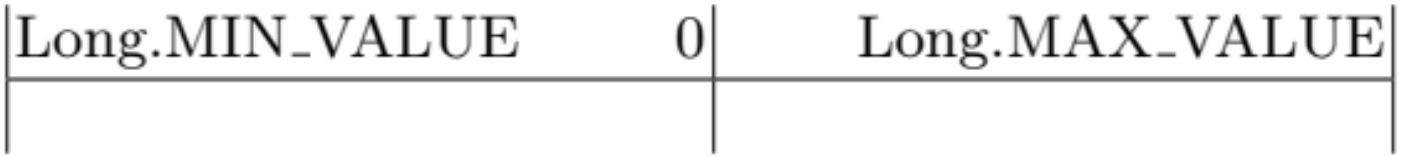



- Walking along the value line, according to the specification, all the subsequent values up to and including the value 0 are treated equivalently: they are all errors



# Continuing with bonusPoints

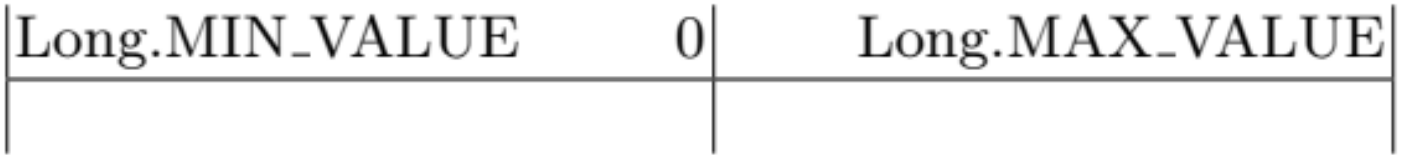
FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)



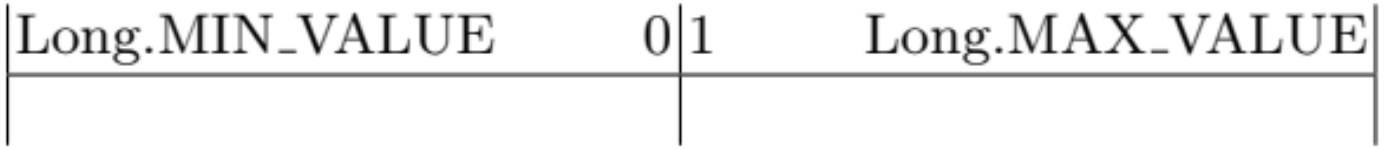

- The next (int) value after 0 is 1

# Continuing with bonusPoints

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

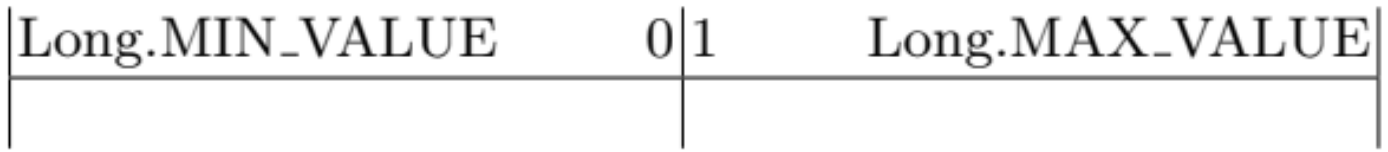



- The next value after 0 is 1
- This is entered as shown:



# Continuing with bonusPoints

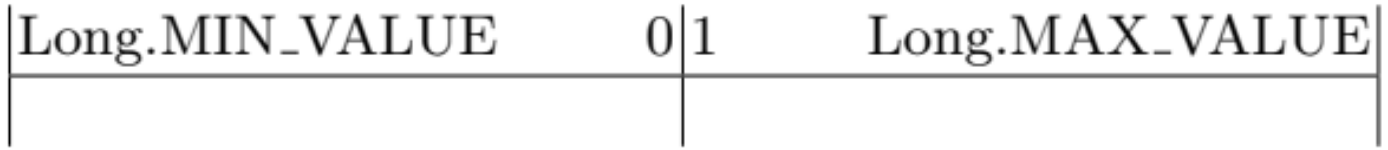
FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)



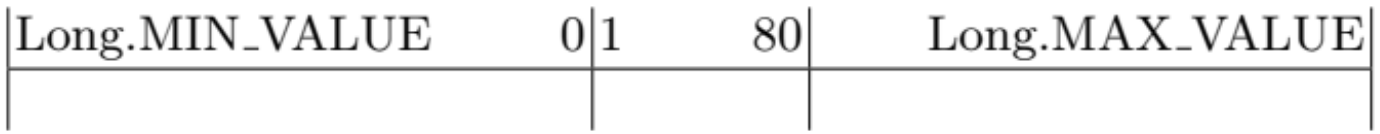

- Walking along the value line from 1, all the subsequent values up to and including the value 80 are treated equivalently

# Continuing with bonusPoints

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)



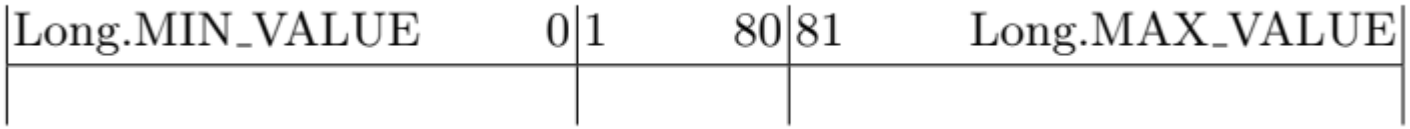

- Walking along the value line from 1, all the subsequent values up to and including the value 80 are treated equivalently
- The value 81 may be treated differently
- The processing is not equivalent for both 80 and 81



# Continuing with bonusPoints

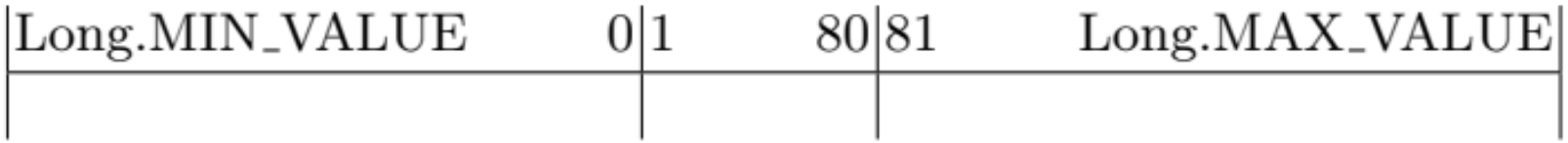
FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)


- The next value after 80 is 81

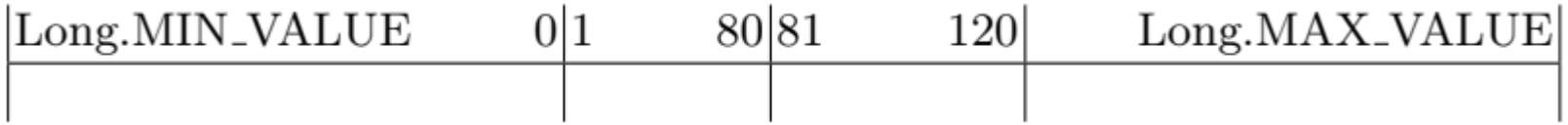


# Continuing with bonusPoints

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)



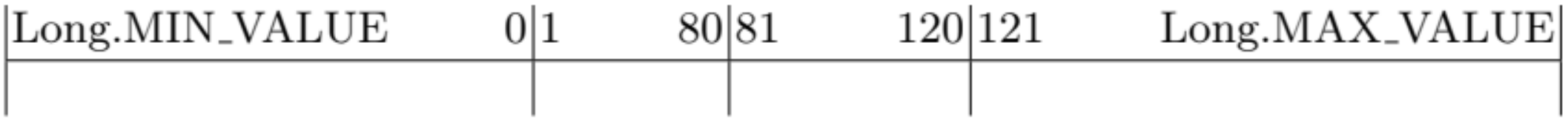

- Walking along the value line from 81, all the subsequent values up to and including the value 120 are treated equivalently



# Continuing with bonusPoints

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

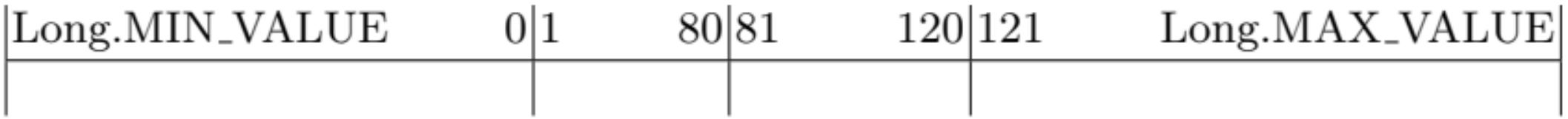
- The next value after 120 is 121



# Continuing with bonusPoints

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

- The next value after 120 is 121




- All values from 121 to Long.MAX_VALUE are treated equivalently
- This is the final specification-based range for bonusPoints

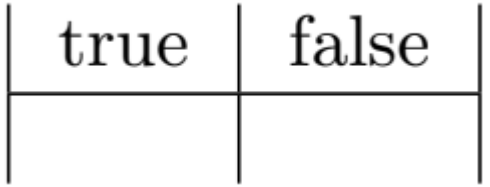
# goldCustomer

- goldCustomer is a Boolean
- The natural ranges were identified:

| true | false |
| --- | --- |
| | |

# goldCustomer

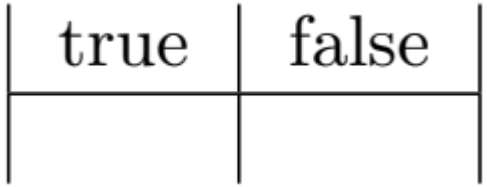
- goldCustomer is a Boolean
- The natural ranges were identified:



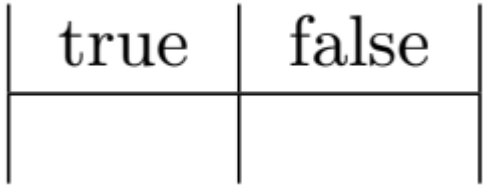

- There are two equivalence partitions, matching the natural ranges, as the processing in each may be different

# goldCustomer

- goldCustomer is a Boolean
- The natural ranges were identified:




- There are two equivalence partitions, matching the natural ranges, as the processing in each may be different
- The specification-based ranges are the same as the natural range:



```
enum Status { FULLPRICE, DISCOUNT, ERROR };
```

# Return Value

- The specification states that each of the natural ranges is the result of a different type of processing
- As with goldCustomer, this produces an identical value line for the specification-based ranges as for the natural ranges

```
enum Status { FULLPRICE, DISCOUNT, ERROR };
```

# Return Value

- The specification states that each of the natural ranges is the result of a different type of processing
- As with goldCustomer, this produces an identical value line for the specification-based ranges as for the natural ranges
- Specification-based ranges for the return value:

| FULLPRICE | DISCOUNT | ERROR |
|---|---|---|
| | | |

# Document your Analysis

- Each specification-based range on the value lines is an equivalence partition
- Input Equivalence Partitions for giveDiscount()
  - Including valid (or normal) and error partitions

| Parameter | Equivalence Partition |
|---|---|

# Document your Analysis

- Each specification-based range on the value lines is an equivalence partition
- Input Equivalence Partitions for giveDiscount()
  - Including valid (or normal) and error partitions

| Parameter | Equivalence Partition |
| --- | --- |
| bonusPoints | |

# Document your Analysis

- Each specification-based range on the value lines is an equivalence partition
- Input Equivalence Partitions for giveDiscount()
  - Including valid (or normal) and error partitions – use (*) to indicate error partitions

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0 |

# Document your Analysis

- Each specification-based range on the value lines is an equivalence partition
- Input Equivalence Partitions for giveDiscount()
  - Including valid (or normal) and error partitions

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80 |

# Document your Analysis

- Each specification-based range on the value lines is an equivalence partition
- Input Equivalence Partitions for giveDiscount()
  - Including valid (or normal) and error partitions

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120 |

# Document your Analysis

- Each specification-based range on the value lines is an equivalence partition
- Input Equivalence Partitions for giveDiscount()
  - Including valid (or normal) and error partitions

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |

# Document your Analysis

- Each specification-based range on the value lines is an equivalence partition
- Input Equivalence Partitions for giveDiscount()
  - Including valid (or normal) and error partitions

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |
| | |

# Document your Analysis

- Each specification-based range on the value lines is an equivalence partition
- Input Equivalence Partitions for giveDiscount()
  - Including valid (or normal) and error partitions

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |
| goldCustomer | |

# Document your Analysis

- Each specification-based range on the value lines is an equivalence partition
- Input Equivalence Partitions for giveDiscount()
  - Including valid (or normal) and error partitions

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |
| goldCustomer | true |

# Document your Analysis

- Each specification-based range on the value lines is an equivalence partition
- Input Equivalence Partitions for giveDiscount()
  - Including valid (or normal) and error partitions

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |
| goldCustomer | true<br>false |

# Document your Analysis

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |
| goldCustomer | true<br>false |

- **Input** equivalence partitions associated with error processing are indicated with an asterisk (*)
- In this example it is straightforward to identify these
- Sometimes, it may be hard to identify these (poorly written specifications)

# Document your Analysis

- Output Equivalence Partitions for giveDiscount()

# Document your Analysis

- Output Equivalence Partitions for giveDiscount()

| Parameter | Equivalence Partition |
| --- | --- |
| | |

# Document your Analysis

- Output Equivalence Partitions for giveDiscount()

| Parameter | Equivalence Partition |
| --- | --- |
| Return Value | |

# Document your Analysis

- Output Equivalence Partitions for giveDiscount()

| **Parameter** | **Equivalence Partition** |
|---|---|
| Return Value | FULLPRICE |

# Document your Analysis

- Output Equivalence Partitions for giveDiscount()

| **Parameter** | **Equivalence Partition** |
|---|---|
| Return Value | FULLPRICE<br>DISCOUNT |

# Document your Analysis

- Output Equivalence Partitions for giveDiscount()

| **Parameter** | **Equivalence Partition** |
|---|---|
| Return Value | FULLPRICE<br>DISCOUNT<br>ERROR |

# In Practice

- Usually this analysis is probably not written down
- Experienced developer identify the equivalence partitions directly
- But, when learning how to test, it is recommended that you fully document the results of the analysis as shown above
- This helps to ensure that you are following the technique correctly
- And helps me to understand and correct any mistakes you make

# BREAK

# 2. Test Coverage Items (TCI)

- We generate test coverage items from the equivalence partitions
- A test coverage item is something to be tested for
- Each equivalence partition is a test coverage item

# 2. Test Coverage Items (TCI)

- We generate test coverage items from the equivalence partitions
- A test coverage item is something to be tested for
- Each equivalence partition is a test coverage item
- Reminder: the EPs

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |
| goldCustomer | true<br>false |

| Parameter | Equivalence Partition |
|---|---|
| Return Value | FULLPRICE<br>DISCOUNT<br>ERROR |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | | | |

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |
| goldCustomer | true<br>false |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | | |

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |
| goldCustomer | true<br>false |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | |

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |
| goldCustomer | true<br>false |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |
| goldCustomer | true<br>false |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | | |

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |
| goldCustomer | true<br>false |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | 1..80 | |

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |
| goldCustomer | true<br>false |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | 1..80 | |
| EP3 | | | |

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |
| goldCustomer | true<br>false |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |
| goldCustomer | true<br>false |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | | |

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |
| goldCustomer | true<br>false |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |
| goldCustomer | true<br>false |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | | | |

| Parameter | Equivalence Partition |
|---|---|
| goldCustomer | true<br>false |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | | |

| Parameter | Equivalence Partition |
|---|---|
| goldCustomer | true |
| | false |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |

| Parameter | Equivalence Partition |
|---|---|
| goldCustomer | true |
| | false |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |

| Parameter | Equivalence Partition |
|---|---|
| goldCustomer | true<br>false |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | | | |

| Parameter | Equivalence Partition |
|---|---|
| Return Value | FULLPRICE |
| | DISCOUNT |
| | ERROR |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | | |

| Parameter | Equivalence Partition |
|---|---|
| Return Value | FULLPRICE |
| | DISCOUNT |
| | ERROR |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |

| Parameter | Equivalence Partition |
|---|---|
| Return Value | FULLPRICE |
| | DISCOUNT |
| | ERROR |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | | |

| Parameter | Equivalence Partition |
|---|---|
| Return Value | FULLPRICE |
| | DISCOUNT |
| | ERROR |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | DISCOUNT | |

| Parameter | Equivalence Partition |
|---|---|
| Return Value | FULLPRICE |
| | DISCOUNT |
| | ERROR |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | DISCOUNT | |
| EP9 | | | |

| Parameter | Equivalence Partition |
|---|---|
| Return Value | FULLPRICE |
| | DISCOUNT |
| | ERROR |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | DISCOUNT | |
| EP9 | | ERROR | |

| Parameter | Equivalence Partition |
|---|---|
| Return Value | FULLPRICE |
| | DISCOUNT |
| | ERROR |

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | DISCOUNT | |
| EP9 | | ERROR | |

- **Unique identifiers** (EP1, EP2, etc) for each test coverage item:
  - Keep track of which test cases cover each test coverage item
  - Make sure nothing is missed
- Note: these TCIs are specific to EP testing

# Test Coverage Items for giveDiscount()

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | DISCOUNT | |
| EP9 | | ERROR | |

- An asterisk (*) is used after the identifier to indicate input test coverage items for **errors** – this is important as only one test coverage item representing an input error can be used for a particular test case
- The equivalence partitions for each parameter should be **grouped** together and presented in order as we can see in the table
- A blank **Test Case column** is also included on the right-hand side of the table (to be completed later)

# Recap: Analysis -> TCI

| Parameter | Equivalence Partition |
|---|---|
| bonusPoints | (*) Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE |
| goldCustomer | true<br>false |

| Parameter | Equivalence Partition |
|---|---|
| Return Value | FULLPRICE<br>DISCOUNT<br>ERROR |

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | DISCOUNT | |
| EP9 | | ERROR | |

# 3. Test Cases

- These specify the tests to be run
- And the actual data that will be used for testing
  - Inputs
  - Expected Outputs (return value in this example)

# 3. Test Cases

- These specify the tests to be run
- And the actual data that will be used for testing
  - Inputs
  - Expected Outputs (return value in this example)

- Two stages:
  1. Select equivalence values for each input and output equivalence partitions
  2. Build the test case table

# Test Cases/Select Equivalence Values

- Select “equivalence” values within the equivalence partitions:
  - For short ranges, a central value is selected
  - For longer ranges, a convenient value is selected, but not the start or end value
  - It is important that boundary values are not selected: picking a value from the center is more likely to expose faults due to incorrect processing of the entire partition which is the goal of equivalence partition testing
  - This makes debugging of any faults easier, as the causes for failures are easier to identify if they are a good match for the fault model of the test technique
  - The boundary values are not “typical” values
  - For non-continuous ranges, such as Boolean or enumerated values, each equivalence partition only has a single value.

# Equivalence Values

| Parameter | Equivalence Partition | Equivalence Value |
|---|---|---|
| bonusPoints | Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE | |
| goldCustomer | true<br>false | |
| Return Value | FULLPRICE<br>DISCOUNT<br>ERROR | |

- Note: in the lab I will give you stricter rules for picking the values!

# Equivalence Values

| Parameter | Equivalence Partition | Equivalence Value |
|---|---|---|
| bonusPoints | Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE | -100 |
| goldCustomer | true<br>false | |
| Return Value | FULLPRICE<br>DISCOUNT<br>ERROR | |

- Note: in the lab I will give you stricter rules for picking the values!

# Equivalence Values

| Parameter | Equivalence Partition | Equivalence Value |
|---|---|---|
| bonusPoints | Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE | -100<br>40 |
| goldCustomer | true<br>false | |
| Return Value | FULLPRICE<br>DISCOUNT<br>ERROR | |

- When possible, select the central value

# Equivalence Values

| Parameter | Equivalence Partition | Equivalence Value |
|---|---|---|
| bonusPoints | Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE | -100<br>40<br>100 |
| goldCustomer | true<br>false | |
| Return Value | FULLPRICE<br>DISCOUNT<br>ERROR | |

- When possible, select the central value

# Equivalence Values

| Parameter | Equivalence Partition | Equivalence Value |
|---|---|---|
| bonusPoints | Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE | -100<br>40<br>100<br>200 |
| goldCustomer | true<br>false | |
| Return Value | FULLPRICE<br>DISCOUNT<br>ERROR | |

- Note: in the lab I will give you stricter rules for picking the values!

# Equivalence Values

| Parameter | Equivalence Partition | Equivalence Value |
|---|---|---|
| bonusPoints | Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE | -100<br>40<br>100<br>200 |
| goldCustomer | true<br>false | true<br>false |
| Return Value | FULLPRICE<br>DISCOUNT<br>ERROR | |

- Booleans: equivalence values are specified by the EP values

# Equivalence Values

| Parameter | Equivalence Partition | Equivalence Value |
|---|---|---|
| bonusPoints | Long.MIN_VALUE..0<br>1..80<br>81..120<br>121..Long.MAX_VALUE | -100<br>40<br>100<br>200 |
| goldCustomer | true<br>false | true<br>false |
| Return Value | FULLPRICE<br>DISCOUNT<br>ERROR | FULLPRICE<br>DISCOUNT<br>ERROR |

- Enums: equivalence values are specified by the EP values

# Test Cases/Creating the Test Case Table

- This is  probably the trickiest part of test design

# Test Cases/Creating the Test Case Table

- Create an empty TC table with columns as shown

| **ID** | **TCI Covered** | **Inputs** | | **Exp. Results** |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| | | | | |

# Test Cases/Creating the Test Case Table

- Create the table with columns as shown

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |

- Work down the TCI table, adding new Test Cases for uncovered TCIs

# Test Cases/Creating the Test Case Table

- Create the table with columns as shown

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |

- Work down the TCI table, adding new Test Cases for uncovered TCIs
- Do the errors marked with * last

# Reminder: Test Coverage Items

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | DISCOUNT | |
| EP9 | | ERROR | |

# Selecting First Uncovered non-error TCIs

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | DISCOUNT | |
| EP9 | | ERROR | |

# Selecting First Uncovered TCIs

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | DISCOUNT | |
| EP9 | | ERROR | |

# Selecting First Uncovered TCIs

| TCI | Parameter | Fe…on | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | | To be completed later |
| EP2 | | | |
| EP3 | | | |
| EP4 | | | |
| EP5 | goldCustomer | | |
| EP6 | | | |
| EP7 | Return Value | …E | |
| EP8 | | DISCOUNT | |
| EP9 | | ERROR | |

NOTE:

YOU CAN'T PICK the Return Values

It will be determined by the specification

# Test Cases/Starting the Test Case Table

- Start with the normal (non-error) EPs in order
- Give each test case an unique identifier: e.g. T1.1

| **ID** | **TCI Covered** | **Inputs** | | **Exp. Results** |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | | | | |

# Test Cases/Starting the Test Case Table

- Start with the normal (non-error) EPs in order
- Give each test case an unique identifier: e.g. T1.1
- Select the next uncovered TCIs
- Indicate the TCI's: abbreviate with EP2,5 to save space

| **ID** | **TCI Covered** | **Inputs** | | **Exp. Results** |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5 | | | |

# Test Cases/Starting the Test Case Table

- Start with the normal (non-error) EPs in order
- Give each test case an unique identifier: e.g. T1.1
- Select value for the first (uncovered) EP for each parameter
  - EP2 for bonusPoints, EP5 for goldCustomer

| **ID** | **TCI Covered** | **Inputs** | | **Exp. Results** |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5 | 40 | | |

# Test Cases/Starting the Test Case Table

- Start with the normal (non-error) EPs in order
- Give each test case an unique identifier: e.g. T1.1
- Select value for the first (uncovered) EP for each parameter
  - EP2 for bonusPoints, EP5 for goldCustomer

| **ID** | **TCI Covered** | **Inputs** | | **Exp. Results** |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5 | 40 | true | |

# Expected Results for Test Case T1.1

- Now use the specification to determine the correct output

**return value:**

FULLPRICE if bonusPoints$\leq$120 and not a goldCustomer

FULLPRICE if bonusPoints$\leq$80 and a goldCustomer

# Expected Results for Test Case T1.1

- Now use the specification to determine the correct output

**return value:**

FULLPRICE if bonusPoints$\leq$120 and not a goldCustomer

FULLPRICE if bonusPoints$\leq$80 and a goldCustomer

- Exp. Results:
  - If bonusPoints is 40 and goldCustomer is true
  - The expected results are the return value FULLPRICE

# Expected Results for Test Case T1.1

- Now use the specification to determine the correct output

**return value:**

FULLPRICE if bonusPoints≤120 and not a goldCustomer

FULLPRICE if bonusPoints≤80 and a goldCustomer

- Exp. Results:
  - If bonusPoints is 40 and goldCustomer is true
  - The expected results are the return value FULLPRICE

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |

# Selecting Next Uncovered TCIs

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | To be completed later |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | DISCOUNT | |
| EP9 | | ERROR | |

# Normal Test Cases for giveDiscount()

- Complete the table, using uncovered TCI's in order

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1<br>T1.2 | EP2,5,7<br>EP3,6 | 40 | true | FULLPRICE |

- For T1.2, select EP3, EP6

# Normal Test Cases for giveDiscount()

- Complete the table, using uncovered TCI's in order

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6 | 100 | | |

- For T1.2, select EP3, EP6
- Add the selected data values

# Normal Test Cases for giveDiscount()

- Complete the table, using uncovered TCI's in order

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6 | 100 | false | |

- For T1.2, select EP3, EP6
- Add the selected data values

# Normal Test Cases for giveDiscount()

- Complete the table, using uncovered TCI's in order

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |

- For T1.2, select EP3, EP6
- Expected output is FULLPRICE EP7
- Indicate duplicate coverage with []'s

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

# Normal Test Cases for giveDiscount()

- Complete the table, using uncovered TCI's in order

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | | | |

- For T1.3, select EP4
  - Have to use EP5 or EP6 (be systematic: keep re-using the last one used)

# Normal Test Cases for giveDiscount()

- Complete the table, using uncovered TCI's in order

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | | |

- For T1.3, select EP4
  - Have to use EP5 or EP6 (be systematic: keep re-using the last one used)
- Data values from table

# Normal Test Cases for giveDiscount()

- Complete the table, using uncovered TCI's in order

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | |

- For T1.3, select EP4
  - Have to use EP5 or EP6 (be systematic: keep re-using the last one used)

# Normal Test Cases for giveDiscount()

- Complete the table, using uncovered TCI's in order

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |

- For T1.3, select EP4
  - Have to use EP5 or EP6
    (be systematic: keep re-using the last one used)
  - This produces DISCOUNT or EP8

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

# Error Cases

- Each input error test coverage item must have a unique test case
- One input error test coverage item covered by any one test case
- This is due to error hiding

| **ID** | **TCI Covered** | **Inputs** | | **Exp. Results** |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | | | | |

# Error Cases

- First uncovered error TCI is EP1*

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1* | | | |

# Error Cases

- First uncovered error TCI is EP1*
- Use the selected data value

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1* | -100 | | |

# Error Cases

- Each input error test coverage item must have a unique test case
- Must pick a value for goldCustomer: be systematic and (if possible) use the last used value – here it is false

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1* | -100 | false | |

# Error Cases

FULLPRICE if bonusPoints≤120 and not a goldCustomer
FULLPRICE if bonusPoints≤80 and a goldCustomer
DISCOUNT if bonusPoints>120
DISCOUNT if bonusPoints>80 and a goldCustomer
ERROR if any inputs are invalid (bonusPoints<1)

- And the specification states this will produce ERROR

| **ID** | **TCI Covered** | **Inputs** | | **Exp. Results** |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

# EP Test Cases for giveDiscount()

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

# EP Test Cases for giveDiscount()

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

- Each input error test coverage item is tested separately
- The test cases are **not necessarily specific** to a particular technique
  - Easier to reuse test cases if the technique abbreviation is not included in test case ID
- Minimising the number of test cases can require multiple iterations
- Target – the maximum number of partitions of any input or output:
  - In this example it is 4 (for the bonusPoints parameter)
  - Not always achievable
- Combinations of input values are not considered
  - In this example, no test for bonusPoints equal to 40 with goldCustomer both true and false

# 4. Test Design Verification

- Two parts:
  1. Complete the test coverage item table
  2. Review your work

# Completed Test Coverage Item Table

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

- Complete the TCI Test Case column using the Test Cases table
- For example, T1.1 covers EP2, EP5, and EP7 (ignore duplicates)

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | DISCOUNT | |
| EP9 | | ERROR | |

# Completed Test Coverage Item Table

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

- Complete the TCI Test Case column using the Test Cases table
- For example, T1.1 covers EP2, EP5, and EP7 (ignore duplicates)

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| EP2 | | 1..80 | |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | DISCOUNT | |
| EP9 | | ERROR | |

# Completed Test Coverage Item Table

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

- Complete the TCI Test Case column using the Test Cases table
- For example, T1.1 covers EP2, EP5, and EP7 (ignore duplicates)

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| EP2 | | 1..80 | T1.1 |
| EP3 | | 81..120 | |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | DISCOUNT | |
| EP9 | | ERROR | |

# Completed Test Coverage Item Table

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

- Complete the TCI Test Case column using the Test Cases table
- For example, T1.1 covers EP2, EP5, and EP7 (ignore duplicates)

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| EP2 | | 1..80 | T1.1 |
| EP3 | | 81..120 | T1.2 |
| EP4 | | 121..Long.MAX_VALUE | |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | DISCOUNT | |
| EP9 | | ERROR | |

# Completed Test Coverage Item Table

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

- Complete the TCI Test Case column using the Test Cases table
- For example, T1.1 covers EP2, EP5, and EP7 (ignore duplicates)

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| EP2 | | 1..80 | T1.1 |
| EP3 | | 81..120 | T1.2 |
| EP4 | | 121..Long.MAX_VALUE | T1.3 |
| EP5 | goldCustomer | true | |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | DISCOUNT | |
| EP9 | | ERROR | |

# Completed Test Coverage Item Table

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

- Complete the TCI Test Case column using the Test Cases table
- For example, T1.1 covers EP2, EP5, and EP7 (ignore duplicates)

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| EP2 | | 1..80 | T1.1 |
| EP3 | | 81..120 | T1.2 |
| EP4 | | 121..Long.MAX_VALUE | T1.3 |
| EP5 | goldCustomer | true | T1.1 |
| EP6 | | false | |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | DISCOUNT | |
| EP9 | | ERROR | |

# Completed Test Coverage Item Table

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

- Complete the TCI Test Case column using the Test Cases table
- For example, T1.1 covers EP2, EP5, and EP7 (ignore duplicates)

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| EP2 | | 1..80 | T1.1 |
| EP3 | | 81..120 | T1.2 |
| EP4 | | 121..Long.MAX_VALUE | T1.3 |
| EP5 | goldCustomer | true | T1.1 |
| EP6 | | false | T1.2 |
| EP7 | Return Value | FULLPRICE | |
| EP8 | | DISCOUNT | |
| EP9 | | ERROR | |

# Completed Test Coverage Item Table

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

- Complete the TCI Test Case column using the Test Cases table
- For example, T1.1 covers EP2, EP5, and EP7 (ignore duplicates)

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| EP2 | | 1..80 | T1.1 |
| EP3 | | 81..120 | T1.2 |
| EP4 | | 121..Long.MAX_VALUE | T1.3 |
| EP5 | goldCustomer | true | T1.1 |
| EP6 | | false | T1.2 |
| EP7 | Return Value | FULLPRICE | T1.1 |
| EP8 | | DISCOUNT | |
| EP9 | | ERROR | |

# Completed Test Coverage Item Table

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

- Complete the TCI Test Case column using the Test Cases table
- For example, T1.1 covers EP2, EP5, and EP7 (ignore duplicates)

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| EP2 | | 1..80 | T1.1 |
| EP3 | | 81..120 | T1.2 |
| EP4 | | 121..Long.MAX_VALUE | T1.3 |
| EP5 | goldCustomer | true | T1.1 |
| EP6 | | false | T1.2 |
| EP7 | Return Value | FULLPRICE | T1.1 |
| EP8 | | DISCOUNT | T1.3 |
| EP9 | | ERROR | |

# Completed Test Coverage Item Table

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

- Complete the TCI Test Case column using the Test Cases table
- For example, T1.1 covers EP2, EP5, and EP7 (ignore duplicates)

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| EP2 | | 1..80 | T1.1 |
| EP3 | | 81..120 | T1.2 |
| EP4 | | 121..Long.MAX_VALUE | T1.3 |
| EP5 | goldCustomer | true | T1.1 |
| EP6 | | false | T1.2 |
| EP7 | Return Value | FULLPRICE | T1.1 |
| EP8 | | DISCOUNT | T1.3 |
| EP9 | | ERROR | T1.4 |

# Reviewing Your Work

1. Every test coverage item is covered by at least one test case with suitable test data:
   - This confirms that the test cases are complete
2. Every new test case covers at least one additional test coverage item:
   - this confirms that there are no unnecessary test cases
   - Ideally, each test case should cover as many new test coverage items as possible (up to three in this example: two input TCIs, and one output TCI)

# Reviewing the Test Coverage Items

- Every TCI is covered by at least one test

| TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|
| EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| EP2 | | 1..80 | T1.1 |
| EP3 | | 81..120 | T1.2 |
| EP4 | | 121..Long.MAX_VALUE | T1.3 |
| EP5 | goldCustomer | true | T1.1 |
| EP6 | | false | T1.2 |
| EP7 | Return Value | FULLPRICE | T1.1 |
| EP8 | | DISCOUNT | T1.3 |
| EP9 | | ERROR | T1.4 |

# Reviewing the Test Coverage Items

- Every TCI is covered by at least one test

| | TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|---|
| ✓ | EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| | EP2 | | 1..80 | T1.1 |
| | EP3 | | 81..120 | T1.2 |
| | EP4 | | 121..Long.MAX_VALUE | T1.3 |
| | EP5 | goldCustomer | true | T1.1 |
| | EP6 | | false | T1.2 |
| | EP7 | Return Value | FULLPRICE | T1.1 |
| | EP8 | | DISCOUNT | T1.3 |
| | EP9 | | ERROR | T1.4 |

# Reviewing the Test Coverage Items

- Every TCI is covered by at least one test

| | TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|---|
| ✓ | EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| ✓ | EP2 | | 1..80 | T1.1 |
| | EP3 | | 81..120 | T1.2 |
| | EP4 | | 121..Long.MAX_VALUE | T1.3 |
| | EP5 | goldCustomer | true | T1.1 |
| | EP6 | | false | T1.2 |
| | EP7 | Return Value | FULLPRICE | T1.1 |
| | EP8 | | DISCOUNT | T1.3 |
| | EP9 | | ERROR | T1.4 |

# Reviewing the Test Coverage Items

- Every TCI is covered by at least one test

| | TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|---|
| ✓ | EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| ✓ | EP2 | | 1..80 | T1.1 |
| ✓ | EP3 | | 81..120 | T1.2 |
| | EP4 | | 121..Long.MAX_VALUE | T1.3 |
| | EP5 | goldCustomer | true | T1.1 |
| | EP6 | | false | T1.2 |
| | EP7 | Return Value | FULLPRICE | T1.1 |
| | EP8 | | DISCOUNT | T1.3 |
| | EP9 | | ERROR | T1.4 |

# Reviewing the Test Coverage Items

- Every TCI is covered by at least one test

| | TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|---|
| ✓ | EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| ✓ | EP2 | | 1..80 | T1.1 |
| ✓ | EP3 | | 81..120 | T1.2 |
| ✓ | EP4 | | 121..Long.MAX_VALUE | T1.3 |
| | EP5 | goldCustomer | true | T1.1 |
| | EP6 | | false | T1.2 |
| | EP7 | Return Value | FULLPRICE | T1.1 |
| | EP8 | | DISCOUNT | T1.3 |
| | EP9 | | ERROR | T1.4 |

# Reviewing the Test Coverage Items

- Every TCI is covered by at least one test

| | TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|---|
| ✓ | EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| ✓ | EP2 | | 1..80 | T1.1 |
| ✓ | EP3 | | 81..120 | T1.2 |
| ✓ | EP4 | | 121..Long.MAX_VALUE | T1.3 |
| ✓ | EP5 | goldCustomer | true | T1.1 |
| | EP6 | | false | T1.2 |
| | EP7 | Return Value | FULLPRICE | T1.1 |
| | EP8 | | DISCOUNT | T1.3 |
| | EP9 | | ERROR | T1.4 |

# Reviewing the Test Coverage Items

- Every TCI is covered by at least one test

| | TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|---|
| ✓ | EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| ✓ | EP2 | | 1..80 | T1.1 |
| ✓ | EP3 | | 81..120 | T1.2 |
| ✓ | EP4 | | 121..Long.MAX_VALUE | T1.3 |
| ✓ | EP5 | goldCustomer | true | T1.1 |
| ✓ | EP6 | | false | T1.2 |
| | EP7 | Return Value | FULLPRICE | T1.1 |
| | EP8 | | DISCOUNT | T1.3 |
| | EP9 | | ERROR | T1.4 |

# Reviewing the Test Coverage Items

- Every TCI is covered by at least one test

| | TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|---|
| ✓ | EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| ✓ | EP2 | | 1..80 | T1.1 |
| ✓ | EP3 | | 81..120 | T1.2 |
| ✓ | EP4 | | 121..Long.MAX_VALUE | T1.3 |
| ✓ | EP5 | goldCustomer | true | T1.1 |
| ✓ | EP6 | | false | T1.2 |
| ✓ | EP7 | Return Value | FULLPRICE | T1.1 |
| | EP8 | | DISCOUNT | T1.3 |
| | EP9 | | ERROR | T1.4 |

# Reviewing the Test Coverage Items

- Every TCI is covered by at least one test

| | TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|---|
| ✓ | EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| ✓ | EP2 | | 1..80 | T1.1 |
| ✓ | EP3 | | 81..120 | T1.2 |
| ✓ | EP4 | | 121..Long.MAX_VALUE | T1.3 |
| ✓ | EP5 | goldCustomer | true | T1.1 |
| ✓ | EP6 | | false | T1.2 |
| ✓ | EP7 | Return Value | FULLPRICE | T1.1 |
| ✓ | EP8 | | DISCOUNT | T1.3 |
| | EP9 | | ERROR | T1.4 |

# Reviewing the Test Coverage Items

- Every TCI is covered by at least one test

| | TCI | Parameter | Equivalence Partition | Test Case |
|---|---|---|---|---|
| ✓ | EP1* | bonusPoints | Long.MIN_VALUE..0 | T1.4 |
| ✓ | EP2 | | 1..80 | T1.1 |
| ✓ | EP3 | | 81..120 | T1.2 |
| ✓ | EP4 | | 121..Long.MAX_VALUE | T1.3 |
| ✓ | EP5 | goldCustomer | true | T1.1 |
| ✓ | EP6 | | false | T1.2 |
| ✓ | EP7 | Return Value | FULLPRICE | T1.1 |
| ✓ | EP8 | | DISCOUNT | T1.3 |
| ✓ | EP9 | | ERROR | T1.4 |

# Reviewing the Test Cases

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

# Reviewing the Test Cases

| | ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|---|
| | | | bonusPoints | goldCustomer | return value |
| ✓ | T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| | T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| | T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| | T1.4* | EP1*,9 | -100 | false | ERROR |

- T1.1 covers EP2, EP5, and EP7 for the first time

# Reviewing the Test Cases

| | ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|---|
| | | | bonusPoints | goldCustomer | return value |
| ✓ | T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| ✓ | T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| | T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| | T1.4* | EP1*,9 | -100 | false | ERROR |

- T1.1 covers EP2, EP5, and EP7 for the first time
- T1.2 covers EP3 and EP6 for the first time. It also covers EP7 again, but this is unavoidable as that is the result of these inputs

# Reviewing the Test Cases

| | ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|---|
| | | | bonusPoints | goldCustomer | return value |
| ✓ | T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| ✓ | T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| ✓ | T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| | T1.4* | EP1*,9 | -100 | false | ERROR |

- T1.1 covers EP2, EP5, and EP7 for the first time
- T1.2 covers EP3 and EP6 for the first time. It also covers EP7 again, but this is unavoidable as that is the result of these inputs
- T1.3 covers EP4 and EP8 for the first time. EP6: unavoidable

# Reviewing the Test Cases

| | ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|---|
| | | | bonusPoints | goldCustomer | return value |
| ✓ | T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| ✓ | T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| ✓ | T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| ✓ | T1.4* | EP1*,9 | -100 | false | ERROR |

- T1.1 covers EP2, EP5, and EP7 for the first time
- T1.2 covers EP3 and EP6 for the first time. It also covers EP7 again, but this is unavoidable as that is the result of these inputs
- T1.3 covers EP4 and EP8 for the first time. EP6: unavoidable
- T1.4 is an error test case and it covers the single input error test coverage item EP1*. It also covers the output test coverage item EP9, but not EP6 (because of error hiding, EP6 is not covered by this test)

# Test Design Review Results

- The review is complete
- All the TCI’s are covered by at least one test
- And there are no (unnecessary) duplicates of TCI coverage

# BREAK

# 5. Test Implementation/Manual

- For EP testing only, we will demonstrate manual testing

# 5. Test Implementation/Manual

- For EP testing only, we will demonstrate manual testing
- Examine the output of manually running a small example program
  - called "**check**"
  - which uses the method giveDiscount() to do the calculations
  - with this test data:

| **ID** | **TCI Covered** | **Inputs** | | **Exp. Results** |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

# 5. Test Implementation/Manual

- For EP testing only, we will demonstrate manual testing
- Examine the output of running a small example program

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

- Note: we can run "check" in two ways:
  - Using the java command directly
  - Using gradle

# 5. Test Implementation/Manual


- For EP testing only, we will demonstrate manual testing
- Examine the output of running a small example program

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

- Java command:

```
%JAVA_HOME%\bin\java.exe -cp ch01\check\build\classes\java\main example.Check 40 true
```

# 5. Test Implementation/Manual


- For EP testing only, we will demonstrate manual testing
- Examine the output of running a small example program

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

- gradle:

```
@gradlew ch01:check:run --args="40 true"
```

# Note on Benefits of gradle

- Automatically compiles
- Build the **classpath** argument automatically from dependencies
- And automatically downloads any external dependencies

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

# 5. Test Implementation/Manual

- For EP testing only, we will demonstrate manual testing
- The results of manually running the small example program:

```
$ check 40 true
FULLPRICE
$ check 100 false
FULLPRICE
$ check 200 false
DISCOUNT
$ check -100 false
ERROR
```

# Manual Testing

- The output matches the specification (as shown in the test cases table)

```
$ check 40 true
FULLPRICE
$ check 100 false
FULLPRICE
$ check 200 false
DISCOUNT
$ check -100 false
ERROR
```

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

# Manual Testing

- Running each test case manually, typing in the input data, and checking the result values is:
  - Tedious, error-prone, and very slow

# Manual Testing

- Running each test case manually, typing in the input data, and checking the result values is:
  - Tedious, error-prone, and very slow
- Software testing needs to be:
  - Easy, error-free, and fast

# Manual Testing

- Running each test case manually, typing in the input data, and checking the result values is:
  - Tedious, error-prone, and very slow
- Software testing needs to be:
  - Easy, error-free, and fast
- Especially in modern software development:
  - Ongoing changes made as functionality is incrementally added
  - Requiring frequent re-tests of the software

# Manual Testing

- Running each test case manually, typing in the input data, and checking the result values is:
  - Tedious, error-prone, and very slow
- Software testing needs to be:
  - Easy, error-free, and fast
- Especially in modern software development:
  - Ongoing changes made as functionality is incrementally added
  - Requiring frequent re-tests of the software
- **For this reason, software testing is usually automated and we will only show automated testing in future examples**

# Automated Test Implementation

- The test cases are now implemented for automated execution

# Automated Test Implementation

- The test cases are now implemented for automated execution
- TestNG is used as a demonstrative example in this book – there are many other test Frameworks
  - JUnit is more complex

# Automated Test Implementation

- The test cases are now implemented for automated execution
- TestNG is used as a demonstrative example in this book – there are many other test Frameworks
  - JUnit is more complex
- A test framework typically includes a “Test Runner” which runs the tests, and collects the test results

# Automated Test Implementation

- The test cases are now implemented for automated execution
- TestNG is used as a demonstrative example in this book – there are many other test Frameworks
  - JUnit is more complex
- A test framework typically includes a “Test Runner” which runs the tests, and collects the test results
- Because of this, no main() method is required in the test class

# Automated Test Implementation

- The test cases are now implemented for automated execution
- TestNG is used as a demonstrative example in this book – there are many other test Frameworks
  - JUnit is more complex
- A test framework typically includes a “Test Runner” which runs the tests, and collects the test results
- Because of this, no main() method is required in the test class
- Tests need to be identified for the runner: in TestNG, Java Annotation is used, which allows method information to be determined at runtime

# Automated Test Implementation

- Test Case T1.1 has
  - Inputs: bonusPoints=40L and goldCustomer=true
  - expected results: return value==FULLPRICE

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

# OnlineSalesTest.java/T1.1

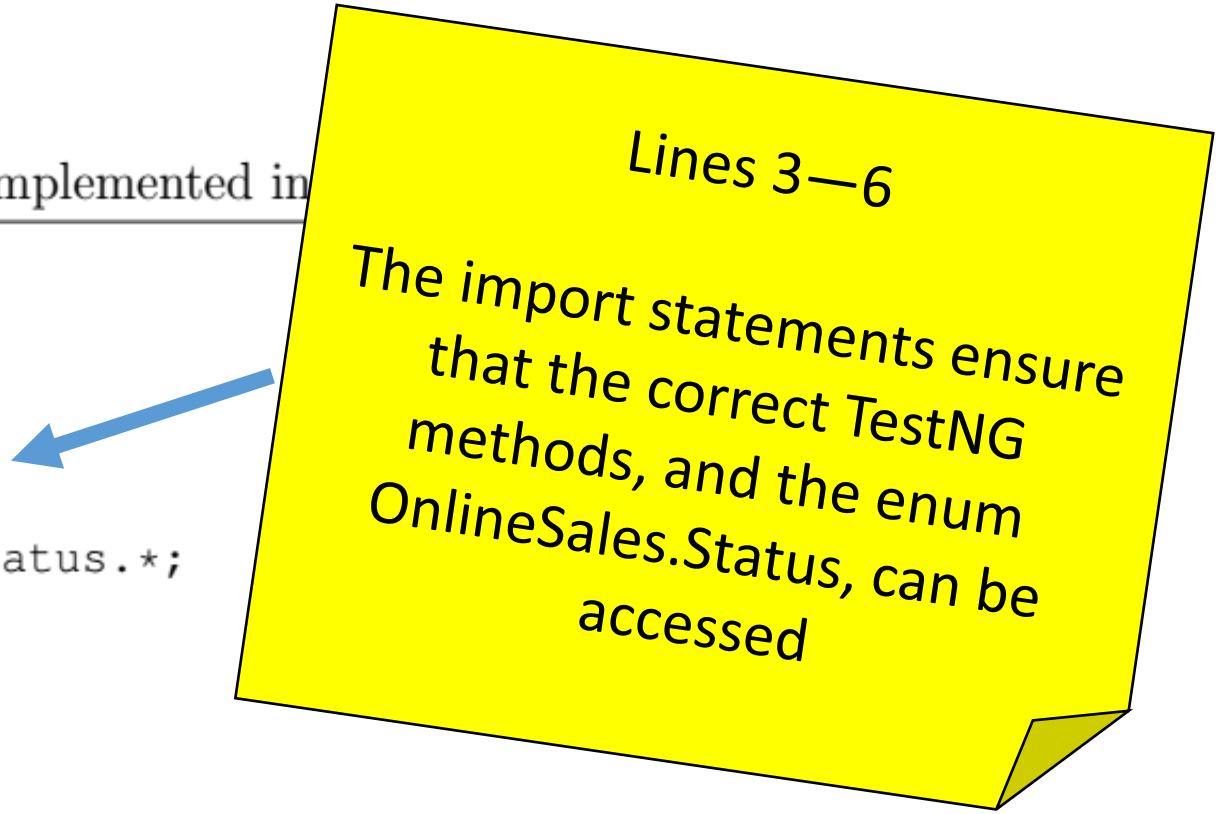
Listing 2.1: Test T1.1 Implemented in TestNG

```
1  package example;
2
3  import static org.testng.Assert.*;
4  import org.testng.annotations.*;
5  import example.OnlineSales.Status;
6  import static example.OnlineSales.Status.*;
7
8  public class OnlineSalesTest {
9
10    // T1.1
11    @Test
12    public void testT1_1() {
13       assertEquals( OnlineSales.giveDiscount(40L,true), FULLPRICE );
14    }
15
16 }
```

# OnlineSalesTest.java

Listing 2.1: Test T1.1 Implemented in

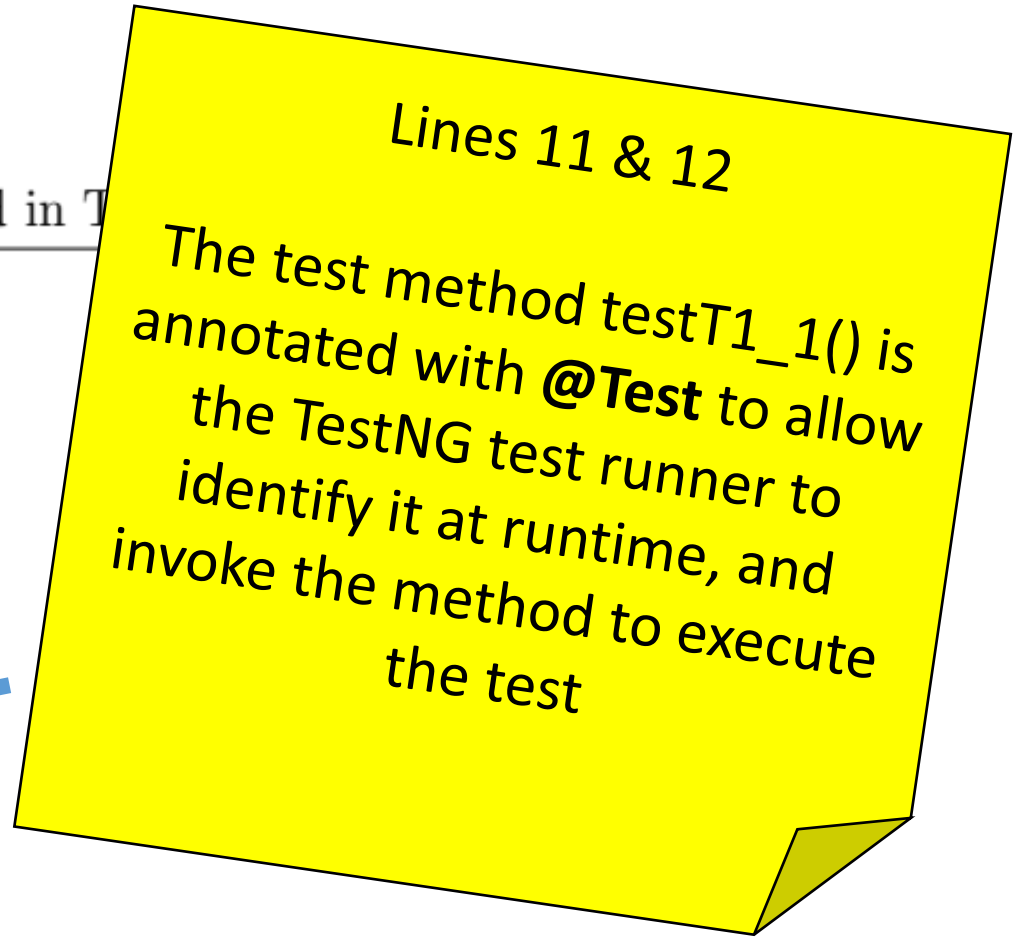
```
1  package example;
2
3  import static org.testng.Assert.*;
4  import org.testng.annotations.*;
5  import example.OnlineSales.Status;
6  import static example.OnlineSales.Status.*;
7
8  public class OnlineSalesTest {
9
10    // T1.1
11    @Test
12    public void testT1_1() {
13       assertEquals( OnlineSales.giveDiscount(40L,true), FULLPRICE );
14    }
15
16 }
```



# OnlineSalesTest.java

Listing 2.1: Test T1.1 Implemented in T

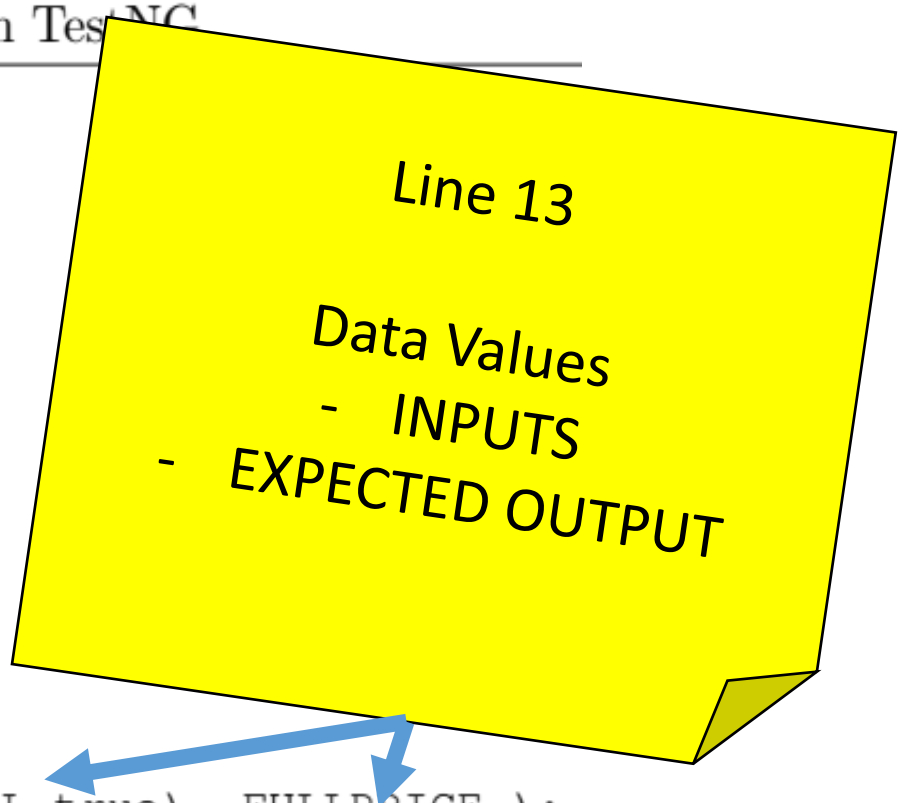
```
1  package example;
2
3  import static org.testng.Assert.*;
4  import org.testng.annotations.*;
5  import example.OnlineSales.Status;
6  import static example.OnlineSales.Status.*;
7
8  public class OnlineSalesTest {
9
10    // T1.1
11    @Test
12    public void testT1_1() {
13       assertEquals( OnlineSales.giveDiscount(40L,true), FULLPRICE );
14    }
15
16 }
```



# OnlineSalesTest.java

Listing 2.1: Test T1.1 Implemented in TestNG

```
1  package example;
2
3  import static org.testng.Assert.*;
4  import org.testng.annotations.*;
5  import example.OnlineSales.Status;
6  import static example.OnlineSales.Status.*;
7
8  public class OnlineSalesTest {
9
10    // T1.1
11    @Test
12    public void testT1_1() {
13       assertEquals( OnlineSales.giveDiscount(40L,true), FULLPRICE );
14    }
15
16 }
```



# OnlineSalesTest.java

Listing 2.1: Test T1.1 Implemented in TestNG

```
1  package example;
2
3  import static org.testng.Assert.*;
4  import org.testng.annotations.*;
5  import example.OnlineSales.Status;
6  import static example.OnlineSales.Status.*;
7
8  public class OnlineSalesTest {
9
10    // T1.1
11    @Test
12    public void testT1_1() {
13       assertEquals( OnlineSales.giveDiscount(40L,true), FULLPRICE );
14    }
15
16 }
```

Line 13

Assertions are used to check that the output from the method being tested is correct.

# OnlineSalesTest.j

Listing 2.1

```
1  package example;
2
3  import static org.testng
4  import org.testng.annot
5  import example.OnlineSa
6  import static example.O
7
8  public class OnlineSal
9
10    // T1.1
11    @Test
12    public void testT1_1() {
13       assertEquals( OnlineSales.giveDiscount(40L,true), FULLPRICE );
14    }
15
16 }
```

Line 13

assertEquals(<actual-value>,<expected-value>) raises an exception if the values are not equal, and the TestNG framework catches this and records it as a failed test.

If the @Test method runs to completion, without any exceptions, then this is recorded as a passed test

# OnlineSalesTest.j

Listing 2.1

```
1  package example;
2
3  import static org.testng
4  import org.testng.annot
5  import example.OnlineSa
6  import static example.O
7
8  public class OnlineSal
9
10    // T1.1
11    @Test
12    public void testT1_1() {
13       assertEquals( OnlineSales.giveDiscount(40L,true), FULLPRICE );
14    }
15
16 }
```

Line 13

assertEquals(<actual-value>,<expected-value>)

The <actual-value> is the return value from calling OnlineSales.giveDiscount(40L,true)

The <expected-value> is the constant FULLPRICE

# OnlineSalesTest.java/T1.2

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

Listing 2.2: Tests T1.1 and T1.2

```
1  public class OnlineSalesTest {
2
3     // T1.1
4     @Test
5     public void testT1_1() {
6        assertEquals( OnlineSales.giveDiscount(40L,true), FULLPRICE );
7     }
8
9     // T1.2
10    @Test
11    public void testT1_2() {
12       assertEquals( OnlineSales.giveDiscount(100L,false), FULLPRICE );
13    }
14
15 }
```

# OnlineSalesTest.java/T1.2

Listing 2.2: Tests T1.1 a

```
1  public class OnlineSalesTest {
2
3      // T1.1
4      @Test
5      public void testT1_1() {
6          assertEquals( OnlineSales.giveDiscount(40L,true), FULLPRICE );
7      }
8
9      // T1.2
10     @Test
11     public void testT1_2() {
12         assertEquals( OnlineSales.giveDiscount(100L,false), FULLPRICE );
13     }
14
15 }
```

What do you notice?

# OnlineSalesTest.java/T1.2

Listing 2.2: Tests T1.1 a

```
1  public class OnlineSalesTest {
2
3      // T1.1
4      @Test
5      public void testT1_1() {
6          assertEquals( OnlineSales.giveDiscount(40L,true), FULLPRICE );
7      }
8
9      // T1.2
10     @Test
11     public void testT1_2() {
12         assertEquals( OnlineSales.giveDiscount(100L,false), FULLPRICE );
13     }
14
15 }
```

Note the duplication!

testT1_2() is the same as T1_1() with different data values

# Parameterised Tests

- To reduce the code duplication, and allow the same test code to be used with different data, parameterised (or data-driven) tests can be used
- Most test tools provide this facility
- In TestNG it is called a “Data Provider”

# OnlineSalesTest with EP Coverage and Data Provider

Listing 2.3: OnlineSalesTest with EP

```
1  public class OnlineSalesTest {
2
3     // EP test data
4     private static Object[][] testData1 = new Object[][] {
5     // test, bonuspoints, goldCustomer, expected output
6        { "T1.1", 40L, true, FULLPRICE },
7        { "T1.2", 100L, false, FULLPRICE },
8        { "T1.3", 200L, false, DISCOUNT },
9        { "T1.4", -100L, false, ERROR },
10     };
```

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

# OnlineSalesTest with EP Coverage

Listing 2.3: OnlineSalesTest with EP

```
1  public class OnlineSalesTest {
2
3     // EP test data
4     private static Object[][] testData1 = new Object[][] {
5     // test, bonuspoints, goldCustomer, expected output
6        { "T1.1", 40L, true, FULLPRICE },
7        { "T1.2", 100L, false, FULLPRICE },
8        { "T1.3", 200L, false, DISCOUNT },
9        { "T1.4", -100L, false, ERROR },
10    };
11
12    // Method to return the EP test data
13    @DataProvider(name="dataset1")
14    public Object[][] getTestData() {
15       return testData1;
16    }
```

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

# OnlineSalesTest with EP Coverage

| ID | TCI Covered | Inputs | | Exp. Results |
|---|---|---|---|---|
| | | bonusPoints | goldCustomer | return value |
| T1.1 | EP2,5,7 | 40 | true | FULLPRICE |
| T1.2 | EP3,6,[7] | 100 | false | FULLPRICE |
| T1.3 | EP4,[6],8 | 200 | false | DISCOUNT |
| T1.4* | EP1*,9 | -100 | false | ERROR |

Listing 2.3: OnlineSalesTest with EP

```
1  public class OnlineSalesTest {
2
3     // EP test data
4     private static Object[][] testData1 = new Object[][] {
5     // test, bonuspoints, goldCustomer, expected output
6        { "T1.1", 40L, true, FULLPRICE },
7        { "T1.2", 100L, false, FULLPRICE },
8        { "T1.3", 200L, false, DISCOUNT },
9        { "T1.4", -100L, false, ERROR },
10    };
11
12    // Method to return the EP test data
13    @DataProvider(name="dataset1")
14    public Object[][] getTestData() {
15       return testData1;
16    }
17
18    // Method to execute the EP tests
19    @Test(dataProvider="dataset1")
20    public void test_giveDiscount( String id, long bonusPoints,
21          boolean goldCustomer, Status expected)
22    {
23       assertEquals(
24       OnlineSales.giveDiscount(bonusPoints, goldCustomer),
25             expected );
26    }
27
28 }
```

```
3     // EP test data
4     private static Object[][] testData1 = new Object[][] {
5     // test, bonuspoints, goldCustomer, expected out
6        { "T1.1", 40L, true, FULLPRICE },
7        { "T1.2", 100L, false, FULLPRICE },
8        { "T1.3", 200L, false, DISCOUNT },
9        { "T1.4", -100L, false, ERROR },
10    };
11
12    // Method to return the EP test data
13    @DataProvider(name="dataset1")
14    public Object[][] getTestData() {
15       return testData1;
16    }
17
18    // Method to execute the EP tests
19    @Test(dataProvider="dataset1")
20    public void test_giveDiscount( String id, long bonusPoints,
21          boolean goldCustomer, Status expected)
22    {
23       assertEquals(
24       OnlineSales.giveDiscount(bonusPoints, goldCustomer),
25             expected );
26    }
```

Line 13: @DataProvider annotation defines the method getTestData() as a data provider with the name “dataset1”

Line 19: @Test annotation specifies the name of the data provider “dataset1” required for the test method test giveDiscount()

```
3     // EP test data
4     private static Object[][] testData1 = new Object
5     // test, bonuspoints, goldCustomer, expected out
6        { "T1.1", 40L, true, FULLPRICE },
7        { "T1.2", 100L, false, FULLPRICE },
8        { "T1.3", 200L, false, DISCOUNT },
9        { "T1.4", -100L, false, ERROR },
10    };
11
12    // Method to return the EP test data
13    @DataProvider(name="dataset1")
14    public Object[][] getTestData() {
15       return testData1;
16    }
17
18    // Method to execute the EP tests
19    @Test(dataProvider="dataset1")
20    public void test_giveDiscount( String id, lo
21          boolean goldCustomer, Status expected)
22    {
23       assertEquals(
24       OnlineSales.giveDiscount(bonusPoints, goldCustomer),
25             expected );
26    }
```

Lines 6-9: testData1[] initialised using the Test Case data

ID's added as a string to allow failed tests to be identified

Line 15: getTestData() returns this test data array

Recommendation: initialise test data arrays at the top of the class, as shown, and not in the method

```
3       // EP test data
4       private static Object[][] testData1 = new Object[][] {
5       // test, bonuspoints, goldCustomer, expe
6          { "T1.1", 40L, true, FULLPRICE },
7          { "T1.2", 100L, false, FULLPRICE },
8          { "T1.3", 200L, false, DISCOUNT },
9          { "T1.4", -100L, false, ERROR },
10      };
11
12      // Method to return the EP test data
13      @DataProvider(name="dataset1")
14      public Object[][] getTestData() {
15         return testData1;
16      }
17
18      // Method to execute the EP tests
19      @Test(dataProvider="dataset1")
20      public void test_giveDiscount( Strin
21            boolean goldCustomer, Status e
22      {
23         assertEquals(
24         OnlineSales.giveDiscount(bonusPo
25               expected );
26      }
```

The TestNG test runner:

1. Finds @Test method and extracts the data provider name
2. Finds the method with the matching @DataProvider name
3. Calls the data provider, which returns the array of test data
4. Calls the test method repeatedly with each row of test data from the array in turn

```
3       // EP test data
4       private static Object[][] testData1 = new Object[][] {
5       // test, bonuspoints, goldCustomer, expe
6          { "T1.1", 40L, true, FULLPRICE },
7          { "T1.2", 100L, false, FULLPRICE },
8          { "T1.3", 200L, false, DISCOUNT },
9          { "T1.4", -100L, false, ERROR },
10      };
11
12      // Method to return the EP test data
13      @DataProvider(name="dataset1")
14      public Object[][] getTestData() {
15         return testData1;
16      }
17
18      // Method to execute the EP tests
19      @Test(dataProvider="dataset1")
20      public void test_giveDiscount( Strin
21            boolean goldCustomer, Status e
22      {
23         assertEquals(
24         OnlineSales.giveDiscount(bonusPo
25               expected );
26      }
```

The TestNG test runner:

1. Finds @Test method and extracts the data provider name
2. Finds the method with the matching @DataProvider name
3. Calls the data provider, which returns the array of test data
4. Calls the test method repeatedly with each row of test data from the array in turn

```
3       // EP test data
4       private static Object[][] testData1 = new Object[][] {
5       // test, bonuspoints, goldCustomer, expe
6          { "T1.1", 40L, true, FULLPRICE },
7          { "T1.2", 100L, false, FULLPRICE },
8          { "T1.3", 200L, false, DISCOUNT },
9          { "T1.4", -100L, false, ERROR },
10      };
11
12      // Method to return the EP test data
13      @DataProvider(name="dataset1")
14      public Object[][] getTestData() {
15         return testData1;
16      }
17
18      // Method to execute the EP tests
19      @Test(dataProvider="dataset1")
20      public void test_giveDiscount( Strin
21            boolean goldCustomer, Status e
22      {
23         assertEquals(
24         OnlineSales.giveDiscount(bonusPo
25               expected );
26      }
```

The TestNG test runner:

1. Finds @Test method and extracts the data provider name

2. Finds the method with the matching @DataProvider name

3. Calls the data provider, which returns the array of test data

4. Calls the test method repeatedly with each row of test data from the array in turn

```
3       // EP test data
4       private static Object[][] testData1 = new Object[][] {
5       // test, bonuspoints, goldCustomer, expe
6          { "T1.1", 40L, true, FULLPRICE },
7          { "T1.2", 100L, false, FULLPRICE },
8          { "T1.3", 200L, false, DISCOUNT },
9          { "T1.4", -100L, false, ERROR },
10      };
11
12      // Method to return the EP test data
13      @DataProvider(name="dataset1")
14      public Object[][] getTestData() {
15         return testData1;
16      }
17
18      // Method to execute the EP tests
19      @Test(dataProvider="dataset1")
20      public void test_giveDiscount(Strin
21            boolean goldCustomer, Status e
22      {
23         assertEquals(
24         OnlineSales.giveDiscount(bonusPo
25               expected );
26      }
```

The TestNG test runner:

1. Finds @Test method and extracts the data provider name

2. Finds the method with the matching @DataProvider name

3. Calls the data provider, which returns the array of test data

4. Calls the test method repeatedly with each row of test data from the array in turn

```
3     // EP test data
4     private static Object[][] testData1 = new Object[][] {
5     // test, bonuspoints, goldCustomer, expected output
6        { "T1.1", 40L, true, FULLPRICE },
7        { "T1.2", 100L, false, FULLPRICE },
8        { "T1.3", 200L, false, DISCOUNT },
9        { "T1.4", -100L, false, ERROR },
10    };
11
12    // Method to return the EP test data
13    @DataProvider(name="dataset1")
14    public Object[][] getTestData() {
15       return testData1;
16    }
17
18    // Method to execute the EP tests
19    @Test(dataProvider="dataset1")
20    public void test_giveDiscount( String id, long bonusPoints,
21          boolean goldCustomer, Status expected)
22    {
23       assertEquals(
24       OnlineSales.giveDiscount(bonusPoints, goldCustomer),
25             expected );
26    }
```

On each call, the parameters are filled using data from the data provider

**Example: 1st call**

*id="T1.1"*

bonusPoints=40L

goldCustomer=true

expected=FULLPRICE

```
3     // EP test data
4     private static Object[][] testData1 = new Object[][] {
5     // test, bonuspoints, goldCustomer, expected output
6        { "T1.1", 40L, true, FULLPRICE },
7        { "T1.2", 100L, false, FULLPRICE },
8        { "T1.3", 200L, false, DISCOUNT },
9        { "T1.4", -100L, false, ERROR },
10    };
11
12    // Method to return the EP test data
13    @DataProvider(name="dataset1")
14    public Object[][] getTestData() {
15       return testData1;
16    }
17
18    // Method to execute the EP tests
19    @Test(dataProvider="dataset1")
20    public void test_giveDiscount( String id, long bonusPoints,
21          boolean goldCustomer, Status expected)
22    {
23       assertEquals(
24       OnlineSales.giveDiscount(bonusPoints, goldCustomer),
25             expected );
26    }
```

On each call, the parameters are filled using data from the data provider

**Example: 2nd call**

*id="T1.2"*

*bonusPoints=100L*

*goldCustomer=false*

*expected=FULLPRICE*

```
3     // EP test data
4     private static Object[][] testData1 = new Object[][] {
5     // test, bonuspoints, goldCustomer, expected output
6        { "T1.1", 40L, true, FULLPRICE },
7        { "T1.2", 100L, false, FULLPRICE },
8        { "T1.3", 200L, false, DISCOUNT },
9        { "T1.4", -100L, false, ERROR },
10    };
11
12    // Method to return the EP test data
13    @DataProvider(name="dataset1")
14    public Object[][] getTestData() {
15       return testData1;
16    }
17
18    // Method to execute the EP tests
19    @Test(dataProvider="dataset1")
20    public void test_giveDiscount( String id, long bonusPoints,
21          boolean goldCustomer, Status expected)
22    {
23       assertEquals(
24       OnlineSales.giveDiscount(bonusPoints, goldCustomer),
25             expected );
26    }
```

On each call, the parameters are filled using data from the data provider

**Example: 3rd call**

*id="T1.3"*

*bonusPoints=200L*

*goldCustomer=false*

*expected=DISCOUNT*

```
3     // EP test data
4     private static Object[][] testData1 = new Object[][] {
5     // test, bonuspoints, goldCustomer, expected output
6        { "T1.1", 40L, true, FULLPRICE },
7        { "T1.2", 100L, false, FULLPRICE },
8        { "T1.3", 200L, false, DISCOUNT },
9        { "T1.4", -100L, false, ERROR },
10    };
11
12    // Method to return the EP test data
13    @DataProvider(name="dataset1")
14    public Object[][] getTestData() {
15       return testData1;
16    }
17
18    // Method to execute the EP tests
19    @Test(dataProvider="dataset1")
20    public void test_giveDiscount( String id, long bonusPoints,
21          boolean goldCustomer, Status expected)
22    {
23       assertEquals(
24       OnlineSales.giveDiscount(bonusPoints, goldCustomer),
25             expected );
26    }
```

On each call, the parameters are filled using data from the data provider

**Example: 4th call**

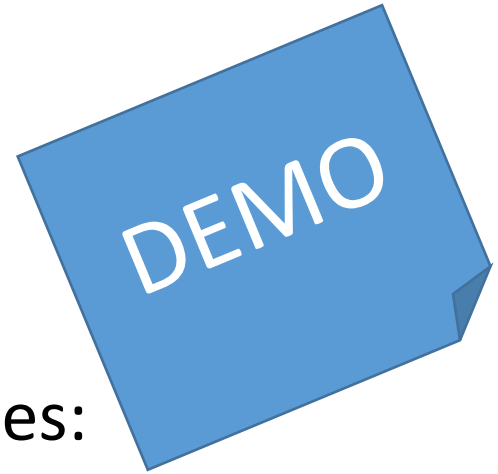
*id="T1.4"*

*bonusPoints=-100L*

*goldCustomer=false*

*expected=ERROR*

# 6. Test Execution




- Running these tests against the class OnlineSales produces:

```
PASSED: test_giveDiscount("T1.1", 40, true, FULLPRICE)
PASSED: test_giveDiscount("T1.2", 100, false, FULLPRICE)
PASSED: test_giveDiscount("T1.3", 200, false, DISCOUNT)
PASSED: test_giveDiscount("T1.4", -100, false, ERROR)
===============================================
Command line suite
Total tests run: 4, Passes: 4, Failures: 0, Skips: 0
===============================================
```

# 7. Test Results

```
PASSED: test_giveDiscount("T1.1", 40, true, FULLPRICE)
PASSED: test_giveDiscount("T1.2", 100, false, FULLPRICE)
PASSED: test_giveDiscount("T1.3", 200, false, DISCOUNT)
PASSED: test_giveDiscount("T1.4", -100, false, ERROR)
=============================================
Command line suite
Total tests run: 4, Passes: 4, Failures: 0, Skips: 0
=============================================
```

- All the tests have passed
- The actual results produced by the software have matched the expected results derived from the specification.

# KEY TERMS

- Equivalence Partitioning & Equivalent Processing
- Value Lines for Natural Ranges & Specification-Based Ranges
- Error Processing
- TCI: **T**est **C**overage **I**tems
- TC: **T**est **C**ases, with Normal and Error cases
- TestNG:
  - TestRunner, @Test, assertEquals
  - DataProvider

# Next Week

- Review Lab 2
- EP
  - Equivalence Partitions in More Detail
    - Fault Model
    - Description, Analysis, Test Coverage Items, Test Cases
    - Pitfalls
  - Evaluation
    - Limitations: injected faults into the code
    - Strengths & Weaknesses
  - Key Points & Notes for Experienced Testers
- Boundary Value Analysis

# This Afternoon

- Lab 2:
  - Analysis, Test Coverage Items, Test Cases, Review, Implement, Execute, Results
  - Deadline: next Monday 13:00 (but try and get it done today)
- Mark via Moodle quiz: Analysis, TCI, TC, review
  - Develop the tests using pen & paper first
  - Read the instructions carefully: **ordering** very important for automated assessment
  - In the quiz, use Y/N for error cases, not * (Moodle Quiz limitations)

# Marking & Grades

- You may discuss the lab work with each other: assist your learning
- But, do the quiz on your own:
  - Quiz
- Your quiz answers should match your written work!!
  - If, while doing the quiz, you notice a mistake in your written work
    - BEFORE you submit the quiz, correct the written work, and then correct the quiz answer
- Quiz marks only returned next Monday (12:00)
  - You may freely navigate the quiz before this, and update your answers

# Independent Study

- Read Chapter 2 of the book
  - Suggestion: also read up on EP in Roper (Library)
- Review your LAB2 lab results (Monday after 13:00)
- Review past exam papers for EP questions
  - https://www.maynoothuniversity.ie/library
  - Exam Papers->Search By Module Code: CS608
  - Note: papers from 2021 onwards most representative – IEEE naming conventions for previous years beware the slightly different terms

# CS608

## Doing the Labs

# First

- **Analysis**

# First

- Analysis
- **Design:**
  - **Test Coverage Items**
  - **Test Cases**
  - **Test Design Verification**

# Then

- Analysis
- Design:
  - Test Coverage Items
  - Test Cases
  - Test Design Verification
- **Implementation**

# Editing the Test Code Template: name

```
/*
 * InsuranceTest - ep
 *
 * Author: stephen brown, 18/1/2021 - replace
this with your own name, id and today's date
 */


…
```

# Editing the Test Code Template: do not change

```
package cs608;

import static org.testng.Assert.*;
import org.testng.annotations.*;
import cs608.Insurance.Status;
import static cs608.Insurance.Status.*;

public class InsuranceTest {
  …
}
```

# Editing the Test Code Template: do not change

```
package cs608;

import static org.testng.Assert.*;
import org.testng.annotations.*;
import cs608.Insurance.Status;
import static cs608.Insurance.Status.*;

public class InsuranceTest {
  …
}
```

# Editing the Test Code Template: do not change

```
package cs608;


import static org.testng.Assert.*;
import org.testng.annotations.*;
import cs608.Insurance.Status;
import static cs608.Insurance.Status.*;


public class InsuranceTest {
  …
}
```

# Editing the Test Code Template: test data

```
…
// This is dummy data - delete it, add you own test data,
and DELETE this comment!

// EP test data
private static Object[][] testData1 = new Object[][] {
    //  test, age,        ncb, lowRisk, expected output
    {"Tx.y",  55, Status.NO,    true,              500},
};
…
```

# Editing the Test Code Template: test data

```
…
// This is dummy data - delete it, add you own test data,
and DELETE this comment!


// EP test data
private static Object[][] testData1 = new Object[][] {
    //  test, age,        ncb, lowRisk, expected output
    {"Tx.y",  55, Status.NO,    true,                500},
};
…
```

# Editing the Test Code Template: do not change

```
// Method to return the EP test data
@DataProvider(name="dataset1")
}
// Method to execute the EP tests
@Test(dataProvider="dataset1")
public void test_premium( String id, int age,
    Status ncb, boolean lowRisk, int expected) {
    …
}
```

# Editing the Test Code Template: do not change

```
// Method to return the EP test data
@DataProvider(name="dataset1")
}
// Method to execute the EP tests
@Test(dataProvider="dataset1")
public void test_premium( String id, int age,
    Status ncb, boolean lowRisk, int expected) {
    …
}
```

# Finally

- Analysis
- Design:
  - Test Coverage Items
  - Test Cases
  - Test Design Verification
- Implementation
- **Test Execution**

  and
- **Review Test Results (and enter in Moodle)**

# Test Execution (lab2 template provided)

```
CS608> @gradlew correct:test --rerun-tasks
```

NOTE: this command is in your lab instructions

# Test Execution of lab2 template provided (SUCCESS)

```
CS608> @gradlew correct:test --rerun-tasks
Starting a Gradle Daemon, 1 incompatible Daemon could not be reused, use --status for details

> Task :correct:test

Gradle suite > Gradle test > cs608.InsuranceTest > test_premium[0](Tx.y, 55, NO, true, 500) PASSED

BUILD SUCCESSFUL in 11s
3 actionable tasks: 3 executed
CS608>
```

# HTML Test Report: correct/build/reports/tests/test/index.html

## Test Summary

| 1 | 0 | 0 | 0.005s |
|---|---|---|---|
| tests | failures | ignored | duration |

**100%**
successful

**Packages** Classes

| Package | Tests | Failures | Ignored | Duration | Success rate |
|---|---|---|---|---|---|
| cs608 | 1 | 0 | 0 | 0.005s | 100% |

Generated by Gradle 8.1.1 at 15 Feb 2024, 10:52:26

# HTML Test Report (click on CS608)

## Package cs608

all > cs608

| 1 | 0 | 0 | 0.005s |
|---|---|---|---|
| tests | failures | ignored | duration |

**100%**
successful

**Classes**

| Class | Tests | Failures | Ignored | Duration | Success rate |
|---|---|---|---|---|---|
| InsuranceTest | 1 | 0 | 0 | 0.005s | 100% |

Generated by Gradle 8.1.1 at 15 Feb 2024, 10:52:26

# HTML Test Report (click on InsuranceTest)

## Class cs608.InsuranceTest

all > cs608 > InsuranceTest

| 1 | 0 | 0 | 0.005s |
|---|---|---|---|
| tests | failures | ignored | duration |

**100%**
successful

**Tests**

| Test | Duration | Result |
|---|---|---|
| test_premium[0](Tx.y, 55, NO, true, 500) | 0.005s | passed |

Generated by Gradle 8.1.1 at 15 Feb 2024, 10:52:26

# Example: full gradle output (FAILURE)

```
CS608> @gradlew [redacted] --rerun-tasks

> Task :[redacted]:test FAILED

Gradle suite > Gradle test > cs608.InsuranceTestSA > test_premium[ ](T1.[redacted]) FAILED
    java.lang.AssertionError at InsuranceTestSA.java:37

[redacted] tests completed, 1 failed

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':[redacted]:test'.
> There were failing tests. See the report at: file:[redacted] build/reports/tests/test/index.html

* Try:
> Run with --stacktrace option to get the stack trace.
> Run with --info or --debug option to get more log output.
> Run with --scan to get full insights.

* Get more help at https://help.gradle.org

BUILD FAILED in 2s
4 actionable tasks: 4 executed
```