CS265 Software Testing
Lab Week 2
EP & BVA
2.5% (+ 2.5% Bonus)

Exercise 1

Consider the following method signature (interface code) from an airline booking system that decides if a seat can be booking:

```
public static Boolean seatsAvailable(int freeSeats, int seatsRequired);
```

And the following test table that was developed to test the code above:

ID	Test Cases Covered	Inputs		Expected Output
		freeSeats	**seatsRequired**	
T1	2, 7, 10	50	75	FALSE
T2	3, 6, 9	50	25	TRUE
T3	1*	-100	25	FALSE
T4	4*	200	25	FALSE
T5	5*	50	-100	FALSE
T6	8*	50	200	FALSE

Tasks:

- Create an Eclipse project CS265_Lab2 and a package cs265, just like in the first lab, where you place your code and your test code.
- Create a class AirlineSeatReservation. Insert the code from AirlineSeatReservation.java that you find on Moodle that also contains the function seatsAvailable. Note: You do not need to understand the code as you are black-box testing the function as a tester.
- Create a TestNG class AirlineSeatReservationTest that tests the method.
- Implement all six tests based on the table above. Use separate test methods in your test code and name them so that you can easily differentiate them (as we have learnt in class).
- Run the tests and keep the console output; you will need it for the **Moodle quiz** later.

Notes:

- You may need to install some TestNG packages, depending on which computer you use.
- Make sure you place the testing class together with the class that you are testing to avoid errors. For that, it is best to place them in the same cs265 package.

Exercise 2: Equivalence Partitions

A bank customer can deposit money in a bank savings account and money is represented as an Integer. Different amounts result in different interest rates:

a) 3% interest rate is awarded if the balance is between €1 and €100.
b) 5% interest rate is awarded if the balance is between €101 and €1000.
c) 7% interest rate is awarded if the balance is above €1000.

Suppose that the balance of the account is €0. The deposit amount must be a positive number. Any error leads to a return value of 0. All of this is implemented in the function

```
double discountRate(int deposit)
```

that determines the interest rate based on above rules. This function DOES NOT calculate the actual interest; only the interest rate (as described above).

Tasks

Use Equivalence Partitioning to test the function:

1. Design test cases and tests (with test data), where you indicate normal cases and error cases. Use the file `CS265_Lab2_Tables` on Moodle (either Word or LibreOffice) or draw on paper. Keep your final tables for the **Moodle quiz** later.
2. Write TestNG tests that implement your designed test cases. Use the code provided on Moodle (SavingsAccount.java). Generate the test code like in the first lab.

Note: We use another TestNG function

```
assertEquals(double expected, double actual, double delta)
```

that is applied in case a data type is a floating point number (e.g. a double number). The new delta parameter in this function allows for certain margins of error that naturally occur when we operate with floating-point numbers (e.g. 0.03) in the output. Delta can be 0.0001, but try different variations and see if it affects the test result.

Exercise 3: Boundary Value Analysis

Tasks

Use the **Boundary Value Analysis** technique to test the function double discountRate(int deposit), based on the specification from exercise 2 (with all conditions staying the same):

1. Design test cases and tests (with test data), where you indicate normal cases and error cases. Use the file CS2652017_Lab2_Tables on Moodle (either Word or LibreOffice) or draw on paper. Keep your final tables for the **Moodle quiz** later.
2. Write TestNG tests that implement your designed test cases. Use the code provided on Moodle (SavingsAccount.java). Generate the test code like in the first lab.

The lab is worth 5%. You must complete the Moodle quiz to earn these.