

Fundamentals of Software Testing*

(A.Y. 2022/2023) – Duration: 1h30m
February 29th, 2024

Exercise 1.

Consider the following program written to establish the total amount that a student has to pay to enroll at the university¹.

```
1 package fst.unicam.it;
2
3 public class StudentManagement {
4
5     public static double taxes2Pay(Student student) {
6         double lateEnrollmentRate = 0.05;
7         double totalAmount = 500;
8         double extraTax = 200;
9         if normalEnrollmentExpired() {
10             totalAmount = totalAmount * (1 + lateEnrollmentRate)
11         }
12         if (student.status == StudentStatus.late &&
13             !((student.status != StudentStatus.late || totalAmount > 500) ||
14              totalAmount < 1000)
15             )
16             totalAmount += extraTax;
17         else
18             if (student.status == student.worklate && student.status == student.work)
19                 totalAmount += extraTax * 0.6;
20         return totalAmount;
21     }
22
23     public static double normalEnrollmentExpired() {
24         /* The function returns true if the deadline for normal
25            enrollment has expired otherwise it returns false */
26         ...
27     }
28 }
```

Apply a test derivation strategy that intends to spot all the possible mistakes (boolean, relational and expressions) related to the conditions of the program. As any good programmer, in case you judge it useful, you can revise the conditional statements to make them simpler and easier to understand. Obviously the global behaviour should not change.

16 points

Exercise 2.

Consider the program in the previous exercise to perform the following activities:

- Compute the condition/decision coverage for the test suite you derived for Exercise 1 (in case you did not derive any test suite yet, manually derive one with at least 5 tests, and then start from such test suite).
- Derive a data-flow graph for the program above. Some suggestions on how to perform the task:
 - Use line numbers to define the blocks in the data flow.
 - consider the attributes in the class `Student` as different variables.
- Provide an assessment for the all-uses coverage criteria

16 points

*For the QAIS module you should solve Exercise 1 in 45 minutes

¹At the end of the document you find all the complementary classes needed to better understand the behaviour of the program

Supporting Java classes

```
1
2 //studentStatus.java
3 package fst.unicam.it;
4
5 public enum StudentStatus {
6     worker,
7     workerlate,
8     late,
9     standard
10 }
11
12 //Student.java
13 package fst.unican.it;
14
15 public class Student {
16     //fields made public to reduce code verbosity
17     public int matriculation;
18     public Date firstEnrollmentDate;
19     public StudentStatus status;
20     public Date dateOfBirth ;
21
22     public Student(int matriculation, Date fED, StudentStatus status, dateOfBirth dob) {
23         this.matriculation = matriculation;
24         this.firstEnrollmentDate = fED;
25         this.studentStatus = stauts;
26         this.dateOfBirth = dOB;
27     }
28 }
```