Week 11: Testing

```
public class IntegerDivision {
      public static int IntDiv(int x, int y) {
            int z = 0;
            int signe = 1;
            if (x < 0) {
                  signe = -1;
                  x = -x;
            }
            if (y < 0) {
                  signe = -signe;
                  y = -y;
            }
            if (y == 0) {
                  throw new IllegalArgumentException("Argument nul:"+y);
            }
            while (x >= y) {
                  x = x - y;
                  z = z + 1;
            }
            z = signe * z;
            return z;
      }
}
```

Exercise 1 *First step in testing*
Here above is the Java program of the Integer Division (Euclid Algorithm)
Q1. Write (and execute JUnit) tests for this program
Q2. Feed-back

- How many test data did you chose? Why?
- What data did you chose? Why?

Exercise 2 *Coverage-based selection (white-box testing)*
Q1. Draw the control flow-graph of the Integer Division program
Q2. Propose a test data set to achieve line coverage.
How many tests (at least) are necessary to achieve line coverage?
Q3. Does the previous set achieve branch coverage? If necessary complete the test set to achieve branch coverage.
How many tests (at least) are necessary to achieve branch coverage?
Q4. What can you say about your tests now? Is the program free of bugs? Why?

Exercise 3 *black-box testing*
Q1. Apply the input partition technique and all-combination strategy to select tests for the IntegerDivision program
Q2. Complete your previous tests
Q3. What can you say about your tests now? Is the program free of bugs? Why?

Exercise 4 *Checking the quality of your tests*
Line 17: while (x >= y) {

- Replace “>=” by “>”
- Execute your tests

Q1. Why is this modification erroneous?
Q2. Do your tests find the error?
Q3. If it doesn’t, what does it means?

Exercise 5

Hereafter is a program that compares two strings and prints if they are equals or not.

Q1. Propose a test set based on an equivalence partition strategy

```
1.  equal : boolean
2.  string1, string2: string
3.  Read(string1, string2)
4.  equal = (string1.length == string2.length)
5.  if equal then
6.     for (i=1 ; i<= string1.length; i++) {
7.        equal = (string1.character[i] == string2.character[i])
8.     }
9.  if equal then
10.    print(« same strings »)
11. else
12.    print(« different strings »)
```

Q2. Draw the control-flow graph of the program.

Q3. Check if the test set produced in question Q1 covers all statements and all branches of the control-flow graph.

Q4. What is the fault in the program?

Exercise 6

Let "example(x,y,z)" be a program with 3 inputs.

Let {x1,x2,x3}, {y1,y2,y3} and {z1,z2} be the selected data for testing.

Q1. How many tests do we obtain with an all-combination approach?

Q2. Propose a test set satisfying a pairwise coverage

HomeWork: Read the following documents

http://www.tutorialspoint.com/software_testing/software_testing_levels.htm

http://www.testingexcellence.com/common-myths-test-automation/

At the end of the week (for the final evaluation), you have to be able to

1. draw the control-flow graph of a simple program
2. select test data in order to achieve line coverage and/or branch coverage
3. use input partitioning method and combination strategies to select data from the specification
4. know the testing philosophy and the practical considerations

Additional exercises to prepare the exam

Exercise 1

Explain the limits of the following requirements. Propose reformulation.

- From the application code, interface with the package must be as simple as possible.
- Installation of the software must be easy and straightforward.

Exercise 2

What are the risks when using an interview method as elicitation technique?

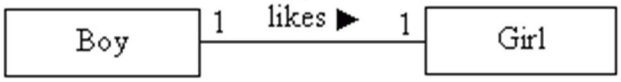
Exercise 3

Q1. What are the weaknesses of a development model based on agile methods?

Q2. What is the main difference between an iterative and an incremental life-cycle?

Exercise 4

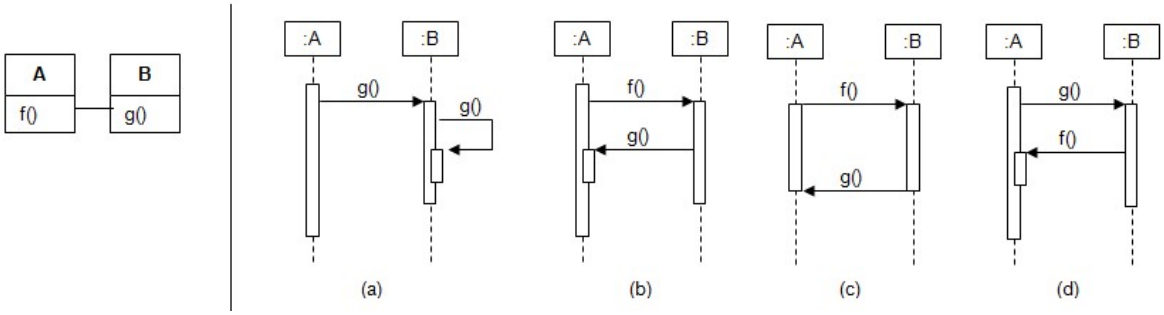
4.1 Consider the following class diagram[1]:



Based on the class diagram above, is there any way to express with an object diagram that Jane (a girl) likes Brian (a boy)?

4.2 *Mark each of the statements below true or false, according to the following diagram1*[1].

a. (True / False) A player can play for two NBA or NFL teams simultaneously.
b. (True / False) A player can be traded among teams.



4.3 Which of the following sequence diagrams given above are potentially valid for the class diagram?

Exercise 5

Create an UML class diagram that models the data relationships described in the following paragraph[2].

We want to model a system for management of flights and pilots. An airline operates flights. Each airline has an ID. Each flight has an ID a departure airport and an arrival airport: an airport as a unique identifier. Each flight has a pilot and a co-pilot, and it uses an aircraft of a certain type; a flight has also a departure time and an arrival time.
An airline owns a set of aircrafts of different types. An aircraft can be in a working state or it can be under repair. In a particular moment an aircraft can be landed or airborne. A company has a set of pilots: each pilot has an experience level: 1 is minimum, 3 is maximum.
A type of airplane may need a particular number of pilots, with a different role (e.g.: captain, co-pilot, navigator): there must be at least one captain and one co-pilot, and a captain must have a level 3.

Exercise 6

You find the following description on Wikipedia.
"Three-tier architecture is a client–server software architecture pattern in which the user interface (presentation), functional process logic ("business rules"), computer data storage and data access are developed and maintained as independent modules, most often on separate platforms Typically, the user interface runs on a desktop PC or workstation and uses a standard graphical user interface, functional process logic that may consist of one or more separate modules running on a workstation or application server, and an RDBMS on a database server or mainframe that contains the computer data storage logic. Data transfer between tiers is part of the architecture.

6.1 To explain the three-tier architecture, draw an architectural description using the principles of (1) separation of concerns and (2) abstraction.

[1] http://sce2.umkc.edu/BIT/burrise/pl/modeling/qanda.html
[2] http://softeng.polito.it/tongji/SE/ex/exercises-w2.pdf

Exercise 7

Hereafter is a program that computes the current day number. It is a number between 1 and 366. January 1st is day 1. For a leap year (also known as an intercalary year or a bissextile year), the 31st of December is day 366. It is day 365 for common year.

```
 1 public int CurrentDayNumber (int day, int month, int year){
 2  if (InvalideDate(day, month, year)){
 3     throw new IllegalArgumentException();
 4  }
 5  int diff = (month-1)* 31+day;
 6  final int[]monthOf30Days = {4,6,9,11};
 7  int i = 0;
 8  while(i< monthOf30Days.length){
 9     if (month > monthOf30Days[i]){
10      diff = diff -1;
11    }
12    i++;
13  }
14  if (month > 2){
15    if (! leap(year)){
16      diff = diff -3;
17    }
18    else {
19       diff = diff -2;
20    }
21  }
22  return diff;
23 }
24
```

7.1 Draw the control-flow graph of the program.

7.2 Give a test data set that covers all branches of the programs.