# Experiment : Coverage-based (white box) Unit Testing

# Experiment

- **An exercise / experiment involving the learning of how test Java classes with Junit and an approach based on structural (white box) testing will be proposed in the practical activities of the course**

# First Step : Pre-questionnaire

- **All the participants to the course and to the experiment have to fill the following form until October 31**
- **https://forms.office.com/e/byfTK6JMXi**

# Second Step : Training

- **For learning purposes, a class to be tested is proposed**
- **It should be tested using JUnit and an IDE of your preference, with the aim to obtain the maximum possible structural coverage**
  - It is strongly suggested to use Eclipse as IDE and Instruction Coverage as objective
- **The class to be tested is Subject Parser**
  - It will be available on Teams

# Second Step : training

- **Training is free: it is not requested to submit the test cases and there is not a time constraint**
- **The purpose of the training is to be ready for a classroom exercise to be taken Monday November 4**
- **In case of problems, I am available for explainations about the training test class**

# Subject Parser

- **SubjectParser receives input parameters via a text string passed to the constructor and perform a parsing looking for the elements in the input string**
- **In this text string the first element is a numeric identifier (id), then there is a string (title), finally a range indicator in the form (x/y) or [x/y] with x and y positive integers.**

**Example:**

**1 subject title [1/2]**

**12 test (7/8)**

**The main method is reported mainly as an example but it is not strictly required that it be tested.**

# Third Step : exercise

- **November 4 a Teams Activity will be assigned to all the students in the classroom, about the structural (white box) unit testing of a Java class, similarly to the training case**
- **The test cases have to be designed and implemented in the classroom in a 3-hour exercise and submitted via Teams activity**
  - The presence in the classroom is mandatory in this date
    - *Exceptional cases will be managed, but it is strongly recommended the participation November 4*

# FontInfo

- This class manages information about a Font, with several utility methods to instantiate FontInfo objects, read and write its properties, compare fonts, copy font objects.
- The class also includes an interface that is not relevant for code coverage.
- Even for this class it should not be possible to reach 100% coverage of the Instructions (I got to 95%).

- The class is composed of several methods, almost all of limited complexity.
- Suggestions: you can start with the easyest methods and with the basic executions in order to immediately reach a good coverage.
- You should use the debugging features to know which tests can reach the remaining lines.
- Among the imports you should also consider java.awt.Font, which is used by the class under test (it is already inserted into the template)
- Please add a minimal internal documentation about each test case