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

Fourth step: gamification

- **Gamification is a way to improve attractiveness of difficult or boring tasks**
  - Coverage-based testing could be classified in this family!
- **Benefits of gamification are currently studied in different researches involving students and practitioners**

ENACTEST

ENACTEST is an ERASMUS+ European Project involving a consortium with four academic partners

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

INOVA+ RESEARCH & TECHNOLOGY

INOVA+

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

CESUR Tu Centro Oficial de FP

U.PORTO

RI. SE

nexo qa

KU LEUVEN RESEARCH CENTRE FOR INFORMATION SYSTEMS ENGINEERING (LIRIS)

CTG Academy

Goals of the Enactest project

1

Reduce the skills mismatches of testing education from three perspectives of key actors: students, industry, and academia

2

Develop bite-sized teaching testing materials designed as capsules (also based on gamification) that can be easily and seamlessly integrated in the current curricula

3

Perform pilot studies and experiments to evaluate the effectiveness, efficiency of the designed capsules and the satisfaction for all the key actors in the project.

Testing against robot challenge

- **Software Architecture Design students and thesis students have collaborated in the realization of a gamification environment supporting coverage-based testing**
- **The tool is currently under development but a prototype will be used to support the activities that will be presented in the next slides**

Activity Objective

- To **compare** the capability of students and of different automatic tools for test case generation in a challenge aiming at cover the most possible amount of source code with Junit 4 test cases
- The educational objective is to **stimulate students to develop test code** by comparing the possibilities offered by automatic test case generation tools

Game

- **A Web-based game is available for playing different challenges versus two different tools**
- **The game is available online or it can be installed on a phisical machine**
- **The game instruction document provides a tutorial of installation and play**

Activity Rules

- **Each student has to achieve at least 7 points by beating (or pairing) the coverage results obtained by the proposed robots**

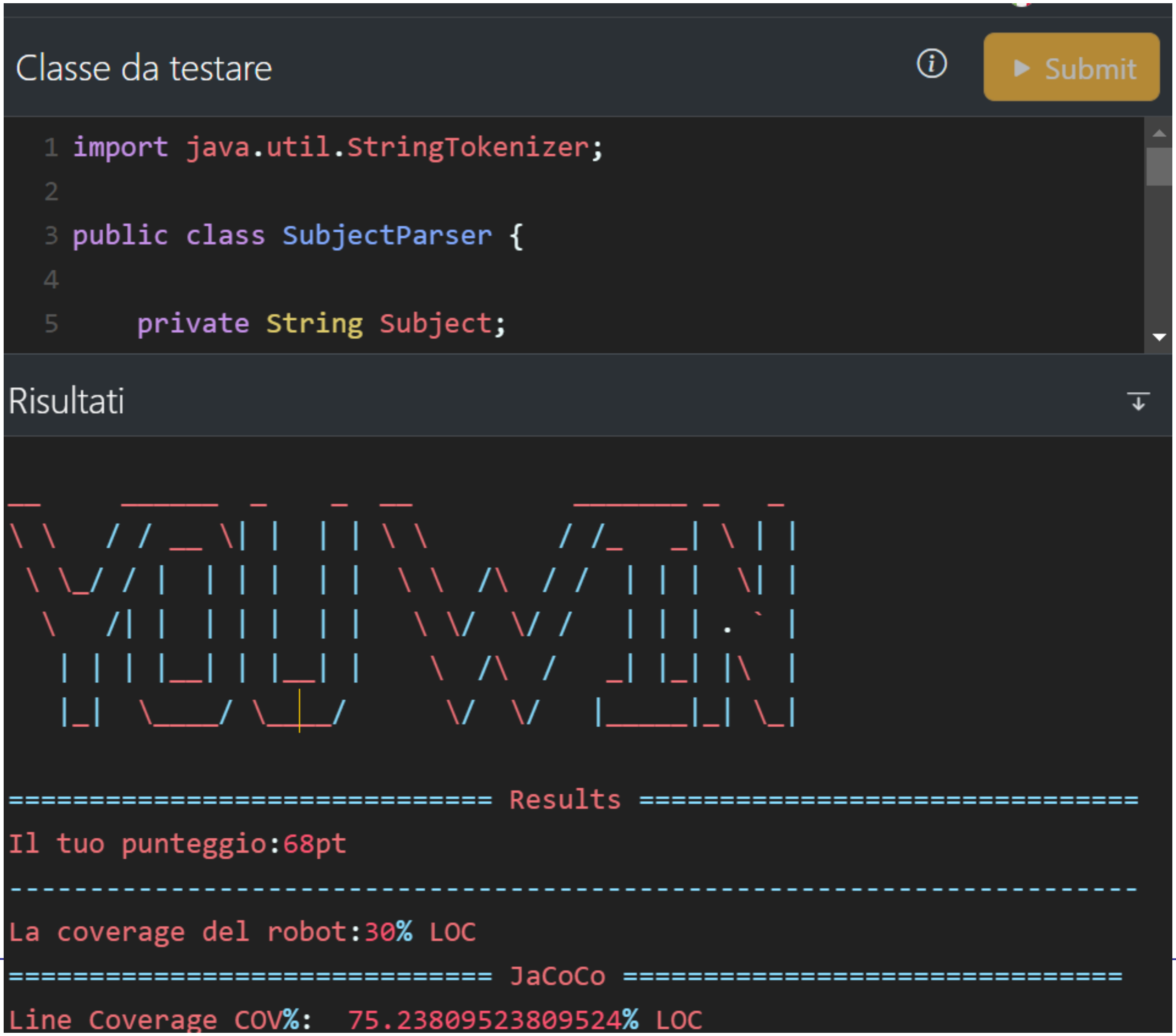
Robots

- **Two different robots have been proposed:**
  - Randoop
  - Evosuite
- **For each of the robots, three different test suites with increasing coverage have been proposed, corresponding to three different difficulty levels:**
  - Easy
  - Medium
  - Difficult

Achieving Points

- **In order to achieve a point, it is needed to view the message YOU WIN from the GUI while playing a Single Challenge and after clicked the Submit button**
  - Exceptionally, will be considered valid also the case in which the LOC coverage is exactly the same of the robot even if a YOU LOSE image is shown
    - *The comparison function is not yet definitive …*

Winning example



```java
import java.util.StringTokenizer;

public class SubjectParser {

    private String Subject;
```

```
__     ______  _    _  __          __ _____ _   _
\ \   / / __ \| |  | | \ \        / /_   _| \ | |
 \ \_/ / |  | | |  | |  \ \  /\  / /  | | |  \| |
  \   /| |  | | |  | |   \ \/  \/ /   | | | . ` |
   | | | |__| | |__| |    \  /\  /   _| |_| |\  |
   |_|  \____/ \____/      \/  \/   |_____|_| \_|

============================== Results ==============================
Il tuo punteggio:68pt
----------------------------------------------------------------------
La coverage del robot:30% LOC
============================== JaCoCo ==============================
Line Coverage COV%:  75.23809523809524% LOC
```

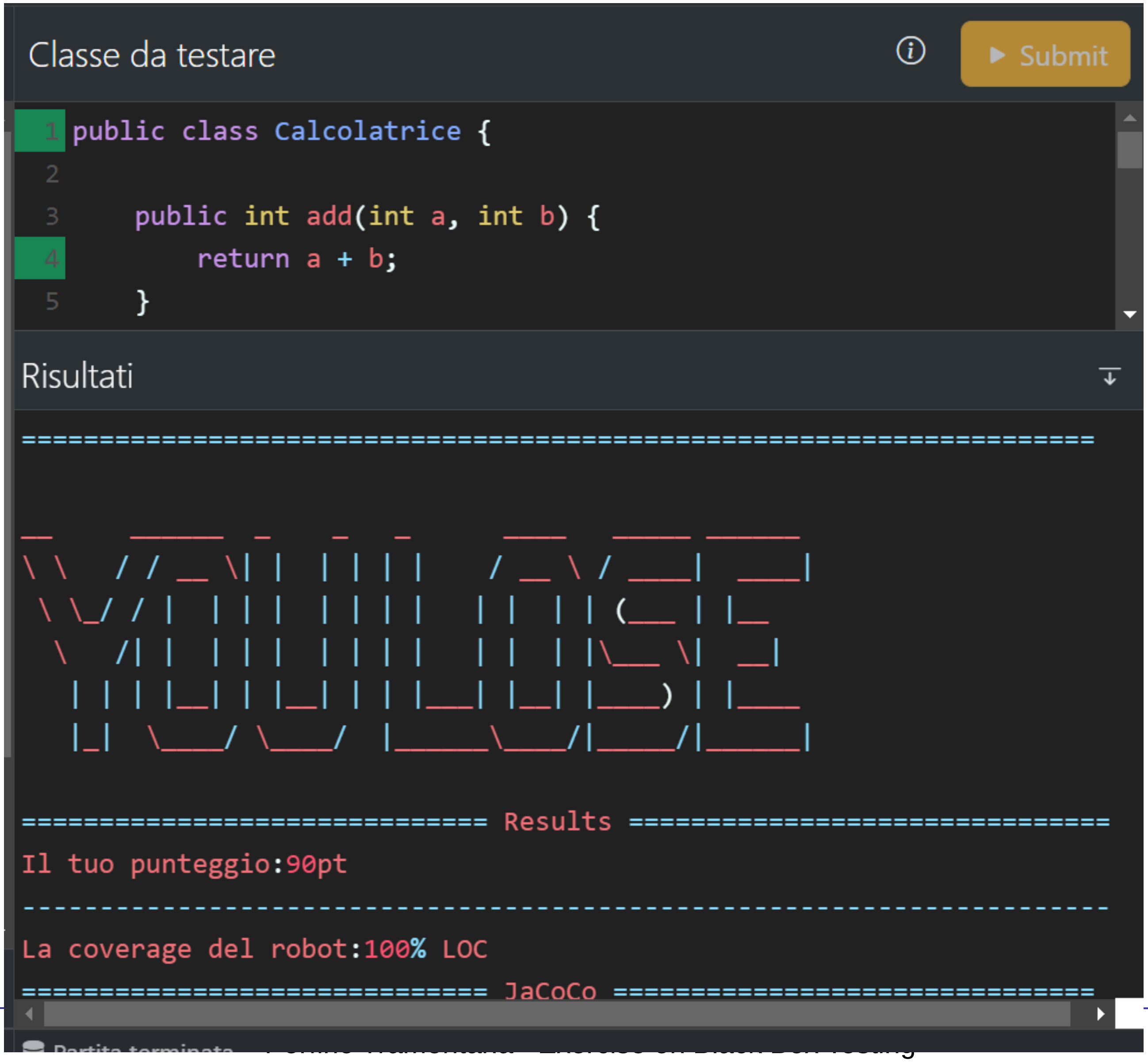
Saving results

- **When a win is obtained, the student has to:**
  1. Save his winning java test class
  2. Copy and paste in a test file the content of the Results window
  3. Save a screenshot showing the winning
     - *Similar to the one reported in the previous slide*
  4. Rename all of them according to the template:
     - ClasseTestataNomeCognomeRobotBattutoLivelloDIfficolta.java , .txt and .png
       - **i.e.** FTPFilePorfirioTramontanaEvosuiteMedium.java
       - FTPFilePorfirioTramontanaEvosuiteMedium.txt
       - FTPFilePorfirioTramontanaEvosuiteMedium.png

Losing a game

- **In case of losing, don’t worry**
  - You can exploit the results to understand how improve test cases
  - You can replay the game until you a better (or equal) result with respect to the robot

Losing game



Classe da testare

▶ Submit

```java
public class Calcolatrice {

    public int add(int a, int b) {
        return a + b;
    }
```

Risultati

```
======================================================================


__     ______  _    _   _      ____   _____ ______
\ \   / / __ \| |  | | | |    / __ \ / ____|  ____|
 \ \_/ / |  | | |  | | | |   | |  | | (___ | |__
  \   /| |  | | |  | | | |   | |  | |\___ \|  __|
   | | | |__| | |__| | | |___| |__| |____) | |____
   |_|  \____/ \____/  |______\____/|_____/|______|


=============================== Results ================================
Il tuo punteggio:90pt
-----------------------------------------------------------------------
La coverage del robot:100% LOC
=============================== JaCoCo ===============================
```

Partita terminata

Clarifications

- **A good test class may be sufficient to achieve 6 points**
  - By beating both the robots at all the three difficulty levels
- **Some robots could be very difficult to be beaten**
  - you can avoid them and select another class to be tested
- **The test class can be saved by the Download function in the menu**

Classes under test

- **Four different test classes have been proposed:**
  - FTPFile
  - HSLColor
  - TimeStamp
  - OutputFormat
- **Test classes documentation is reported in attached documents**
  - With some test examples

Assignments

- **Each student should start his game from the first class in his row of the Assignments spreadsheet**
- **If he has problems with the first class, he can move to the second, third and fourth class until 7 points are reached**

Clarifications

- **The achievement of a better score (more than 7 points) is positively evaluated**
- **Test classes should have an internal documentation to understand the ratio behind the test case design**
  - By means of internal comments (//)
- **Test classes should be original and not copied among students**

Available installations

- **Two instances of the applications are currently available**
  - The addresses are reported in the assignment spreadsheet
  - Each student should use preferably the first one listed in his row and, in case of unavailability, the second one
  - In case of blocks to ngrok, the direct addresses could be used
  - The web application is quite unstable: it is possible that it should be reset by me and in this case a new registration is needed
  - Please provide me a feedback in case of unavailability and I will try to restart it
- **Optionally and/or in case of problems to all the servers, each student can install his own instance of the web application and access via localhost**
  - All the instructions are in the game instruction document

Results questionnaire

- **At the end of the exercise you have to fill questionnaire**
- **For each class you have to provide**
  - The order in which you have tested (the first one, the second one, the third one, the fourth one or no tested if you already achieved sufficient points by testing the other classes)
  - The number of points achieved by beating Randoop and Evosuite
  - The achieved line coverage (readable from the attached screesnshot/ text)
  - The time spent in testing (in minutes)
  - Possible feedbacks / comments / difficulties
- **https://forms.office.com/e/YSw3cMqe4G**

Usefulness questionnaire

- **After sending the results questionnaire, there is a last questionnaire to be compiled**
- **This last questionnaire is about the perceived usefulness of the tool**
  - Its results will be used as feedbacks to evaluate and improve the tool
  - The answers represent your opinions about it and obviously will not be object of evaluation
- **https://forms.office.com/e/cZThkawmvv**