Project

1. High-level goals

The high-level goals of this lab project are to (1) learn about systematic unit testing (2) reason about test quality, using coverage and mutation adequacy criteria.

You will be using an extended version of Defects4J. Defects4J is a collection of reproducible *bugs* and a supporting infrastructure with the goal of advancing software engineering research. By default, Defects4J supports one mutation tool; the extended version supports multiple tools.

2. Setup box

1. Install Oracle VM VirtualBox -- the latest version of VM can be found here.
2. Install Ubuntu 22.04.3 in the VM – the video here may help.
3. Requirements:
    a. Java 1.8
        i. `sudo apt-get install openjdk-8-jdk`
    b. Git >= 1.9
        i. `sudo apt-get install git-all`
    c. SVN >= 1.8
        i. `sudo apt-get install subversion`
    d. Perl >= 5.0.12
        i. `sudo apt-get install perl`
    e. Python 3
        i. `sudo apt-get install python3`
        ii. `sudo apt-get install python3-pip`

For additional information, read the Requirements section reported here.

3. Setup Defects4J

1. `cd $HOME/Desktop`
2. Clone the extended version of Defect4J from here as
    - `git clone https://github.com/PedroTav/defects4j.git`
3. Initialize Defects4J (download the project repositories and external libraries, which are not included in the git repository for size purposes and to avoid redundancies).
    - `cd defects4j`
    - `sudo cpanm --installdeps .`
    - `./init.sh`
    - `sudo pip install pandas==2.1.1`
    - `sudo apt-get install python3-tk`
4. Add Defects4J's executables to your PATH:
    - `gedit ~/.profile`
    - `export PATH="{path2defects4j}/framework/bin:$PATH"`

    - `reboot`
5. Check installation:
    - `defects4j info -p Lang`

```
Summary of configuration for Project: Lang
------------------------------------------------------------------------------
    Script dir: /home/demo/defects4j/framework
      Base dir: /home/demo/defects4j
    Major root: /home/demo/defects4j/major
      Repo dir: /home/demo/defects4j/project_repos
------------------------------------------------------------------------------
    Project ID: Lang
       Program: commons-lang
    Build file: /home/demo/defects4j/framework/projects/Lang/Lang.build.xml
------------------------------------------------------------------------------
           Vcs: Vcs::Git
    Repository: /home/demo/defects4j/project_repos/commons-lang.git
     Commit db: /home/demo/defects4j/framework/projects/Lang/active-bugs.csv
Number of bugs: 64
------------------------------------------------------------------------------
```

4. Assignment

4.1 Subjects under test

	Project (PID)	**Version (VID)**	**Bug Id (BID)**	**Class under test**	**Test class for student-written tests**
P1	Cli	32f	32	org.apache.commons.cli.HelpFormatter	Cli-32f/StudentTest.java (*package: org.apache.commons.cli*)
P2	Gson	15f	15	com.google.gson.stream.JsonWriter	Gson-15f/StudentTest.java (*package: com.google.gson.stream*)
P3	Lang	53f	53	org.apache.commons.lang.time.DateUtils	Lang-53f/StudentTest.java (*package: org.apache.commons.lang.time*)

4.2 Setup

For the tool/*bug* assigned to you (see table below)

1. `cd analyzer`
2. `python3 defects4jGUI.py`
    - You can find a user guide of the Defects4J GUI tool here.
3. Select the project and *bug* version assigned to you from the dropdown menu.
4. **Checkout** to export the source code of the project/*bug* version to a working directory. The project location should be `/$HOME/`. (e.g., `$HOME/Cli-32f`).
5. **Compile** the subject by pressing the `Compile` button.

6. **Load** the project/*bug* version from the dropdown menu.
7. Obtain **coverage** information for the project/*bug* version by pressing the `Coverage` button.
8. Obtain **mutation** information for the project/*bug* version by (1) selecting the mutation tool from the dropdown menu, and (2) by pressing the `Generate` button.

Note: Students assigned with the Judy tool should ignore the mutants colored with a gray background. According to Judy, these mutants are applied to lines -1 or 0, which do not exist.

Student ID	**Student Name**	**Mutation Tool**	***Bug***
M.EIC			
201904795	Ana Matilde Guedes Perez da Silva Barra	PIT	Cli-32
201907925	Catarina Oliveira Pires	Judy	Gson-15
201806829	Fabio Miguel Chen Huang	Major	Gson-15
202200589	Fernando Adriano Ramalho Rocha	Judy	Lang-53
202301425	Filip Balder	Judy	Cli-32
201905337	Francisco Gonçalves Cerqueira	Major	Cli-32
202204185	Francisco Gonçalves Fernandes	Major	Gson-15
201907361	Francisco Pinto de Oliveira	Major	Cli-32
202302723	Frey Dominik	PIT	Gson-15
201906852	Henrique Ribeiro Nunes	PIT	Cli-32
201907878	Joana Teixeira Mesquita	Major	Lang-53
201906478	João Pedro Rodrigues da Silva	Judy	Lang-53
201905962	Luís Filipe Carvalhais dos Santos de Matos	PIT	Gson-15
201906086	Marcelo Henriques Couto	Major	Gson-15
201800177	Marcos William Ferreira Pinto	Judy	Gson-15
201905046	Margarida Assis Ferreira	Major	Lang-53
201905871	Mário Manuel Seixas Travassos	Major	Lang-53
201907756	Miguel Faria Amorim	Judy	Gson-15
202202491	Miguel Ribeiro Azevedo	PIT	Lang-53
201905427	Patrícia do Carmo Nunes Oliveira	PIT	Lang-53
201906712	Pedro Miguel Sampaio Ferreira Machado	Judy	Cli-32
201905853	Rui Filipe Teixeira Alves	Major	Cli-32
201906355	Rui Pedro Mendes Moreira	PIT	Cli-32
202204189	Sara Lima Pereira	PIT	Gson-15

201907483	Valentina Wu	PIT	Lang-53
201907226	Victor Saldanha Nunes	Judy	Cli-32
MESW			
202308706	Abdouroihamani Mze	Judy	Lang-53
202308707	Adalberto Oliveira Filipe	PIT	Gson-15
202200587	Ademola Adekoyejo Plumptre	PIT	Lang-53
202308708	Allan Karlus de Medeiros Ramos	Judy	Lang-53
202308709	Álvaro Manuel Reis Torcato	Major	Gson-15
202308711	Anna Luiza Medeiros Nascimento	PIT	Gson-15
202302324	Bhavesh Parmanand Peswani	Major	Lang-53
202300395	Bruno de Sena Pereira	Major	Cli-32
202310091	Daniel Filipe Carvalho Morais	PIT	Lang-53
202103629	Diogo André Pereira Rodrigues	PIT	Gson-15
202300438	Dominik Klippert	PIT	Cli-32
202300436	Iarina Majeri	Major	Cli-32
202000161	Igor Liberato de Castro	Major	Gson-15
201902617	Isabella Vitória Gonçalves Colombarolli	Major	Cli-32
202308712	Joana Sofia Ferreira Maia	Major	Gson-15
201208144	João Valente Polónia Coelho da Silva	Judy	Gson-15
202000163	José Artur Lima Assunção	PIT	Lang-53
202302818	José Diogo Teixeira Pessoa	Judy	Cli-32
202302380	Juan Ignacio Medone Sabatini	Judy	Cli-32
202302527	Justino Orlando Lumingo Sachilombo	PIT	Lang-53
202308713	Layenis Gonçalves Sobrinho	Judy	Gson-15
201710494	Manuel Enrique Salgado Pietrini	PIT	Cli-32
202308715	Marija Jakovleva	Judy	Lang-53
201304504	Marta Maria de Sousa Cardoso	Major	Cli-32
202300391	Matheus Rodrigues Felizardo	Major	Lang-53
202308716	Muhammad Waqas	Judy	Cli-32
202302862	Rangel Soares de Souza	PIT	Gson-15
200201153	Sara Maria Silva Oliveira	Major	Lang-53
202300431	Sergei Korepanov	PIT	Cli-32
202308718	Tiago Alexandre Delgado de Pina	Major	Gson-15

202300369	Vinicius Correia Honorato	PIT	Cli-32
202308719	Xavier da Silva Costa	Judy	Gson-15

4.3 Assignment

1. Analyze the live mutants and write additional unit tests to kill all-non equivalent mutants.
2. Create a test class named **StudentTest** in the root directory of your assigned project/*bug* version, i.e.,
    - For Cli-32, the **StudentTest** class should live under `$HOME/Cli-32f/`
    - For Gson-15, the **StudentTest** class should live under `$HOME/Gson-15f/`
    - For Lang-53, the **StudentTest** class should live under `$HOME/Lang-53f/`

    Note: **StudentTest** must include the packages specified in the table found in section 4.1.

Given the set of dependencies of each project/*bug* version, you cannot use Mockito and you may be able to use JUnit 3 or 4.

- Cli-32 supports JUnit 3.8.2 (API in here https://javadoc.io/doc/junit/junit/3.8.2/index.html)
- Gson-15 supports JUnit 4.12 (API in here https://javadoc.io/doc/junit/junit/4.12/index.html)
- Lang-53 supports JUnit 4.11 (API in here https://javadoc.io/doc/junit/junit/4.11/index.html)

Note that in JUnit 3:

- The test class must extend `junit.framework.TestCase`.
- There is no `@Test` in each test case. Each test case must be named `testX`, where X is the actual behavior you will be testing, e.g., `testSum`.
- Some "new" assert statements are only available on JUnit 4, e.g., `assertNotEquals`. Please check the API links above for a complete list of assert statements available in each version.

Here an example of a JUnit 3 test class:

```java
import junit.framework.*;

public class StudentTest extends TestCase {

  @Override
  protected void setUp() throws Exception {
    // Similar to the @Before in JUnit 4 and @BeforeEach in JUnit 5
  }

  @Override
```

```
  protected void tearDown() throws Exception {
    // Similar to the @After in JUnit 4 and @AfterEach in JUnit 5
  }

  public void testX() {
    // ...
  }
}
```

and here the equivalent test class using JUnit 4:

```Java
import org.junit.Before;
import org.junit.After;
import org.junit.Test;
import static org.junit.Assert.*;

public class StudentTest {

  @Before
  protected void setUp() throws Exception {
    // Similar to the `setUp` in JUnit 3 and @BeforeEach in JUnit 5
  }

  @After
  protected void tearDown() throws Exception {
    // Similar to the `tearDown` in JUnit 3 and @AfterEach in JUnit 5
  }

  @Test
  public void testX() {
    // ...
  }

}
```

3. Describe the equivalence and the productiveness of equivalent mutants:
    - For each equivalent mutant, provide a justification for why it is equivalent.

- For each analyzed mutant, justify why it is productive (see definition of productive and unproductive in [1]).
- Write these justifications in your own copy of the spreadsheet available here.

4.4 Deliverables (deadline December ~~15~~ 18, 2023, 11:59:00 pm)

Each student should upload the following **two deliverables** on Moodle (M.EIC's link / MESW's link) in a zip file.

1. Your **StudentTest.java** file.
2. Answer the spreadsheet in Microsoft Excel format
   a. Make a copy of the spreadsheet available here.
   b. In your copy, fill in the answers to the questions and the table. The spreadsheet has two tabs: one for answering the questions above (**Questions**), and one to fill in a table with information about analyzed mutants (**Mutants**). See below **an example** of the Mutants tab filled in.

ID of analyzed mutant	Is the mutant equivalent?	Provide a rationale for equivalence	Is the mutant productive?	Provide a rationale for why you think it is productive or unproductive	Test inputs (if mutant is not equivalent)
1000	No	-	**Yes**	**It is productive because ...**	1, 2, 3
5000	**Yes**	**It is equivalent because...**	No	-	-
10000	No	-	No	-	-1, 2, 2
14000	**Yes**	**It is equivalent because...**	**Yes**	**It is productive because ...**	-

5. Troubleshooting

If you are working with Lang and PIT you should perform the following step to successfully run test cases:

- Go to file /$HOME/Lang-53f/src/dev_backup/DateUtilsTest.java and comment line 96 (suite.setName("DateUtils Tests")).

6. Miscellaneous (additional documentation)

- Defects4J: A Database of Existing Faults to Enable Controlled Testing Studies for Java Programs
- PIT Mutation Testing
- The Major Mutation Framework
- Judy

7. References

[1] Goran Petrovic and Marko Ivankovic and Robert Kurtz and Paul Ammann and René Just, "An Industrial Application of Mutation Testing: Lessons, Challenges, and Research Directions", in Proceedings of the 13th International Workshop on Mutation Analysis, 2018.

Note: If you have issues installing or using the `defects4jGUI.py` script, you may contact Pedro Tavares (up201406991@up.pt).