ITCS473
Software Quality Assurance and Testing (2021)
JUnit Exercise

This exercise is to be done **individually**.

Please note that you should do the lab work sheet completely by continuing at home after the lab session is over but you must submit before 23.55 of the class day. Once it is finished, please submit your solution as a .zip file on MyCourses. Name your file as ITCS473_[YOUR ID]_JUnit.zip.

Exercise 1

Consider the class `Rational` with its partial implementation below (also available on MyCourses).

```java
class Rational {
    long numerator,denominator;

    class Illegal extends Exception {
        String reason;
        Illegal (String reason) {
            this.reason = reason;
        }
    }

    Rational () {
        ...
    }

    Rational(long numerator, long denominator) throws Illegal {
        ...
    }

    // find the reduce form
    private void simplestForm() {
        long computeGCD;
        computeGCD = GCD(Math.abs(numerator), denominator);
        numerator /= computeGCD;
        denominator /= computeGCD;
    }

    // find the greatest common denominator
    private long GCD(long a, long b) {
        if (a%b ==0) return b;
        else return GCD(b,a%b);
    }

    public void add(Rational x) {
        numerator = (numerator * x.denominator) + (x.numerator * denominator);
        denominator = (denominator * x.denominator);
        simplestForm();
    }

    public void subtract(Rational x) {
        ...
    }

    public void multiply(Rational x) {
        ...
    }

    public void divide(Rational x) {
        ...
    }

    public boolean equals(Object x) {
        ...
    }

    public long compareTo(Object x) {
        ...
    }

    public String toString() {
        ...
    }
}
```

- Select any Java editor that you are familiar with (Eclipse, IntelliJ, Vim, etc.).
- Implement a `RationalTest` JUNIT test case that tests the above class **before** you create a full implementation for `Rational`. Apply the knowledge of input space partitioning that you learned in class to the test case design.
- After you have defined the test cases, create a a full implementation for `Rational`.
- Ensure that all tests pass and that your implementation of `Rational'` is complete.
- Note: To execute your JUNIT test case, you need `junit.jar` and `hamcrest.jar` in your classpath. To download and install them, please follow the instructions here: `https://github.com/junit-team/junit4/wiki/Download-and-Install`.

Exercise 2

Now use ANT to test your implementation.

Step 1

- In your project directory, create an ANT build file (build.xml as shown below and available on MyCourses) with a `compile` target that compiles all Java source files using `javac` ANT task.

```
<project>
  <target name="clean">
    <delete dir="build"/>
  </target>
  <target name="compile" depends="clean">
    <mkdir dir="build/classes"/>
    <javac srcdir="src" destdir="build/classes">
```

```
        <classpath location="lib/junit-4.13.jar" />
      </javac>
      </target>
      <target name="jar" depends="compile">
        <mkdir dir="build/jar"/>
        <jar destfile="build/jar/Rational.jar" basedir="build/classes">
        <manifest>
          <attribute name="Main-Class" value="Rational"/>
        </manifest>
        </jar>
      </target>
      <target name="run" depends="jar">
        <java jar="build/jar/Rational.jar" fork="true"/>
      </target>
    </project>
```

- Inside the compile target, add junit.jar to the classpath.

```
<javac srcdir="." destdir="classes">
  <classpath location="lib/junit-4.13.jar" />
</javac>
```

- Ensure that your target can be executed successfully.

Step 2

- Create a test target which depends on the compile target.
- Add the following junit task inside the test target.

```
<target name="test" depends="compile">
  <junit showoutput="yes" printsummary="yes" haltonfailure="no">
    <classpath location="build/classes" />
    <classpath location="lib/junit-4.13.jar" />
    <classpath location="lib/hamcrest-core-1.3.jar" />
    <test name="RationalTest" />
  </junit>
</target>
```

- Run ant test and see the results of your test cases.

Step 3

You can tell the junit task to create a report of running the test cases by adding todir attribute with a directory name and specify the report format.

- Modify the junit task as shown below.

```
<target name="test" depends="compile">
  <mkdir dir="report"/>
    <junit showoutput="yes" printsummary="yes" haltonfailure="no">
      <classpath location="build/classes" />
      <classpath location="lib/junit-4.13.jar" />
      <classpath location="lib/hamcrest-core-1.3.jar" />

      <test name="RationalTest" todir="report">
        <formatter type="plain" />
        <formatter type="xml" />
    </test>
  </junit>
</target>
```

- Modify your `junit` target to create a `report` folder before the `junit` task.
- After running `ant test`, check the `report` folder. What do you find in there?

Exercise 3

Now use MAVEN to test your implementation by following the instruction.

Step 1

Create a Maven project by using the Eclipse/IntelliJ wizard or using **mvn** command with the below parameters:

```
mvn archetype:generate -DgroupId=th.ac.mahidol.itcs473 -DartifactId=JunitTest
-DarchetypeArtifactId=maven-archetype-quickstart
```

Step 2

Open **pom.xml** file and add the JUnit dependency.

```
<dependencies>
<!-- https://mvnrepository.com/artifact/junit/junit -->
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.13</version>
        <scope>compile</scope>
    </dependency>
</dependencies>
```

Step 3

Check the default project structure with file explorer or using the `dir` (Windows) or `ls` (macOS, Ubuntu, Linux) command.

Step 4

Build the project with the command **mvn install**

Step 5

As you can see in **Step 3**, Maven automatically create **src/main** and **src/test** directories with their subdirectories. **src/test** is a directory for your test codes. The other than test codes will belong to **src/main**. Please change **AppTest.java** file to **RationalTest.java**. Then, please write test case for the **Rational** class as same as in **Exercise 1**.

Step 6

Please change **App.java** file to **Rational.java** and fill the file as same as **Exercise 1**.

Step 7

Build the project with the command **mvn install**. You can see the test result with the command. Make all your test case pass.