Application development in Java – Assignment 1

Unit Testing Framework - MyUnitTester
Version 1.0

Sofia Leksell
`sole0037@ad.umu.se, id20sll@cs.umu.se`

2021-11-18

Table of Contents

1 Introduction

This assignment is covering the ability to use Reflection API, create a GUI and implement a MVC design for coding. The MVC(Model-view-controller) design pattern can be seen in the classes that have been constructed. The Reflection API is implemented in the Model class. This allows us to get information about a class's members, constructor, methods etc without specifying directly in the code which class it is. The user will enter a name of a class and this will be examined. A given interface and a test class have been used as well as additional constructed test classes.

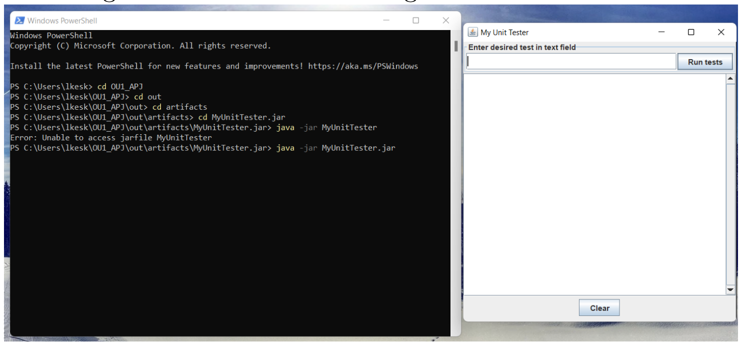
2 User manual

When the all the classes were compiled and ready, the classes were packaged into a JAR file. The JAR file was built through.. The program MyUnitTester is invoked by the command line

```
java -jar MyUnitTester.jar
```

Windows Powershell was used as a commando line interface. When the program is invoked, a window pops up where the user can write the desired test to be executed. The printed text represents the result from each of the test methods as well as a summarize of the methods invoked. The figure beneath shows how the program looks when started by the command line.

Figure 1: Run JAR file using Windows Powershell



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\lkesk> cd OU1_APJ
PS C:\Users\lkesk\OU1_APJ> cd out
PS C:\Users\lkesk\OU1_APJ\out> cd artifacts
PS C:\Users\lkesk\OU1_APJ\out\artifacts> cd MyUnitTester.jar
PS C:\Users\lkesk\OU1_APJ\out\artifacts\MyUnitTester.jar> java -jar MyUnitTester
Error: Unable to access jarfile MyUnitTester
PS C:\Users\lkesk\OU1_APJ\out\artifacts\MyUnitTester.jar> java -jar MyUnitTester.jar
```

When the user has decided what test to be run, the user can either press the enter button or click on "Run Test". To remove text from the display, the user can click on the "Clear" button. By closing the window, the program will stop running.

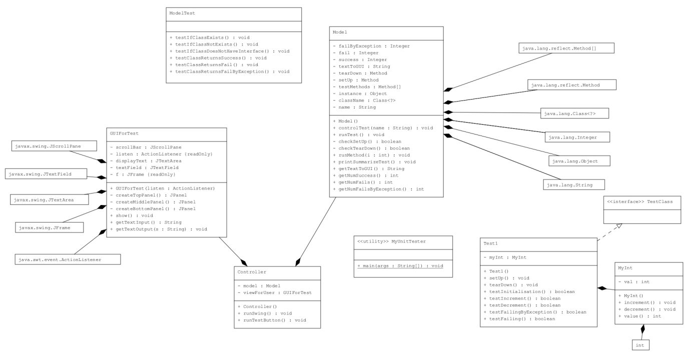
3 System description

The system is mainly constructed by the three classes forming the MVC pattern, that is Model.java, GUIForTest.java and Controller.java. These are placed in a package situated in the source directory. There is also a ModelTest class implemented to test the different methods in the class Model. Additional test classes were also constructed to ensure that correct error messages were printed out to the user. For example, Test3 is implemented without an interface and Test4 takes a parameter in one method.

The program will start by running MyUnitTester.java, were the main class is situated. A Controller object will be created and the GUI will be updated using SwingWorker. The GUI is viewed on the event dispatch thread and the background tasks executes on the worker thread(by methodes doinbackground() and done()). The worker thread will be executed when the user presses the "Run Test". An action listener has been added to the run button in GUIForTest class. This listens for an action from the Controller class. The model and GUI class are connected. The only communication between these two goes through the Controller class.

When the thread has been executed the controller will firstly call for the Model class and the method controlTest. This method controls if the test class and associated methods fulfil the requirements. The results from the tests will be saved as a string. The string will be updated in the worker thread.

Figure 2: UML Diagram representing the MVC system and test class

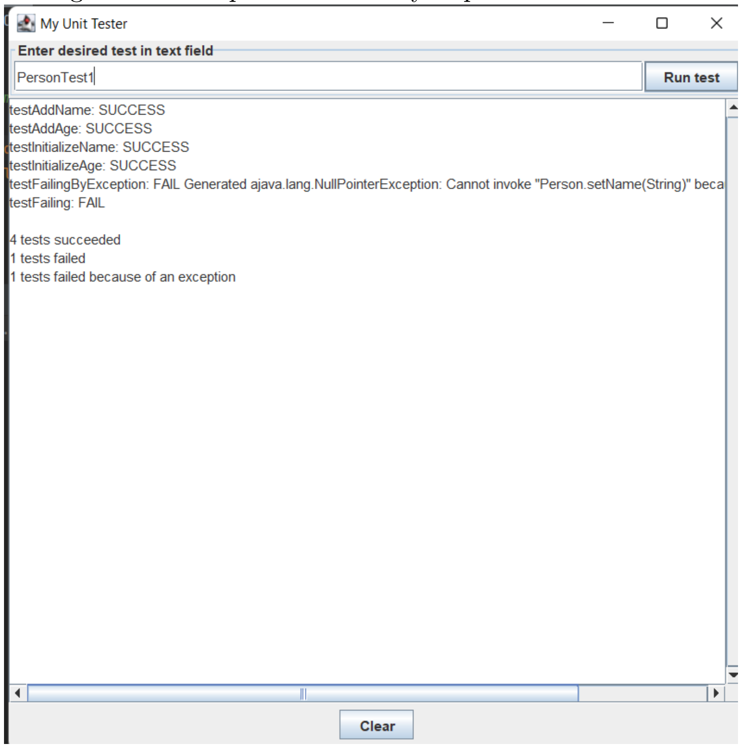

ModelTest
+ testIfClassExists() : void
+ testIfClassNotExists() : void
+ testIfClassDoesNotHaveInterface() : void
+ testClassReturnsSuccess() : void
+ testClassReturnsFail() : void
+ testClassReturnsFailByException() : void
Model
- failByException : Integer
- fail : Integer
- success : Integer
- textToGUI : String
- tearDown : Method
- setUp : Method
- testMethods : Method[]
- instance : Object
- className : Class<?>
- name : String
+ Model()
+ controlTest(name : String) : void
+ runTest() : void
- checkSetUp() : boolean
- checkTearDown() : boolean
+ runMethod(i : int) : void
+ printSummarizeTest() : void
+ getTextToGUI() : String
+ getNumSuccess() : int
+ getNumFails() : int
+ getNumFailsByException() : int
java.lang.reflect.Method[]
java.lang.reflect.Method
java.lang.Class<?>
java.lang.Integer
java.lang.Object
java.lang.String
GUIForTest
- scrollBar : JScrollPane
- listen : ActionListener {readOnly}
- displayText : JTextArea
- textField : JTextField
- f : JFrame {readOnly}
+ GUIForTest(listen : ActionListener)
- createTopPanel() : JPanel
- createMiddlePanel() : JPanel
- createBottomPanel() : JPanel
+ show() : void
+ getTextInput() : String
+ getTextOutput(s : String) : void
javax.swing.JScrollPane
javax.swing.JTextField
javax.swing.JTextArea
javax.swing.JFrame
java.awt.event.ActionListener
Controller
- model : Model
- viewForUser : GUIForTest
+ Controller()
+ runSwing() : void
+ runTestButton() : void
<<utility>> MyUnitTester
+ main(args : String[]) : void
<<interface>> TestClass
Test1
- myInt : MyInt
+ Test1()
+ setUp() : void
+ tearDown() : void
+ testInitialisation() : boolean
+ testIncrement() : boolean
+ testDecrement() : boolean
+ testFailingByException() : boolean
+ testFailing() : boolean
MyInt
- val : int
+ MyInt()
+ increment() : void
+ decrement() : void
+ value() : int
int

4 Test runs

To make sure that the program works correctly, severeal test classes has been implemented. The given class Test1 has been used and has been modified and three additional class was created. Another class, Person.java, was constructed and two associated test classes.

- Test1 - correctly implemented test class
- Test2 - methods returns false but should return true
- Test3 - does not implement interface
- Test4 - takes parameter in one method
- PersonTest1 - correctly implemented test class
- PersonTest2 - takes parameter in constructor

Figure 3: Example of a correctly implemented test class



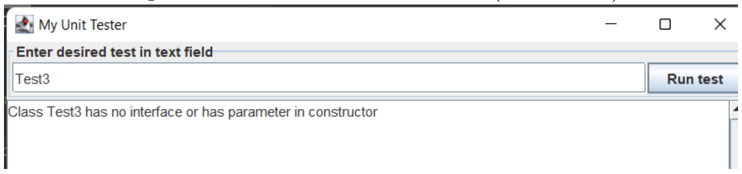
```
PersonTest1
```

```
testAddName: SUCCESS
testAddAge: SUCCESS
testInitializeName: SUCCESS
testInitializeAge: SUCCESS
testFailingByException: FAIL Generated ajava.lang.NullPointerException: Cannot invoke "Person.setName(String)" beca
testFailing: FAIL

4 tests succeeded
1 tests failed
1 tests failed because of an exception
```

If any exception is thrown, the user will be informed when running the test. The figure beneath shows how the output looks like when entering a non-correctly implemented test class.

Figure 4: Test3 has no interface(TestClass)

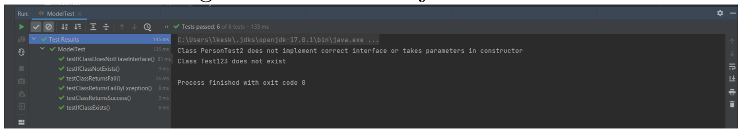


```
Test3
```

```
Class Test3 has no interface or has parameter in constructor
```

As mentioned before, a test class(A JUnit Test) regarding the Model class has been constructed. By running the test class, it implifies that the class has been correctly implemented.

Figure 5: ModelTest.java test run

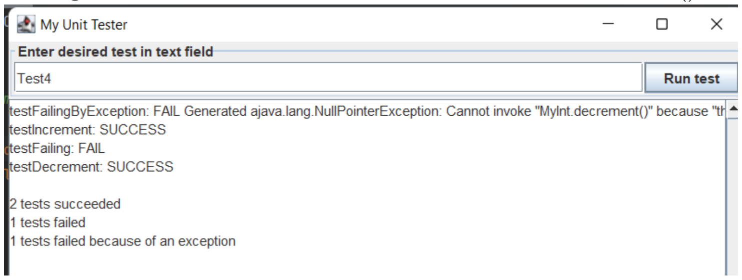

Run: ModelTest
Tests passed: 6 of 6 tests – 135 ms
Test Results 135 ms
ModelTest 135 ms
testIfClassDoesNotHaveInterface() 81 ms
testIfClassNotExists() 9 ms
testClassReturnsFail() 26 ms
testClassReturnsFailByException() 8 ms
testClassReturnsSuccess() 3 ms
testIfClassExists() 8 ms

```
C:\Users\lkesk\.jdks\openjdk-17.0.1\bin\java.exe ...
Class PersonTest2 does not implement correct interface or takes parameters in constructor
Class Test123 does not exist

Process finished with exit code 0
```

A limitation with the model class is that nothing is printed out if a test method is not correct. It just simply ignores the test method and continue on with printing out the correctly implemented test methods(see Figure 6)

Figure 6: Test4 that does not include testInitialisation()



```
Test4
```

```
testFailingByException: FAIL Generated ajava.lang.NullPointerException: Cannot invoke "MyInt.decrement()" because "th
testIncrement: SUCCESS
testFailing: FAIL
testDecrement: SUCCESS

2 tests succeeded
1 tests failed
1 tests failed because of an exception
```