PRT452

SOFTWARE ENGINEERING: PROCESS AND TOOLS

Dr Charles Yeo

M D K S Gunathilaka

321904

Semester 2

Charles Darwin University

Table of Contents

[Introduction 3](#_Toc19476384)

[Unit Test and Refactoring Questions 3](#_Toc19476385)

[Github for active project 3](#_Toc19476386)

[Test Cases out of Expected User Scenario 3](#_Toc19476387)

[Requirement 01 - Input validation 3](#_Toc19476388)

[Requirement 02 - - Input evaluation 6](#_Toc19476389)

[Requirement 03 – Display Results 8](#_Toc19476390)

[Requirement 04 – Display the number of User Guessing Attempts 11](#_Toc19476391)

[Requirement 05 – Check the Guess button will disable after the correct guess or quit game 13](#_Toc19476392)

[Requirement 06 – Check the application will close after clicking quite button twice 15](#_Toc19476393)

[Code Refactoring 17](#_Toc19476394)

[Conclusion 17](#_Toc19476395)

# Introduction

The project expected to demonstrate the knowledge in the Test Driven Development , based on the below scenario, including the code refactoring. In order to meet the expectation, android application developed with Junit and Expresso User interface testing framework.

The report consists of five sections. The first section indicates the question and the URL of the Github repository given in the next section. The remaining sections cover the test cases out of identified requirements and how the development performed. The fourth section identified the possible code refactoring within the project.

# Unit Test and Refactoring Questions

1. Write a program using TDD to do the following (20 marks):
2. This program will generate a random number between 1 to 100. User will be asked to guess the number or enter ‘q’ to quit the program. The program will keep asking the user to guess the number until the user guesses it correctly or has quitted. Once the user has guessed it correctly or quitted, the program will tell the user how many attempts have been taken.
3. Refactor your code – Code smell can give indications that there is some issue with the codes and can be solved by refactoring. Identify issues and their related code smells in your code. Include solutions to fix up the issues and implement them.
4. Create a Git directory for your assignment (including word or pdf documents and programming code)
5. Capture screenshots and write briefly the steps you have taken to create this program

# Github for active project

Project can be find in the below github URL .In order to execute the code Android Studio required. The project located inside the master branch [GuessingGame](https://github.com/kanchana-sankalpa/GussingGame/tree/master/GuessingGame) folder.

<https://github.com/kanchana-sankalpa/GussingGame>.git

The code after refactoring can be found in the “refactored” branch.

# Test Cases out of Expected User Scenario

*This program will generate a random number between 1 to 100. User will be asked to guess the number or enter ‘q’ to quit the program. The program will keep asking the user to guess the number until the user guesses it correctly or has quitted. Once the user has guessed it correctly or quitted, the program will tell the user how many attempts have been taken.*

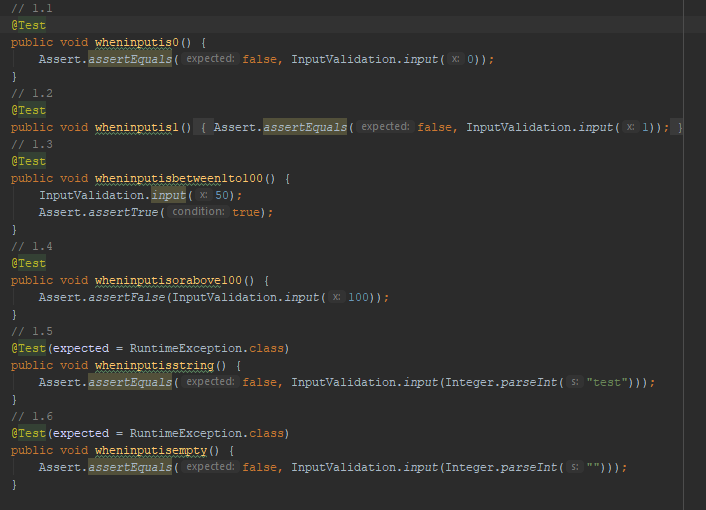
Following unit, component and system requirements can be identified from the Scenario.

## Requirement 01 - Input validation

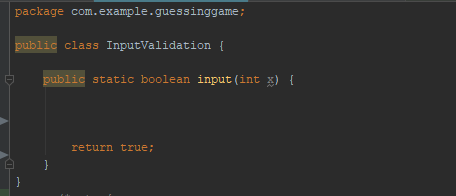
The user input validation is an indirect system requirement need to consider within the program. Following test cases developed from the requirement.

* Test cases -
  1. When input is 0
  2. When input is 1
  3. When input is between 1 to 100
  4. When input is above 100 (including 100)
  5. When input is string
  6. When input is empty

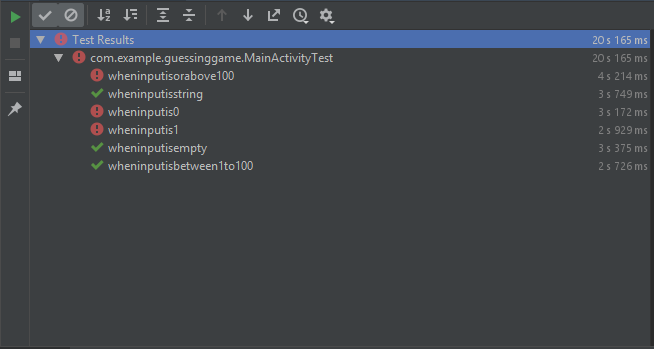
The test cases are written in src\androidTest\java\com\example\guessinggame\ MainActivityTest.java



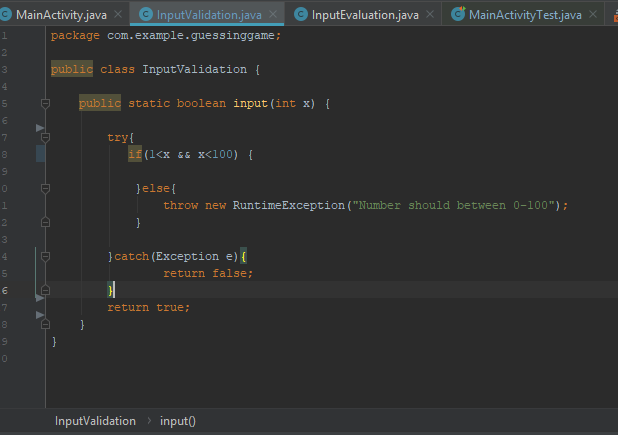
InputValidation is the Class reside the input validation code which is not written still.

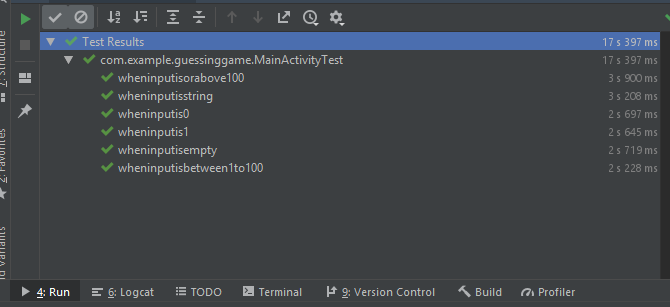


In this context, the test results look like follows.



After impelmenting the InputValidation Class, the test cases results look line follows.





## Requirement 02 - - Input evaluation

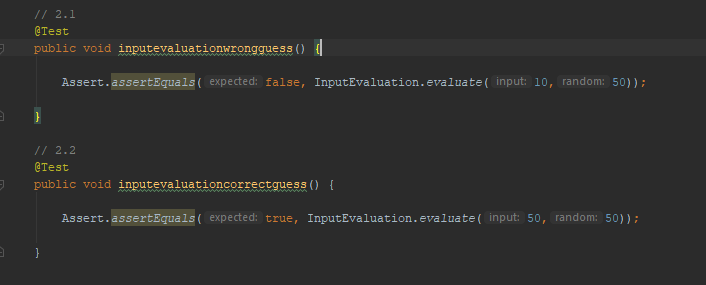
Value of the input compared against the program gassed value is the next system requirement. Input should be valid passing the requirement 1 test cases.

Test cases

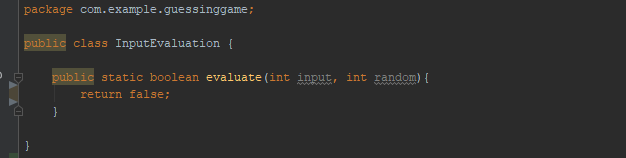
2.1 Check for wrong guess

2.2 Check for correct guess

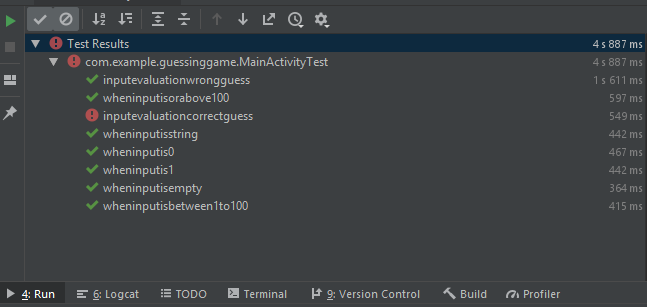
The test cases are written in src\androidTest\java\com\example\guessinggame\ MainActivityTest.java



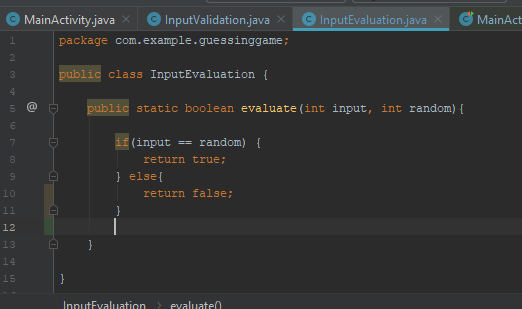
Without the implementation for input evaluation class, the code section like below (\src\main\java\com\example\guessinggame\InputEvaluation.java).

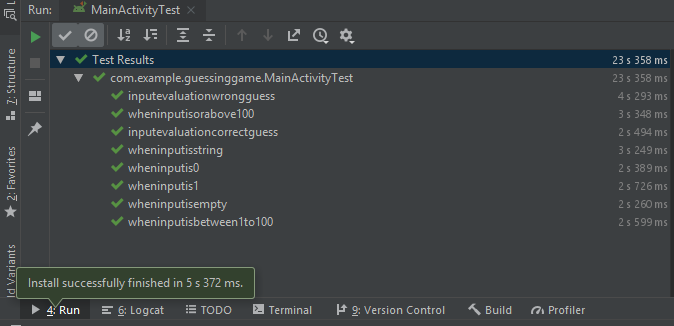


The input evaluation test results for the above code is like below.



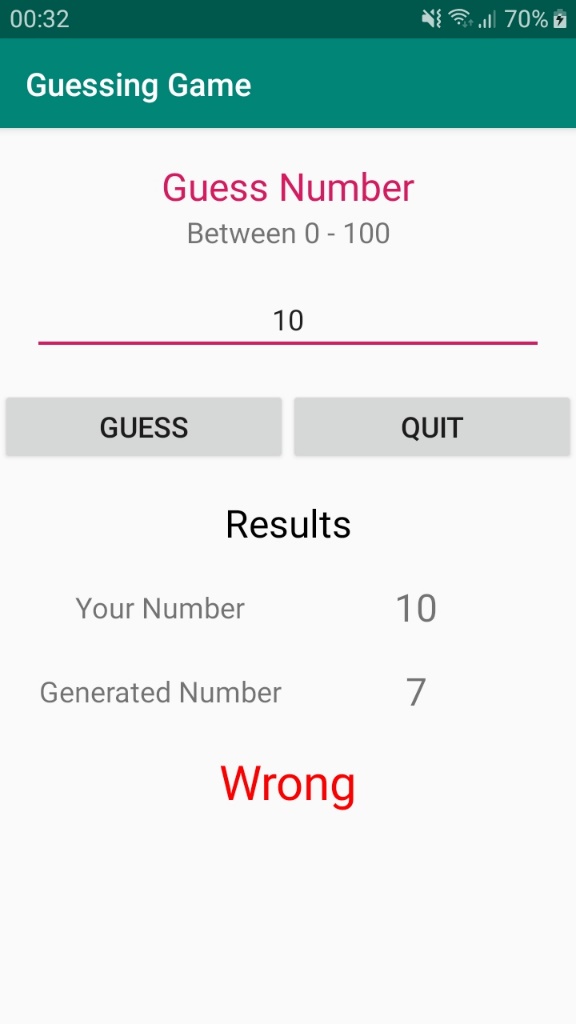
After implementing the InputEvaluation Class Code like flows the test results make all passed.





## Requirement 03 – Display Results

The application compares the number generated (number) with guessed value (guess) and display the guess in TextView as “correct” or “wrong” as Indicated below.

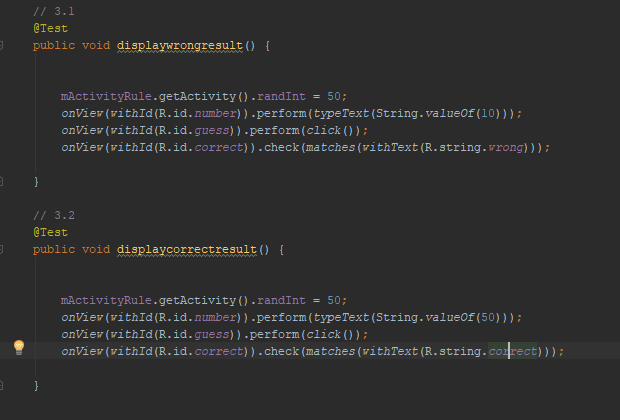


Test cases

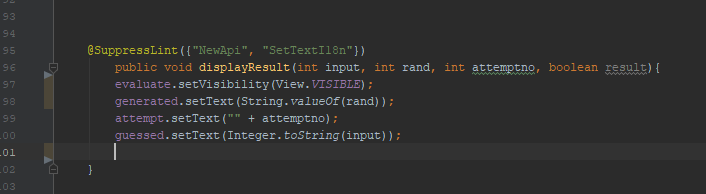
3.1 Check for wrong guess – Display “Wrong”

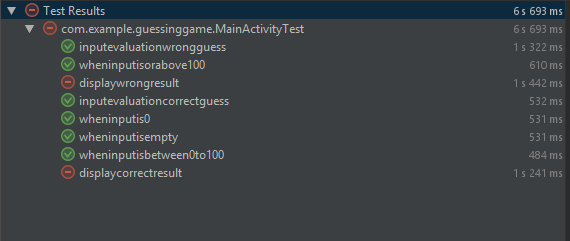
3.2 Check for correct guess – Display “Correct”

The test cases are written in src\androidTest\java\com\example\guessinggame\ MainActivityTest.java. it set the value to user input and click the “guess” button. The value of the result TextView compared weather “Wrong” or “Correct”.

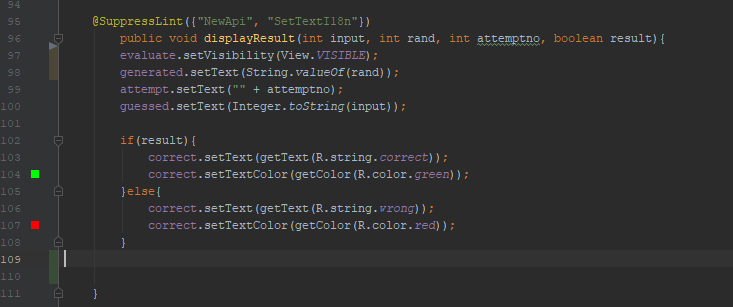


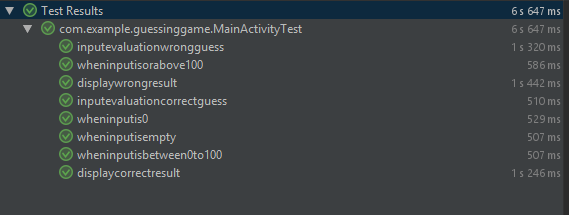
Without the implementation displayResult method, the test cases (displaywrongresult) will be failed (\src\main\java\com\example\guessinggame\ MainActivity.java).





With the implementation displayResult method, the test cases will be passed as follows.





## Requirement 04 – Display the number of User Guessing Attempts

According to the requirements, the no of attempts should display after the correct guess or after quitting the game.

Test cases

4.1 Display no of attempts – When guess is correct

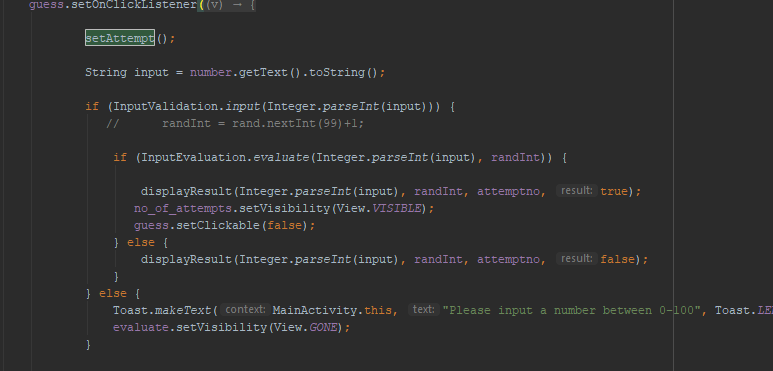
4.2 Display no of attempts – When click quit button

The test cases are written in src\androidTest\java\com\example\guessinggame\ MainActivityTest.java. The attempt count set to attempt\_no TextView field and check weather it is appropriate based on conditions.

A screenshot of a cell phone

Description automatically generated

setAttempt method called each time the guess button clicked and without implementing the setAttempt method , the test cases will be failed .This include not implementing special display code on quit button.



A screenshot of a cell phone

Description automatically generated

After implementing the setAttempt method the test cases for quit and the guess correct will be passed.

A screenshot of a cell phone

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a cell phone

Description automatically generated

## Requirement 05 – Check the Guess button will disable after the correct guess or quit game

This is an indirect requirement based on the user experience to disable the guess button after a success guess or click quite button.

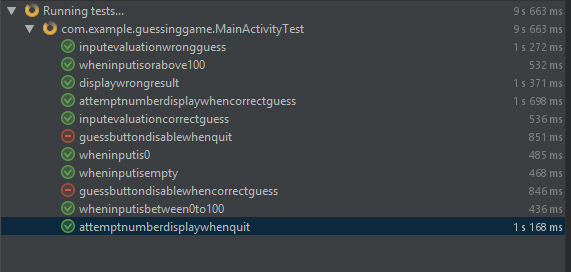
Test cases

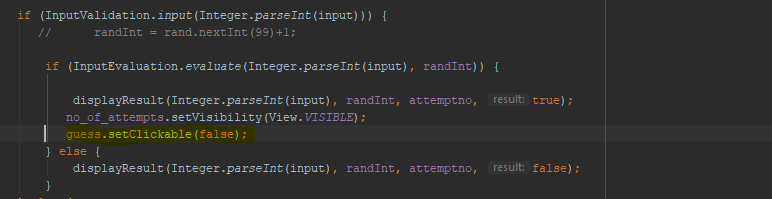
4.1 Disable guess button – when guess is correct

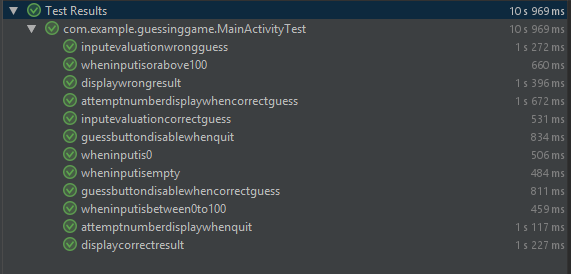
4.2 Disable guess button – when click quit button

Two test written out of the requirement in src\androidTest\java\com\example\guessinggame\ MainActivityTest.java. The code for disable button written in the MainActivity.java as highlighted below.







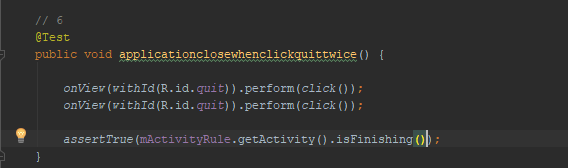


## Requirement 06 – Check the application will close after clicking quite button twice

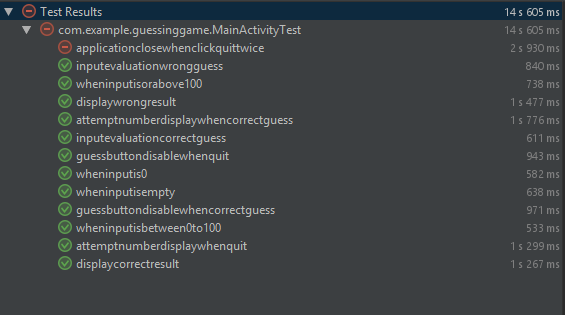
Application will close after clicking the quit button twice. If the guess is correct the user can quit or user can click quit button anytime .

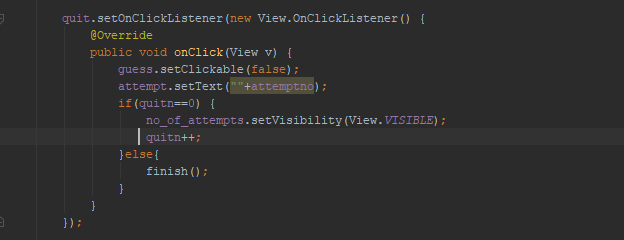
Test cases -

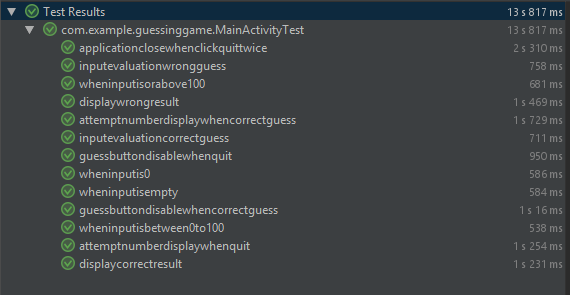
4.1 Close application – after clicked the quit button twice



Test written out of the requirement in src\androidTest\java\com\example\guessinggame\ MainActivityTest.java. The code user the “quitn” variable to keep the tarack of number of quit button clicks within the MainActivity.java







# Code Refactoring

After walking through, the project following list of improvement identified.

* Improve the test execution order. Current implementation does not force the order of the tests. However, some of the tests need to execute before other tests given in the order of the requirements. After searching the possible approach to order the test execution found based on the alphabetical order. Test cases prefixed based on the alphabetical order.
* In addition to that the naming of following variables, declaration of control improved in order to increase the readability.
  + guess >> btnGuess
  + quit >> btnQuit
  + number >> editTxtGuessedNumber
  + generated >> txtViewGeneratred
  + correct>> txtViewDecision
  + guessed>> txtViewGuessed
  + attempt>> txtViewNumberOfAttempts
  + attemptno >> numberOfAttempts
  + quitn >> numberOfQuitClicked

All the changes in refactored code can be found in refactored branch.

# Conclusion

The jUnit and Expresso can cover most of the Test Driven Development Requirement in both logic and User interfaces in the given set of requirements. However , there are limitations in handling some of the test cases such as matching the text “wrong” or “correct”, which lead to errors, without implementation in Exproesso code ,though it indicated test is failed.