# Software Testing and Maintenance

## Test Plan


# Tiny Monkey

## Software Quality Assurance Plan

BY

Krishna Yavasani

Ravi Saluru

Sandeep Saripalli

**Version: 1.0** **Date: 05-16-2015**

**Document History and Distribution**

## 1. Revision History

| Revision # | Revision Date | Description of Change | Author |
|---|---|---|---|
| 1 | 04-23-15 | Repository creation, version control and game selection | Ravi |
| 2 | 04-28-15 | Initial commit | Ravi |
| 3 | 04-28-15 | Issue tracker | Krishna |
| 4 | 04-29-15 | Game selection | Sandeep |
| 5 | | | |
| | | | |

# TABLE OF CONTENTS

# 1. INTRODUCTION

Super Tiny Monkey is a platformer and endless runner hybrid, an endless platformer, if you will. It's an endless runner, because your character is in an infinite state of movement across the level. It's a platformer because instead of running, your character will be moving from one platform to the next. You control the titular character and you must use precise timing on your input so the character can land safely on the next platform. Along the way there are golden bugs to collect, which serve as both your money and resource for executing power ups. Also, obstacles such as spikes, creatures, and poachers will try to block your path and bring you down!

## 1.1 Objectives

To test the game to fix bugs and check test case coverage. Including better Jautodocs and do refactoring to change methods names so that they are easy to read and understand

## 1.2 Testing Strategy

We have used CodePro and Metrics to select 9 classes to test. Initially we used find bugs tool to find bugs in the classes we selected and those bugs were fixed before writing test cases so that coverage should be more than 90 %. We have also decided to refactor some methods, package names because the game we selected is in French, in-order to have better understandability of code.

## 1.3 Scope

This Test Plan describes the unit tests and test cases that will be conducted on the application to check for coverage.

It is assumed that unit testing already provided thorough black box testing, extensive coverage of source code, and testing of all module interfaces.

## 1.4. Definitions and Acronyms

**Refactor**: The techniques enable programmers to restructure code so that the design of a program is clearer.

**Bug**: It is defined as an error, flaw, failure, or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.

# 2. TEST ITEMS

## 2.1 Program Modules

| Package | Class |
|---|---|
| tinymonkeys | Tinymonkey.java |
| model | Pirate.java |
| Model | Single.java |
| | |
| | |
| | |
| | |
| | |
| | |

## 2.2 User Procedures

**N/A**

# 3. FEATURES TO BE TESTED

Based on the time we have we have decided to test monkey moving directions, check if it can collect coins, out of bound exceptions if there are any.

# 4. FEATURES NOT TO BE TESTED

*N/A*

# 5. APPROACH

- Used CodePro and Metrics to select 9 classes
- Document classes using javadocs
- Use the CodePro, CheckStyle, FindBugs, PMD, ECLEmma, Randoop, JMetrics, PIT Mutation, Junit test case generator tools to discover weakness
- Fix any high level issues discovered by test tools

Following are the classes that we worked on and the description of the steps we did to improve the classes:

**Group Member:** Krishna Yavasani

**Package Name:** tinymonkey.tinymonkeys

**1. Class Name:** Tinymonkey

**Tool used**: 1. Find bugs

No bugs were found by the tool

**Tool used: 2**. Eclipse Refactoring

I have changed the name that is easy to read and understand in English

**Tool used: 3** Jautodocs

Generated docs

**Group Member:** Krishna Yavasani

**Package Name:** tinymonkey.tinymonkeys.model

**2. Class Name: Pirate.java**

**Tool used**: 1. Find bugs

One bug detected

Private int butin bug was resolved.

**Tool used 2**: eclipse refactor

I have refactored this package for better understanding.

**Tool used 3**: PMD

The complexity of some of the methods was reduced by refactoring. Again, PMD violations were removed.
Average complexity got reduced to 11.22 from 16.71

**Group Member:** Krishna Yavasani

**Package Name:** tinymonkey.tinymonkeys

**3. Class Name:** single.java

**Tool used: 1.** Find bugs
No bugs were detected

**Tool used**: 2. PMD

49 violations were found.

The number of violations was reduced to 31, after removing unnecessary parentheses, renaming the short variables and removing unnecessary returns.

Tool used: 3. Jautodoc

Used this tool to add file headers.

**Group Member:** Sandeep Saripalli

**Overall Project**

**1. Class Name:** Tinymonkey

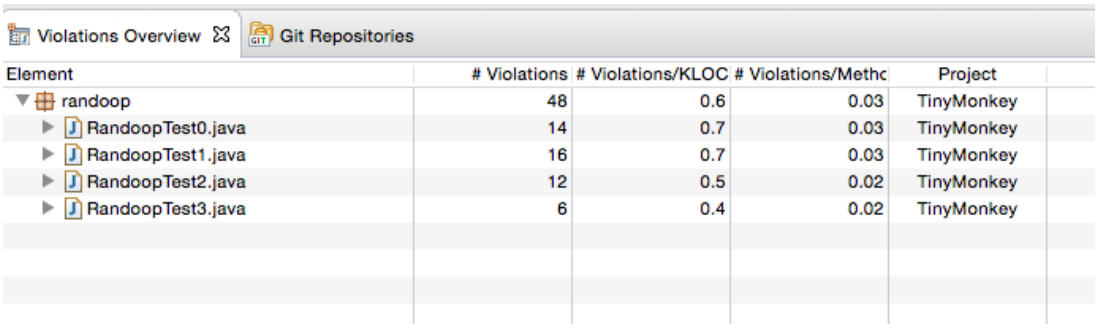
**Tool used**: 1. Find bugs

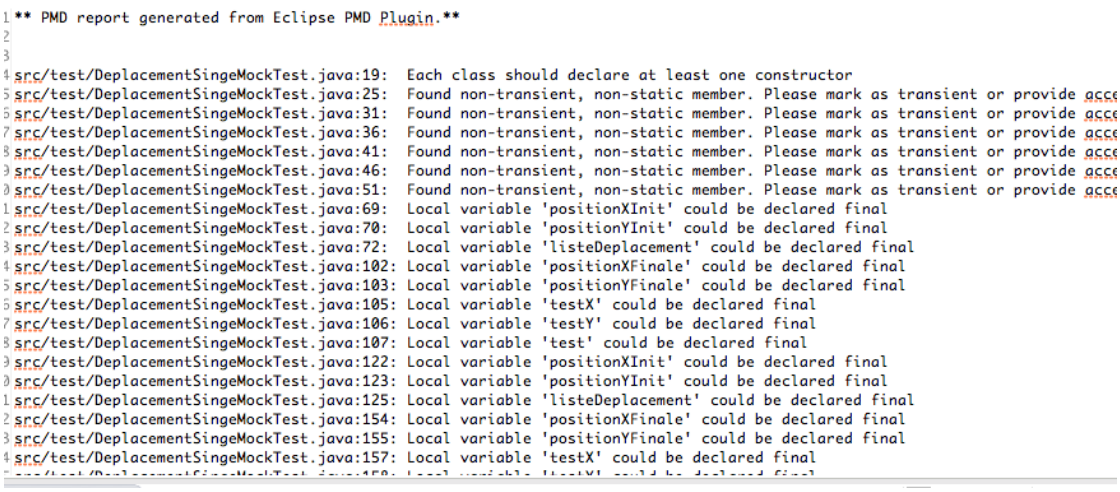
No bugs were found by the tool

```
<BugCollection version="3.0.1-dev-20150306-5afe4d1" sequence="1" timestamp="1431925913430" analysisTi
  <Project projectName="TinyMonkey">
    <Jar>/Users/sandeepsaripalli/git/software_testing_project/TinyMonkey/bin</Jar>
    <AuxClasspathEntry>/Users/sandeepsaripalli/git/software_testing_project/TinyMonkey/bin</AuxClassp
    <AuxClasspathEntry>/Library/Java/JavaVirtualMachines/jdk1.8.0_31.jdk/Contents/Home/jre/lib/rt.jar
    <SrcDir>/Users/sandeepsaripalli/git/software_testing_project/TinyMonkey/src</SrcDir>
    <SrcDir>/Users/sandeepsaripalli/git/software_testing_project/TinyMonkey/test</SrcDir>
    <Cloud id="edu.umd.cs.findbugs.cloud.doNothingCloud" online="false"></Cloud>
  </Project>

  <Errors errors="0" missingClasses="0"></Errors>
  <FindBugsSummary timestamp="Mon, 18 May 2015 01:11:53 -0400" total_classes="0" referenced_classes="
    <FindBugsProfile>
      <ClassProfile name="edu.umd.cs.findbugs.classfile.engine.ClassInfoAnalysisEngine" totalMillisec
      <ClassProfile name="de.tobject.findbugs.builder.FindBugs2Eclipse" totalMilliseconds="93" invoca
    </FindBugsProfile>
  </FindBugsSummary>

  <ClassFeatures></ClassFeatures>
  <History>
    <AppVersion sequence="0" timestamp="1431925913430" release="" codeSize="0" numClasses="0"/>
  </History>
</BugCollection>
```

**Tool used: 2**. PMD

Found 48 voilations





```
** PMD report generated from Eclipse PMD Plugin.**


src/test/DeplacementSingeMockTest.java:19:  Each class should declare at least one constructor
src/test/DeplacementSingeMockTest.java:25:  Found non-transient, non-static member. Please mark as transient or provide acce
src/test/DeplacementSingeMockTest.java:31:  Found non-transient, non-static member. Please mark as transient or provide acce
src/test/DeplacementSingeMockTest.java:36:  Found non-transient, non-static member. Please mark as transient or provide acce
src/test/DeplacementSingeMockTest.java:41:  Found non-transient, non-static member. Please mark as transient or provide acce
src/test/DeplacementSingeMockTest.java:46:  Found non-transient, non-static member. Please mark as transient or provide acce
src/test/DeplacementSingeMockTest.java:51:  Found non-transient, non-static member. Please mark as transient or provide acce
src/test/DeplacementSingeMockTest.java:69:  Local variable 'positionXInit' could be declared final
src/test/DeplacementSingeMockTest.java:70:  Local variable 'positionYInit' could be declared final
src/test/DeplacementSingeMockTest.java:72:  Local variable 'listeDeplacement' could be declared final
src/test/DeplacementSingeMockTest.java:102: Local variable 'positionXFinale' could be declared final
src/test/DeplacementSingeMockTest.java:103: Local variable 'positionYFinale' could be declared final
src/test/DeplacementSingeMockTest.java:105: Local variable 'testX' could be declared final
src/test/DeplacementSingeMockTest.java:106: Local variable 'testY' could be declared final
src/test/DeplacementSingeMockTest.java:107: Local variable 'test' could be declared final
src/test/DeplacementSingeMockTest.java:122: Local variable 'positionXInit' could be declared final
src/test/DeplacementSingeMockTest.java:123: Local variable 'positionYInit' could be declared final
src/test/DeplacementSingeMockTest.java:125: Local variable 'listeDeplacement' could be declared final
src/test/DeplacementSingeMockTest.java:154: Local variable 'positionXFinale' could be declared final
src/test/DeplacementSingeMockTest.java:155: Local variable 'positionYFinale' could be declared final
src/test/DeplacementSingeMockTest.java:157: Local variable 'testX' could be declared final
```

**Tool used: 3** Checkstyle

None Found

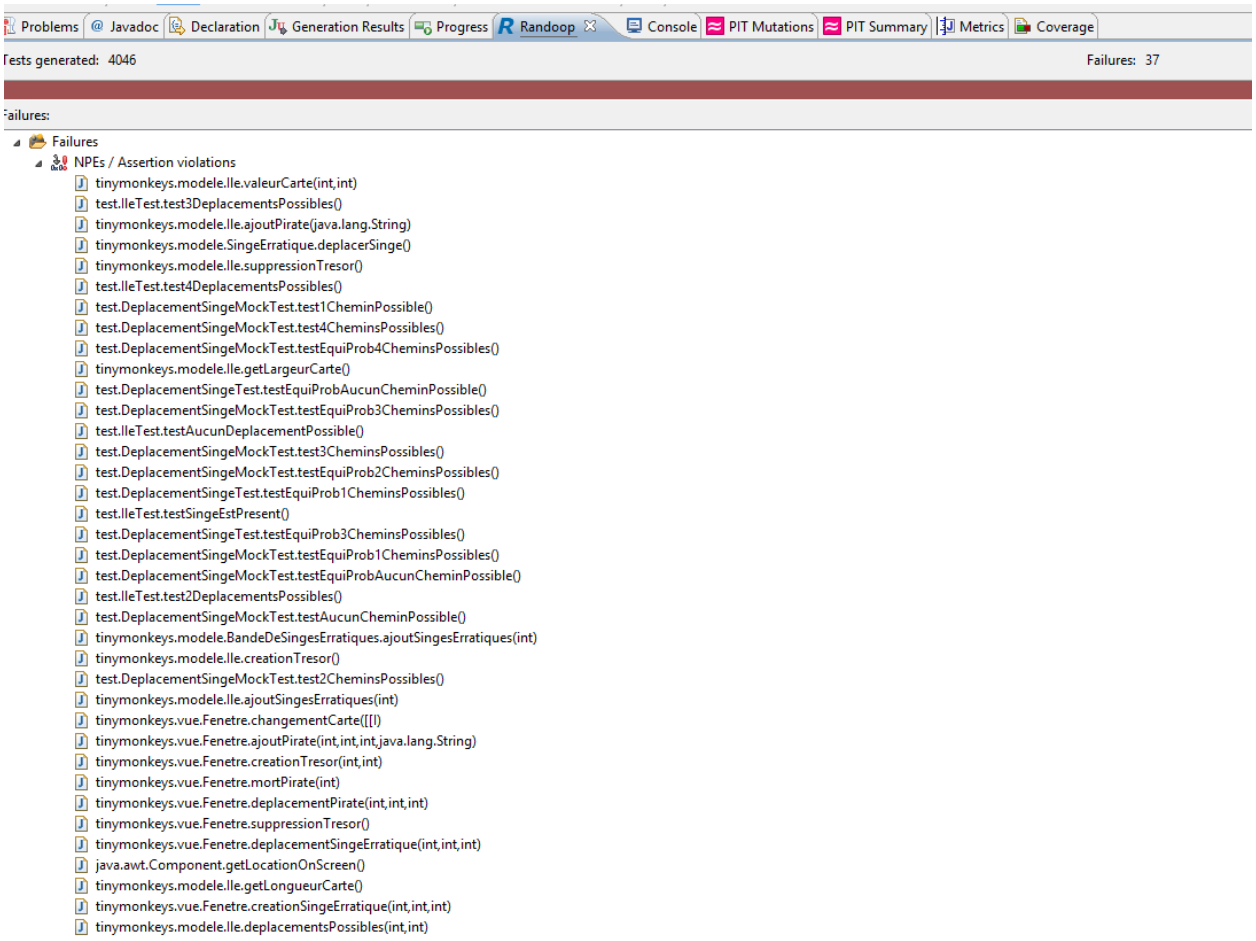
**Group Member:** Ravi Saluru

**Package Name:** Complete Project

**1. Class Name:** All classes

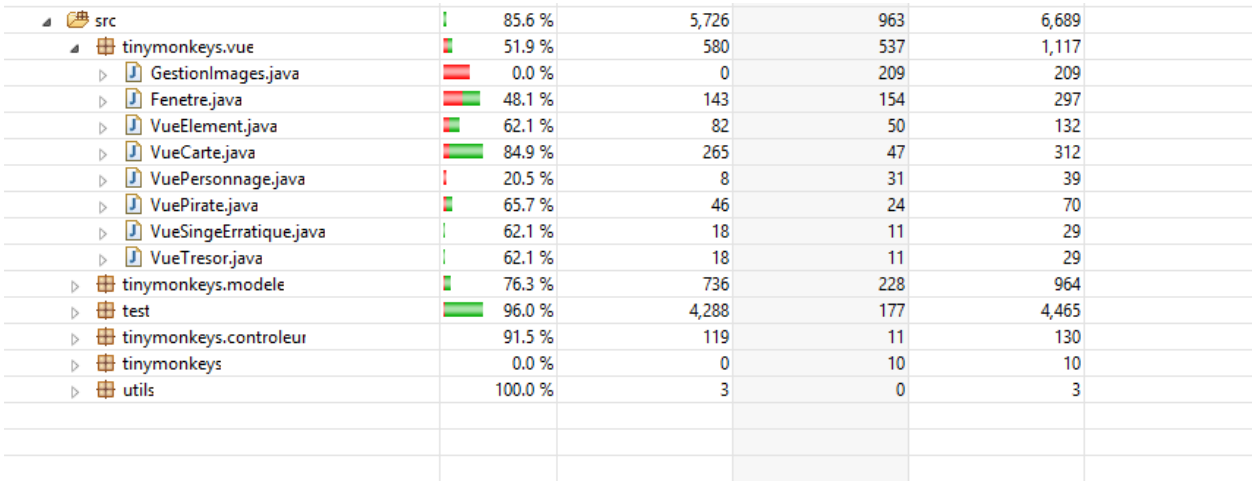
**Tool used**: 1. Randoop (From Google)

Generated around 4000 plus test cases testing the code and the GUI packages. Some Issues were found but in the test cases hence they were ignored.



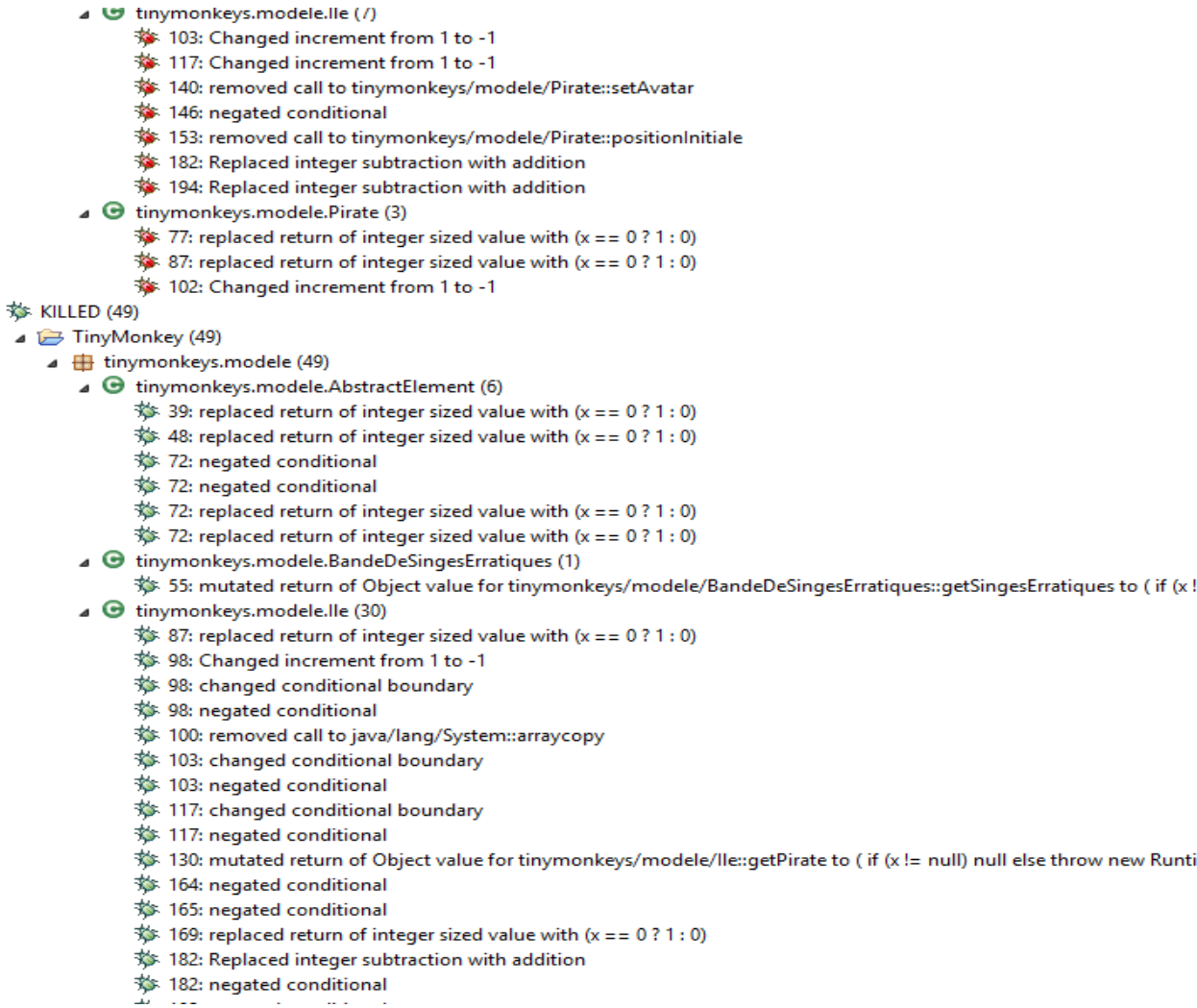

**Tool used: 2**. Coverage Criteria ECLEmma

This checked for the code coverage, test cases for the source code was at 86%.



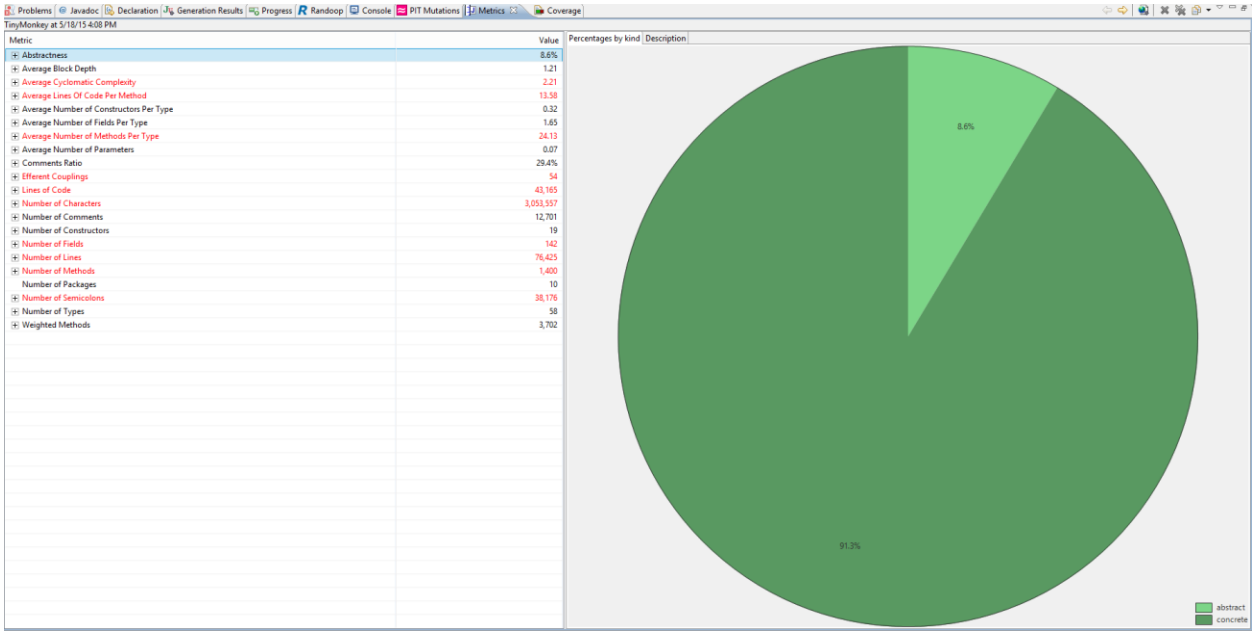

**Tool used: 3** PIT Mutations, Overall Project

Ran PIT Mutations to understand where code may go wrong.




**Tool used**: 4. CodePro – Metrics (New)

Ran metrics to check the code complexity, and analyzed for cyclomatic complexities and found the code to be within acceptable limits.




## 5.1 Component Testing

We did test each module to make sure it is working as it is supposed to be working. Later we combined each module to test how they are interacting. Finally component testing is successful. When all the components are combine, the system didn't break.

## 5.2 Integration Testing

We did run the application and it is working fine. So we assumed that it is successful.

## 5.3 Interface Testing

*N/A*

## 5.4 Security Testing

*N/A*

## 5.5 Performance Testing

We have used ***GrinderStone*** plugin from eclipse market place to perform the performance testing. It's a pretty good tool to check the performance. We have identified a couple of class where the performance can be improved. It's a pretty cool tools to use for performance testing.

## 5.6 Regression Testing

We did regression testing to make sure the system doesn't break after fixing bugs

and refactoring. All the tests ran successfully without any errors.

### 5.7 Acceptance Testing

We did ask our friends to play the game with previous version and new version. They felt there is improvement when the monkey goes to corners or edges it is not dead and still in play.

### 5.8 Beta Testing

We did check the game after all the tests are performed to make sure we have met the initial requirements of the test plan.

## 6. PASS / FAIL CRITERIA

*(Specify the criteria to be used to determine whether each item has passed or failed testing.)*

### 6.1 Suspension Criteria

When the test fails, and it cannot perform actions as it is intended to, then we considered it as fail. Later we changed the code to make it work

### 6.2 Resumption Criteria

Once the units test for the corrected code passed, then we resumed testing it and integrating with other components of system.

### 6.3 Approval Criteria

Game works fine and can perform actions as it is in requirements.

## 7. TESTING PROCESS

### 7.1 Test Deliverables

- Test Plan
- Improved code in GitHub repository
- Issue tracker
- Javadocs for the improved classes

### 7.2 Testing Tasks

- Write test cases
- Write test cases to code coverage more than 85%
- Refactor code where necessary
- Use eclipse plugins to find bugs, code complexity, mutations etc.

## 7.3 Responsibilities

| Team member | Class selected |
|---|---|
| Krishna | 1. Tinymokey.java<br>2. Pirate.java<br>3. Single.java |
| Ravi | 4. Generate Test Cases<br>5. Analyze code for performance issues, like cyclomatic complexity<br>6. Check code coverage, and Mutation testing |
| Sandeep | 7. Checkstyle coverage<br>8. Findbugs to check for common critical bugs<br>9. PMD for the dead code and other leaks in the code |

## 7.4 Resources

1. Eclipse market place to get plugins
2. sourceforge to check for some plugins
3. Game from github

## 7.5 Schedule

| Task | Date of completion |
|---|---|
| Select game and setup repository | 04/25/2015 |
| Issue tracker | 04/26/2015 |
| write test cases | 04/30/2015 |
| Pick 3 classes for testing | 05/01/2015 |
| Improvise code and use tools | 05/08/2015 |
| Document test plan | 05/18/2015 |

# 8. ENVIRONMENTAL REQUIREMENTS

- Eclipse Integrated Development Environment
- Java Development Kit (JDK )
- Java virtual environment
- Testing tools: PMD, CodePro, Find Bugs, Check Style, Metrics, Jautodoc, Randoop, JMetrics, PIT Mutation, Junit test case generator
- Dependent jars which are packaged along with the project.

# 9. CHANGE MANAGEMENT PROCEDURES

*N/A*