**System Test Plan**

**Logic Based Testing**

By: Dalton Sparks

**Table of Contents**

[**1.**](#_heading=h.30j0zll) **SYSTEM UNDER TEST 3**

[1.1.](#_heading=h.1fob9te) Details 3

[1.2.](#_heading=h.3znysh7) Software Architecture 3

[**2.**](#_heading=h.2et92p0) **TEST ENVIRONMENT 4**

[**3.**](#_heading=h.3dy6vkm) **LOGIC BASED TESTING 5**

[3.1.](#_heading=h.3rdcrjn) Method Under Test [create as many subsections as needed] 5

[**4.**](#_heading=h.4d34og8) **TEST REQUIREMENTS 6**

[4.1.](#_heading=h.2s8eyo1) CACC Coverage Criteria 6

[**5.**](#_heading=h.17dp8vu) **TEST RESULTS WITH TRACEABILITY 7**

# **System Under Test**

The project is a text based role playing game called Jadventure since it is coded in Java. Basically when you run the game it will print to the terminal a text and you have to make a reply back with what you want to do for example move, fight, run. The game will first prompt you with a menu screen where you will select to start a new game, load a new game, or delete a game. I will be testing the main menu in this project. I will be running the test within the com.jadventure.game file, specifically the JAdventure.java file.

## ***Details***

Source of the project under test:

<https://github.com/Progether/JAdventure.git>

Total Project Details:

* 5,930 lines of code total
* 62 class files
* 9 packages

File Under Test Details:

* 1 package (com.jadventure.game/JAdventure.java)
* 1 classes
* 94 lines of code

## ***Software Architecture***

The structure of the Jadventure game is based on a coordinate map that tracks the location of the player within the world as well as non player characters and items. The game runs on a client/server on the machine it is played on so you can save and load any game state. The first file called com.jadventure.game includes all the code that initially starts the text based game. The conversations file handles all the text conversations between the player and the non player characters. The entities file handles all the stats of the player and non player characters. The menus file handles the structure of all the menus within the game. Monsters file handles all of the monsters including the stats, health, and varieties of different monsters. Navigation file keeps track of the location of the player as they traverse through the virtual world. prompts file handles all of the commands the user can use to play the game starting with h or help which will display a menu of all the prompts that can be used. Then the last part of the software is the repository file which keeps track of the items the player and non player characters handle within the game which include food, armor, and weapons. That basically sums up the structure of the project Jadventure. I will be running several of my own tests on the initial start up menu where the player selects the game mode type.

# **Test Environment**

The code that is being tested is located at src/main/java/com.jadventure.game/JAdventure.java the environment and versions that are used to run the test are listed below. I will include the entire project in a file called JAdventure which will contain all of the tests and can be run in eclipse. I will also include the test files separately just in case.

Environment and Versions:

* Windows 10 version - 21H2
* Eclipse Version - 4.24.0
* Java Version - 17.0.4
* Maven Version - 3.8.4
* Junit Version - 5

# **Logic Based Testing**

## ***Method Under Test*** Main

Predicate [P1]: if (mode == GameModeType.SERVER)

Clauses:

* C1 = mode == GameModeType.SERVER

Reachability: r(p1) = True

Predicate Table (PT) [Unique ID]: P1

| Row ID | C1 | P1 | PC1 |
| --- | --- | --- | --- |
| 1 | T | T | T |
| 2 | F | F | F |

Predicate [P2]: if (mode == GameModeType.CLIENT)

Clauses:

* C2 = (mode == GameModeType.CLIENT)

Reachability: r(p2) = True

Predicate Table (PT) [Unique ID]: P2

| Row ID | C2 | P2 | PC2 |
| --- | --- | --- | --- |
| 1 | T | T | T |
| 2 | F | F | F |

Predicate [P3]: if (GameModeType.CLIENT == mode)

Clauses:

* C3 = (GameModeType.CLIENT == mode)

Reachability: r(p3) = True

Predicate Table (PT) [Unique ID]: P3

| Row ID | C3 | P3 | PC3 |
| --- | --- | --- | --- |
| 1 | T | T | T |
| 2 | F | F | F |

Predicate [P4]: if (GameModeType.SERVER == mode))

Clauses:

* C4 = (GameModeType.SERVER == mode)

Reachability: r(p4) = True

Predicate Table (PT) [Unique ID]: P4

| Row ID | C4 | P4 | PC4 |
| --- | --- | --- | --- |
| 1 | T | T | T |
| 2 | F | F | F |

## Method Under Test getGameMode

Predicate [P1]: if (args == null || args.length == 0 || "".equals(args[0].trim()))

Clauses:

* C1 = args == null
* C2 = args.length == 0
* C3 = "".equals(args[0].trim())

Reachability: r(P1) = True

Predicate Table (PT) [P1]:

| Row ID | C1 | C2 | C3 | P1 | PC1 | PC2 | PC3 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | T | T | T | T | T | T | T |
| 2 | T | T | F | T | T | T | F |
| 3 | T | F | T | T | T | F | T |
| 4 | F | T | T | T | F | T | T |
| 5 | T | F | F | T | T | F | F |
| 6 | F | T | F | T | F | T | F |
| 7 | F | F | T | T | F | F | T |
| 8 | F | F | F | F | F | F | F |

## Method Under Test toString

Predicate [P1]: if (args.length == 0)

Clauses:

* C1 = (args.length == 0)

Reachability: r(P1) = True

Predicate Table (PT) [P1]:

| Row ID | C1 | P1 | PC1 |
| --- | --- | --- | --- |
| 1 | T | T | T |
| 2 | F | F | F |

Predicate [P2]: for (int index = 0; index < args.length; index++)

Clauses:

* C1 = index < args

Reachability: r(P2) = true

| Row ID | C2 | P2 | PC2 |
| --- | --- | --- | --- |
| 1 | T | T | T |
| 2 | F | F | F |

Predicate [P3]: if (index > 0)

Clauses:

* C1 = (index > 0)

Reachability: r(P3) = True

| Row ID | C3 | P3 | PC3 |
| --- | --- | --- | --- |
| 1 | T | T | T |
| 2 | F | F | F |

# **Test Requirements**

## Main ***CACC Coverage Criteria***

MUT: Main

| **TR** | **PT ID** | **Row** | **Coverage** | | **Feasible?** | |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | P1 | 1 | PC1 where P1 = True | | Y | |
| 2 | P1 | 2 | PC1 where P1 = False | | Y | |
| 3 | P2 | 1 | PC2 where P2 = True | | Y | |
| 4 | P2 | 2 | PC2 where P2 = False | | Y | |
| 5 | P3 | 1 | PC3 where P3 = True | | Y | |
| 6 | P3 | 2 | PC3 where P3 = False | | Y | |
| 7 | P4 | 1 | PC4 where P4 = True | | Y | |
| 8 | P4 | 2 | PC4 where P4 = False | | Y | |

## 4.2. getGameMode CACC Coverage Criteria

MUT: getGameMode

| **TR** | **PT ID** | **Row** | **Coverage** | | **Feasible?** | |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | P1 | 1 | PC1,2,3 where P1 = True | | Y | |
| 2 | P1 | 2 | PC1,2 where P1 = True | | Y | |
| 3 | P1 | 3 | PC1,3 where P1 = True | | Y | |
| 4 | P1 | 4 | PC2,3 where P1 = True | | Y | |
| 5 | P1 | 5 | PC1 where P1 = True | | Y | |
| 6 | P1 | 6 | PC2 where P1 = True | | Y | |
| 7 | P1 | 7 | PC3 where P1 = True | | Y | |
| 8 | P1 | 8 | PC1,2,3 where P1 = False | | Y | |

## 4.3. ToString CACC Coverage Criteria

MUT: ToString

| **TR** | **PT ID** | **Row** | **Coverage** | | **Feasible?** | |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | P1 | 1 | PC1 where P1 = True | | Y | |
| 2 | P1 | 2 | PC1 where P1 = False | | Y | |
| 3 | P2 | 1 | PC2 where P2 = True | | Y | |
| 4 | P2 | 2 | PC2 where P2 = False | | Y | |
| 5 | P3 | 1 | PC3 where P3 = True | | Y | |
| 6 | P3 | 2 | PC3 where P3 = False | | Y | |

**Test Results with Traceability**

Method Under Test: Main

| **Test ID** | **Targeted TR** | **MUT** | **Input** | **Observed Output** | **Result** |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | Main | Sever | runs server | Pass |
| 2 | 2 | Main | Client | creates client | Pass |
| 3 | 3 | Main | Client | creates client | Pass |
| 4 | 4 | Main | Server | creates server | Pass |
| 5 | 5 | Main | Client | creates client | Pass |
| 6 | 6 | Main | Server | creates server | Pass |
| 7 | 7 | Main | Server | creates server | Pass |
| 8 | 8 | Main | Client | creates client | Pass |

Method Under Test: getGameMode

| **Test ID** | **Targeted TR** | **MUT** | **Input** | **Observed Output** | **Result** |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | getGameMode | null | Stand Alone | Pass |
| 2 | 2 | getGameMode | “” | Stand Alone | Pass |
| 3 | 3 | getGameMode | “ “ | Stand Alone | Pass |
| 4 | 4 | getGameMode | 1 | Stand Alone | Pass |
| 5 | 5 | getGameMode | 2 | Stand Alone | Pass |
| 6 | 6 | getGameMode | 3 | Stand Alone | Pass |
| 7 | 7 | getGameMode | 11 | Stand Alone | Pass |
| 8 | 8 | getGameMode | 12 | Stand Alone | Pass |

Method Under Test: toString

| **Test ID** | **Targeted TR** | **MUT** | **Input** | **Observed Output** | **Result** |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | toString | 0 | Stand Alone | Pass |
| 2 | 2 | toString | 1 | Stand Alone | Pass |
| 3 | 3 | toString | 1 | Stand Alone | Pass |
| 4 | 4 | toString | 0 | Stand Alone | Pass |
| 5 | 5 | toString | 1 | Stand Alone | Pass |
| 6 | 6 | toString | 0 | Stand Alone | Pass |