

**CS 1632 Software Quality Assurance**

**Deliverable 1**

Member 1 Name: Maxwell Benko

Member 2 Name: Zohaib Wasim

1. **Introduction**

For this project, testing is slight difficult. Many of the requirements are contingent on one feature which requires only testing that feature. However, some requirements, like FUN-LOSE, FUN-INPUT-CAPS will require testing of multiple user inputs and consequently will require multiple test cases to fully resolve them. In this case, we tested all the user inputs in a single test case to be more efficient. For requirements like FUN-LOSE, we tested the lose condition by making the user drink without any of the requirements. We believe this was sufficient to establish if this requirement was properly met. Also, for FUN-MOVE requirement, we tested the edge case of going to the furthest room and trying to move North. We also tested whether directions not listed on the print resulted in the message telling the user it cannot go in that direction. We believe these are sufficient as the user is most likely to travel in the North, South direction and rarely will try to go in an unspecified direction. Thus this edge case was important to test.

Also, many of the requirements require previous steps to be accomplished. This means that the preconditions for many test cases in this deliverable will be lengthy and repetitive.

1. **Traceability Matrix**

**FUN-ITERATION** - SIX\_COMMANDS\_TEST,

**FUN-UNKNOWN** - COMMAND – UNKNOWN\_COMMAND\_MESSAGE,

**FUN-INPUT- CAPS** - IGNORE\_CASE\_TEST,

**FUN-MOVE** - DOOR\_OR\_NO\_DOOR,

**FUN-WIN** - GOOD\_ENDING\_ACHIEVED,

**FUN-LOSE** - BAD\_ENDING\_ACHIEVED,

**FUN-INVENTORY** - INVENTORY\_CONTENTS\_MESSAGE,

**FUN-LOOK** - PLAYER\_LOOK\_AND\_COLLECT,

**FUN-HELP** - PLAYER\_NEEDS\_HELP,

**FUN-UNIQ-ROOM** - ROOM\_DESCRIPTION\_TEST,

**FUN-UNIQ-ROOM-FURNISHING** - LONE\_FURNISHING\_TEST

1. **Test Cases**

**FUN-ITERATION**:

Identifier: SIX\_COMMANDS\_TEST

Test Case: Testing to see if at each iteration of the game, the user is be able enter one of six commands - "N" to go North, "S" to go South, "L" to Look for items, "I" for Inventory, "H" for Help, or "D" to Drink.

Preconditions: Java 8 is installed and is in the OS Path environment variable. The game has started. Steps:

1. Launch program by: "java -jar coffeemaker.jar"

Execution Steps:

1. Type “N” then press enter.
2. Type “S” then press enter.
3. Type “L” then press enter.
4. Type “I” then press enter.
5. Type “H” then press enter.
6. Type “D” then press enter.

Postconditions: After each press of enter the player is greeted with a message and a new iteration of the game begins unless they win or lose in which the system exits.

**FUN-UNKNOWN**:

Identifier: COMMAND – UNKNOWN\_COMMAND\_MESSAGE

Test Case: Testing to see if a player enters a command not specified by FUN-ITERATION, and the system responds with the phrase "What?".

Preconditions: Java 8 is installed and is in the OS Path environment variable. The game has started. Steps:

1. Launch program by: "java -jar coffeemaker.jar"

Execution Steps:

1. Type “Z” then press enter.

Postconditions: The “What?” message is printed to the console and a new iteration of the game starts.

**FUN-INPUT- CAPS**:

Identifier: IGNORE\_CASE\_TEST

Test Case: Testing to see if the system is case-insensitive in regard to input values; that is, "n", "s", "l", "i", "h", "d" is equivalent to their uppercase counterparts "N", "S", "L", "I", "H", "D" respectively.

Preconditions: Java 8 is installed and is in the OS Path environment variable. The game has started. Steps:

1. Launch program by: "java -jar coffeemaker.jar"

Execution Steps:

1. Type “N” then press enter.
2. Type “S” then press enter.
3. Type “L” then press enter.
4. Type “I” then press enter.
5. Type “H” then press enter.
6. Type “D” then press enter.
7. Launch program by: "java -jar coffeemaker.jar"
8. Type “n” then press enter.
9. Type “s” then press enter.
10. Type “l” then press enter.
11. Type “i” then press enter.
12. Type “h” then press enter.
13. Type “d” then press enter.

Postconditions: Each command is recognized by the game and the “What?” message never prints.

**FUN-MOVE**:

Identifier: DOOR\_OR\_NO\_DOOR

Test Case: Testing to see if the system allows a player to move North only if a door exists going North, and South only if a door exists going South. If a door does not exist, the move shall be disallowed with the message: "A door in that direction does not exist."

Preconditions: Java 8 is installed and is in the OS Path environment variable. The game has started. Steps:

1. Launch program by: "java -jar coffeemaker.jar"
2. Type “N” then press Enter. Complete this step a total of five times in a row.

Execution Steps:

1.Type “N” then press Enter.

Postconditions:

The user is greeted with the Message "A door in that direction does not exist." and a new iteration of the game will begin.

**FUN-WIN**:

Identifier: GOOD\_ENDING\_ACHIEVED

Test Case: Testing to see if the player wins the game if and only if Coffee, Sugar, and Cream have been collected by the player and then drunk.

Preconditions: Java 8 is installed and is in the OS Path environment variable. The game has started. Steps:

1. Launch program by: "java -jar coffeemaker.jar"
2. Type “L” then press enter.
3. Type “N” then press enter.
4. Type “L” then press enter.
5. Type “N” then press enter.
6. Type “L” then press enter.
7. Type “N” then press enter.
8. Type “L” then press enter.
9. Type “N” then press enter.
10. Type “L” then press enter.
11. Type “N” then press enter.
12. Type “L” then press enter.

Execution Steps:

1. Type “D” then press enter.

Postconditions: The message “You drink the beverage and are ready to study!

You win!” is printed to the screen and the system exits with error code 0.

**FUN-LOSE**:

Identifier: BAD\_ENDING\_ACHIEVED

Test Case: Testing to see if the player loses the game if and only if the player drinks but has not collected all of the items (Coffee, Sugar, and Cream).

Preconditions: Java 8 is installed and is in the OS Path environment variable. The game has started. Steps:

1. Launch program by: "java -jar coffeemaker.jar"

Execution Steps:

1. Type “D” then press enter.

Postconditions: The player is notified of their loss and the system exits with error code 1.

**FUN-INVENTORY**:

Identifier: INVENTORY\_CONTENTS\_MESSAGE

Test Case: Testing to see if upon entering "I" for inventory, the player is informed of the items that he/she has collected (consisting of Coffee, Sugar, and Cream).

Preconditions: Java 8 is installed and is in the OS Path environment variable. The game has started. Steps:

1. Launch program by: "java -jar coffeemaker.jar"

Execution Steps:

1. Type “I” then press enter.

Postconditions: A message prints to the console consisting of the player’s inventory.

**FUN-LOOK**:

Identifier: PLAYER\_LOOK\_AND\_COLLECT

Test Case: Testing to see if upon entering "L" for Look, the player collects any items in the room and those items are added to the player's inventory.

Preconditions: Java 8 is installed and is in the OS Path environment variable. The game has started. Steps:

1. Launch program by: "java -jar coffeemaker.jar"

Execution Steps:

1. Type “L” then press enter.
2. Type “I” then press enter.

Postconditions: If an item was in the room it will be shown in the message on the screen. If there was no item in the room it will not be shown in the message on the screen

**FUN-HELP**:

Identifier: PLAYER\_NEEDS\_HELP

Test Case: Testing to see if upon entering "H" for Help, the player is shown a listing of possible commands and what their effects are.

Preconditions: Java 8 is installed and is in the OS Path environment variable. The game has started. Steps:

1. Launch program by: "java -jar coffeemaker.jar"

Execution Steps:

1. Type “H” then press enter.

Postconditions: The console displays a message that contains information on possible commands the player can make.

**FUN-UNIQ-ROOM**:

Identifier: ROOM\_DESCRIPTION\_TEST

Test Case: Testing to see if each room in the house a unique adjective describing it.

Preconditions: Java 8 is installed and is in the OS Path environment variable. The game has started. Steps:

1. Launch program by: "java -jar coffeemaker.jar"

Execution Steps:

1. Type “N” then press enter. Complete this step a total of five times in a row.

Postconditions: There are 6 unique adjectives for each room shown in the history of the console’s text.

**FUN-UNIQ-ROOM-FURNISHING**:

Identifier: LONE\_FURNISHING\_TEST

Test Case: Testing to see if each room in the house has one and only one unique furnishing visible to the user upon entering the room.

Preconditions: Java 8 is installed and is in the OS Path environment variable. The game has started. Steps:

1. Launch program by: "java -jar coffeemaker.jar"

Execution Steps:

1. Type “N” then press enter. Complete this step a total of five times in a row.

Postconditions: There are 6 unique pieces of furniture shown in the history of the console’s text.

1. **Defects** Write your defects here.

IDENTIFIER: **HELPLESS\_PLAYER\_ERROR**

SUMMARY: It is not possible for a player to receive any help if they request it.

DESCRIPTION: According to FUN-HELP “h” or “H” is supposed to be a recognized command that players can type in to get a message that tells them valid commands and what they do. However, typing in these commands is greeted with the response “What?”

REPRODUCTION STEPS:

1. Launch program by: "java -jar coffeemaker.jar"
2. Type “H” and press enter.

EXPECTED BEHAVIOR: A message that tells the player all the commands and their effects is supposed to print to the screen.

OBSERVED BEHAVIOR: “What?” is printed to the screen.

IDENTIFIER: **CAPS\_SENSITVE\_GAME\_N**

SUMMARY: Using “n” as the first command is not equivalent to using “N” as the first command

DESCRIPTION: According to FUN-INPUT-CAPS there should be no difference between using lower-case and upper-case versions of all known commands. In this case using “n” to move north does not work like using “N” should.

REPRODUCTION STEPS:

1. Launch program by: "java -jar coffeemaker.jar"
2. Type “H” and press enter.

EXPECTED BEHAVIOR: The following should be printed to the Screen.

You see a Funny room.

It has a Sad record player.

A Beige door leads North.

A Massive door leads South.

OBSERVED BEHAVIOR: “What?” is printed to the screen.

IDENTIFIER: **MAGICAL\_LAND\_NOT\_ALLOWED**

SUMMARY: Upon going north where no door exists the player enters a magical land.

DESCRIPTION: According to FUN-MOVE if a door does not exist in a certain direction a player is not allowed to move in that direction. However, in the Rough room where there is no northern door, if the player moves north they enter a “magical land” and return to the first room as if my magic.

REPRODUCTION STEPS:

1. Launch program by: "java -jar coffeemaker.jar"
2. Type “N” and press enter. Do this for a total of six times.

EXPECTED BEHAVIOR: The last northern move is not allowed.

OBSERVED BEHAVIOR: “You are in a magical land! But you are returned to the beginning!” is printed to the screen.