

**CS 1632 Software Quality Assurance**

**Deliverable 1**

Member 1 Name: Myles Conlon

Member 2 Name: Joshua Trant

1. **Introduction**

Write intro after testing.

1. **Traceability Matrix**

**FUN-ITERATION:** COMMAND-ANYTIME-TEST, ALL-COMMANDS-TEST

**FUN-UNKNOWN-COMMAND:** UNKNOWN-COMMAND-TEST

**FUN-INPUT-CAPS:** CAPS-TEST, LOWER-CASE-TEST

**FUN-MOVE:** NORTH-BOUND-TEST, SOUTH-BOUND-TEST

**FUN-WIN:** DRINK-ALL-TEST, DRINK-LESS-TEST

**FUN-LOSE:** DRINK-NONE-TEST, DRINK-ONE-TEST

**FUN-INVENTORY:** CHECK-INVENTORY-TEST

**FUN-LOOK:** LOOK-AND-GRAB-TEST

**FUN-HELP:** ENTER-HELP-TEST

**FUN-UNIQ-ROOM:** UNIQUE-ROOM-TEST

**FUN-UNIQ-ROOM-FURNISHING:** ONE-UNIQUE-TEST, UNIQUE-FURNISHING-TEST

1. **Test Cases**

IDENTIFIER: COMMAND-ANYTIME-TEST

TEST CASE: Ensure that each command works at any iteration of the game

PRECONDITIONS: Can run coffeemaker.jar and can get to each iteration.

EXECUTION STEPS:

1. Run the program
2. Enter each command
3. Observe results
4. Reload game and move to new instance.
5. Go back to step 2 until all instances have been tested.

POSTCONDITIONS: Executes each command correctly

IDENTIFIER: ALL-COMMANDS-START-TEST

TEST CASE: Ensure each command works as intended at the start of the game

PRECONDITIONS: Can run coffeemaker.jar

EXECUTION STEPS:

1. Run the program
2. Enter the 6 commands
3. Observe results

POSTCONDITIONS: Executes each command correctly

IDENTIFIER: UNKNOWN-COMMAND-TEST

TEST CASE: Ensure that no unknown commands can be entered. All responses should be “What”

PRECONDITIONS: Can run coffeemaker.jar

EXECUTION STEPS:

1. Run the program
2. Enter incorrect characters for commands
3. Enter strings for commands
4. Observe behavior.

POSTCONDITIONS: Prints “What?” for each command.

IDENTIFIER: CAPS-TEST

TEST CASE: Ensure all commands can be entered in upper-case

PRECONDITIONS: Can run coffeemaker.jar

EXECUTION STEPS:

1. Run the program
2. Enter one of the six commands in upper case (N, S, L, I, H, D)
3. Observe the result

POSTCONDITIONS: Executes the command entered

IDENTIFIER: LOWER-CASE-TEST

TEST CASE: Ensure all commands can be entered in lower-case

PRECONDITIONS: Can run coffeemaker.jar

EXECUTION STEPS:

1. Run the program
2. Enter one of the six commands in lower case (n, s, l, i, h, d)
3. Observe the result

POSTCONDITIONS: Executes the command entered

IDENTIFIER: NORTH-BOUND-TEST

TEST CASE: Ensure that the user cannot go through a north door that does not exist

PRECONDITIONS: Can run coffeemaker.jar, Navigate North until there is no more door leading that leads to the North

EXECUTION STEPS:

1. Enter “N”
2. Observe Results

POSTCONDITIONS: The move was not allowed and “A door in that direction does not exist” is displayed

IDENTIFIER: SOUTH-BOUND-TEST

TEST CASE: Ensure that the user cannot go through a south door that does not exist

PRECONDITIONS: Can run coffeemaker.jar, Ensure you are in a room with no door that leads to the South

EXECUTION STEPS:

1. Enter “S”
2. Observe results

POSTCONDITIONS: The move was not allowed and “A door in that direction does not exist” is displayed

IDENTIFIER: DRINK-ALL-TEST

TEST CASE: Ensure drinking all ingredients wins the game

PRECONDITIONS: Coffeemaker.jar can run, all three ingredients are in the user’s inventory

EXECUTION STEPS:

1. Enter “D”
2. Observe results

POSTCONDITIONS: The player wins the game

IDENTIFIER: DRINK-LESS-TEST

TEST CASE: Ensure that drinking less than all three ingredients does not win the game

PRECONDITIONS: Coffeemaker.jar can run, less than three ingredients have been gathered

EXECUTION STEPS:

1. Enter “D”
2. Observe results

POSTCONDITIONS: The player drinks those ingredients but loses the game

IDENTIFIER: DRINK-NONE-TEST

TEST CASE: Ensure that drinking none of the ingredients loses the game

PRECONDITIONS: Coffeemaker.jar can run, no ingredients have been gathered

EXECUTION STEPS:

1. Enter “D”
2. Observe results

POSTCONDITIONS: The user drinks nothing and loses the game

IDENTIFIER: DRINK-ONE-TEST

TEST CASE: Ensure that drinking only one ingredient will lose the game

PRECONDITIONS: Coffeemaker.jar can run, one ingredient is in the player’s inventory

EXECUTION STEPS:

1. Enter “D”
2. Observe results

POSTCONDITIONS: The user drinks the one ingredient and loses the game

IDENTIFIER: CHECK-INVENTORY-TEST

TEST CASE: Ensure that the user can always check the inventory during the game

PRECONDITIONS: coffeemaker.jar can run, the user did not drink yet

EXECUTION STEPS:

1. Enter “I”
2. Observe results

POSTCONDITIONS: The user is shown what ingredients have been and have not been collected of the three

IDENTIFIER: LOOK-AND-GRAB-TEST

TEST CASE: Ensure that when looking around, the user will add anything in the room to their inventory

PRECONDITIONS: Can run coffeemaker.jar

EXECUTION STEPS:

1. Run the program.
2. Enter L in each room.
3. Observe results

POSTCONDITIONS: Collects available items in room.

IDENTIFIER: ENTER-HELP-TEST

TEST CASE: Ensure the user can enter the help command at any time

PRECONDITIONS: coffeemaker.jar can run

EXECUTION STEPS:

1. Enter “H”
2. Observe results

POSTCONDITIONS: A list of possible commands is displayed

IDENTIFIER: UNIQUE-ROOM-TEST

TEST CASE: Ensure that each room has a unique identifier adjective

PRECONDITIONS: coffeemaker.jar can run

EXECUTION STEPS:

1. Record the adjective describing the room the user is in
2. Enter “N”
3. Repeat steps 1 and 2 until all rooms have been visited and the user can no longer move North
4. Compare all adjectives

POSTCONDITIONS: Each adjective recorded is unique and not the same as any other room

IDENTIFIER: ONE-UNIQUE-TEST

TEST CASE: Ensure that each room has only one furniture identifier

PRECONDITIONS: coffeemaker.jar can run

EXECUTION STEPS:

1. Record the number of furnishings in the room the user is in
2. Enter “N”
3. Repeat steps 1 and 2 until all rooms have been visited and the user can no longer move North
4. Observe

POSTCONDITIONS: The number of furnishings in each room is equal to 1

IDENTIFIER: UNIQUE-FURNISHING-TEST

TEST CASE: Ensure that a furnishing in any room is unique

PRECONDITIONS: coffeemaker.jar can run

EXECUTION STEPS:

1. Record the furnishing in the room the user is in
2. Enter “N”
3. Repeat steps 1 and 2 until all rooms have been visited and the user can no longer move North
4. Compare results

POSTCONDITIONS: Each furnishing is unique and different from any furnishing in any other room

1. **Defects**

**IDENTIFIER: D1**

**SUMMARY: No help command.**

**DESCRIPTION:** ENTER-HELP-TEST, COMMAND-ANYTIME-TEST, ALL-COMMANDS-TEST, there is no help command present in the game. Neither ‘H’ nor ‘h’ are recognized as valid commands and there is nothing mentioning them in game.

**REPRODUCTION STEPS:**

1. Run coffeemaker.jar
2. Input ‘H’ or ‘h’ when prompted for a command.
3. Observe results.

**EXPECTED BEHAVIOR:** Should display a help menu display possible commands and what they do.

**OBSERVED BEHAVIOR:** Program prints “What?” suggesting an incorrect command and no menu is displayed

**IDENTIFIER: D2**

**SUMMARY: ‘n’ not recognized as a command**

**DESCRIPTION:** LOWER-CASE-TEST, ‘n’ is not recognized as a command while ‘N’ is. This is the only command with this behavior.

**REPRODUCTION STEPS:**

1. Run coffeemaker.jar
2. Input ‘n’ when prompted for a command.
3. Observe results

**EXPECTED BEHAVIOR:** Should move north like ‘N’ if the move is possible.

**OBSERVED BEHAVIOR:** Program prints “What?” suggesting an incorrect command and stays in the same room.

**IDENTIFIER: D3**

**SUMMARY: User can exit the bounds and be reset to starting room.**

**DESCRIPTION:** NORTH-BOUND-TEST, SOUTH-BOUND-TEST, user can move north or south in a room without a corresponding door resulting in a different message being display and the user being reset to the initial room.

**REPRODUCTION STEPS:**

1. Run coffeemaker.jar
2. Enter ‘S’ or ‘s’ in the initial room or advance to the rough room and enter ‘N’
3. Observe behavior

**EXPECTED BEHAVIOR:** The move shall be disallowed with the message: "A door in that direction does not exist."

**OBSERVED BEHAVIOR:** The program says “You are in a magical land! But you are returned to the beginning!” and resets the user to the start room.