

**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: TEST-INPUT-CASE**

TEST CASE: Testing the user input to make sure it accepts all commands in upper and lower case.

PRECONDITIONS: coffeemaker.jar can run

EXECUTION STEPS: Run coffeemaker.jar and input each command

POSTCONDITIONS:

IDENTIFIER: COMMAND-ANYTIME-TEST

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

PRECONDITIONS:

EXECUTION STEPS:

POSTCONDITIONS:

IDENTIFIER: ALL-COMMANDS-START-TEST

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

PRECONDITIONS:

EXECUTION STEPS:

POSTCONDITIONS:

IDENTIFIER: UNKNOWN-COMMAND-TEST

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

PRECONDITIONS:

EXECUTION STEPS:

POSTCONDITIONS:

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:

EXECUTION STEPS:

POSTCONDITIONS:

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

PRECONDITIONS:

EXECUTION STEPS:

POSTCONDITIONS:

IDENTIFIER: ONE-UNIQUE-TEST

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

PRECONDITIONS:

EXECUTION STEPS:

POSTCONDITIONS:

IDENTIFIER: UNIQUE-FURNISHING-TEST

TEST CASE: Ensure that any item in any room is unique

PRECONDITIONS:

EXECUTION STEPS:

POSTCONDITIONS:

1. **Defects**

**IDENTIFIER: D1**

**SUMMARY:**

**DESCRIPTION:**

**REPRODUCTION STEPS:**

**EXPECTED BEHAVIOR:**

**OBSERVED BEHAVIOR:**

**IDENTIFIER: D2**

**SUMMARY:**

**DESCRIPTION:**

**REPRODUCTION STEPS:**

**EXPECTED BEHAVIOR:**

**OBSERVED BEHAVIOR:**

**IDENTIFIER: D3**

**SUMMARY:**

**DESCRIPTION:**

**REPRODUCTION STEPS:**

**EXPECTED BEHAVIOR:**

**OBSERVED BEHAVIOR:**