ESOF 322 Test Plan – A3 – Matthew Rohrlach

1 TEST PLAN IDENTIFIER

ESOF-322 Assignment 3

2 REFERENCES

There are no required references.

3 INTRODUCTION

This is the Test Plan for the text-based game, modified as part of a group project, in Assignment 2 of the Software Engineering course ESOF322. This document is written as part of Assignment 3, an individual assignment to test the validity of work done in Assignment 2.

This document details the methods and approach of the testing, with further details in the appropriate sections below.

4 TEST ITEMS (FUNCTIONS)

A. class “Door”

   methods:

   1. “enter”
      - Test the ability of a door to be passed through in all use cases

B. class “Player”

   methods:

   1. “go”
      - Test the ability of a player to ‘go’ in a direction
   2. “pickUp”
      - Test the ability of a player to retrieve items from the ground, and the placement of those items in the player’s inventory
   3. “drop”
      - Test the ability of a player to drop items, and the whereabouts of those items in both the inventory of the player and at the player’s location.

C. class “Room”

   methods:

   1. “addItem”

- Test the ability of a room to take in items, and the associated stats of that room.

2. “removeItem”
   - Test the ability of the room to delete items from its inventory.
3. “enter”
   - Test the ability of a room to be entered by a player, and the status reported.
4. “exit”
   - Test the ability of a room to be left by a player.

5 SOFTWARE RISK ISSUES

There are no software risks to note.

6 FEATURES TO BE TESTED

Features to be tested include the ability to enter doors with a key object, the ability of a player to move between rooms, to pick up and drop items, to add and remove items from rooms, and to enter and exit rooms.

7 FEATURES NOT TO BE TESTED

The behavior of the interface classes, the BreezySwing library, and the flavor text will not be tested.

8 APPROACH (STRATEGY)

Verification will be performed by instantiating test objects of the requisite classes. These test objects will be run through a coverage of use cases. After each action, an assertSame() call will be used to verify the correctness of the action. By determining the originators of unexpected results, testing will illustrate areas which need improvement.

Verification tests are created with code coverage in mind. Using the JaCoCoverage plugin for Netbeans, created by the EclEmma team, coverage is ensured on the branches and instructions of the methods to test.

9 ITEM PASS/FAIL CRITERIA

An action is considered to have ‘passed’ if its actions match expected results in the assertSame() call.

An action is considered to have ‘failed’ if its actions do not match expected results in the assertSame() call.

A class is only considered to have ‘passed’ when all of its actions produce expected results.

10 SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS

There is no cause for suspension. All testing will be carried out immediately.

11 TEST DELIVERABLES

Coverage reports in HTML form.

The executable JAR file of tests and classes.

12 REMAINING TEST TASKS

There are no tasks remaining.

13 ENVIRONMENTAL NEEDS

There are no specific environmental needs of the tests.

14 STAFFING AND TRAINING NEEDS

No advanced staffing and training needs are required.

15 RESPONSIBILITIES

All testing responsibilities fall to myself as the holder of all testing positions.

16 SCHEDULE

All testing will be done immediately after the conclusion of this document, with no delay.

17 PLANNING RISKS AND CONTINGENCIES

There are no risks to the testing procedure.

18 APPROVALS

No extraneous approval is need for the testing.

19 GLOSSARY

There are no terms within this document that require definition.