***Greedy Dwarf Deliverables, Testing and Inspections Report***

****

***A Dwarf Steals as Much Treasure as He Can Without Getting Caught by the Dragon***

**by**

**Alyssa Mihok, Jezebel Lopez and Ivana Pavlovic**

**in CS 440**

**at the**

**University of Illinois Chicago**

**November 2020**

**Table of Contents**

List of Figures 4

[List of Tables 5](#_Toc57496213)

[I Project Description 6](#_Toc57496214)

[1 Project Overview 6](#_Toc57496215)

[2 Project Domain 6](#_Toc57496216)

[3 Relationship to Other Documents 6](#_Toc57496217)

[II Project Deliverables 7](#_Toc57496218)

[1 First Release 7](#_Toc57496219)

[2 Second Release 8](#_Toc57496220)

[3 Comparison with Original Project Design Document 8](#_Toc57496221)

[III Testing 8](#_Toc57496222)

[1 Items to be Tested 8](#_Toc57496223)

[2 Test Specifications*)* 8](#_Toc57496224)

[3 Test Results 11](#_Toc57496225)

[4 Regression Testing 12](#_Toc57496226)

[IV Inspection 12](#_Toc57496227)

[1 Items to be Inspected 12](#_Toc57496228)

[2 Inspection Procedures 13](#_Toc57496229)

[3 Inspection Results 13](#_Toc57496230)

[V Recommendations and Conclusions 15](#_Toc57496231)

[VI Project Issues 15](#_Toc57496232)

[1 Open Issues 15](#_Toc57496233)

[2 Waiting Room 15](#_Toc57496234)

[3 Ideas for Solutions 16](#_Toc57496235)

[4 Project Retrospective 16](#_Toc57496236)

[VII Glossary 16](#_Toc57496237)

[VIII References / Bibliography 16](#_Toc57496238)

[IX Index 17](#_Toc57496239)

**List of Figures**

[*Figure 1.1 - Basic Map for Release 1*](#_heading=h.23ckvvd) *7*

[*Figure 1.2 - Map 2*](#_heading=h.ihv636) *8*

*Figure 1.3 - Map 3 8*

*Figure 1.4 - Map 4 8*

*On a related note, the references in the paragraph above, “( see* ***Error! Reference source not found.*** *on page )” include cross-references to the Figure and page number that will adjust automatically when other Figures or pages are added or removed. This is done with the “cross-reference” button in the “Captions” section of the “References” tab in MS Word.* ***Note: Remove this instructional paragraph.***

### List of Tables

[*Table 2.1 - Input/Output for Test 2.1*](#_heading=h.32hioqz) 9

*Table 2.2 - Input/Output for Test 2.2*10

*Table 2.3 - Input/Output for Test 2.3*11

# Project Description

## Project Overview

Greedy Dwarf is a top-down *Pac-Man* style arcade game that features a dwarf in a fantasy setting stealing gold and other treasures from a dragon. The more treasure that a player collects the higher their score will be but the player must collect at least one treasure item per level. The more treasure that is picked up the slower the dwarf will move forcing players to carefully select which treasures to collect on each level while the dragon chases them.

## Project Domain

This project is a product created for profit and entertainment value. Its domain is the gaming industry and has been created as an independent application to be bought and sold for a profit by its developers.

## Relationship to Other Documents

This document outlines the creation and testing of the game ***Greedy Dwarf*** based on the ***Greedy Dwarf Project Report* [5]*.*** The project requirements and overall design in this document were taken from the original report and tested after development and have been reported here.

## Project Deliverables

After 2 scenarios and 8 weeks of development our group managed to get a working game from the project report Greedy Dwarf.

## First Release

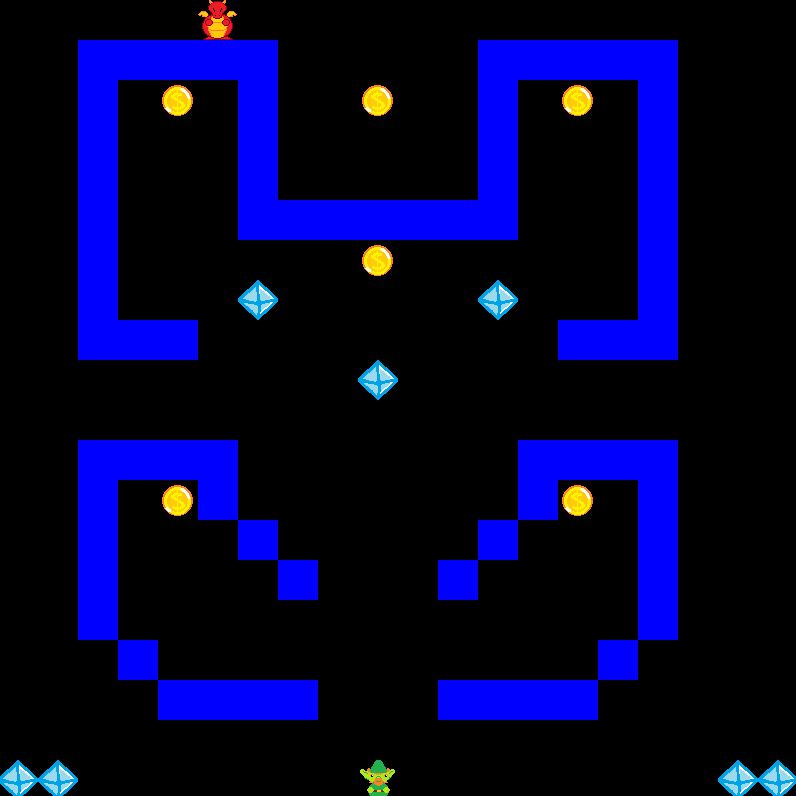
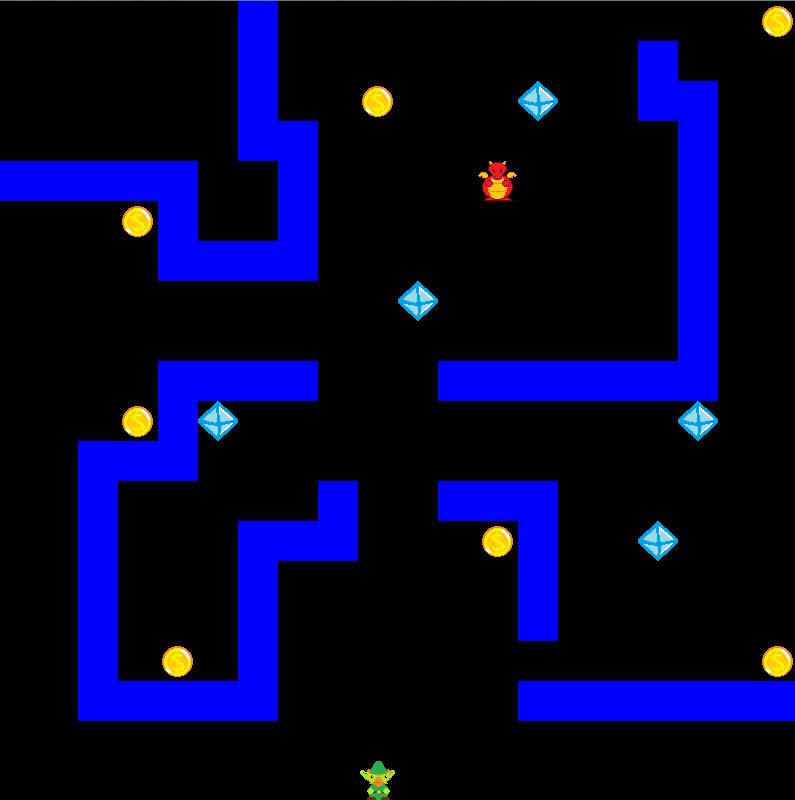
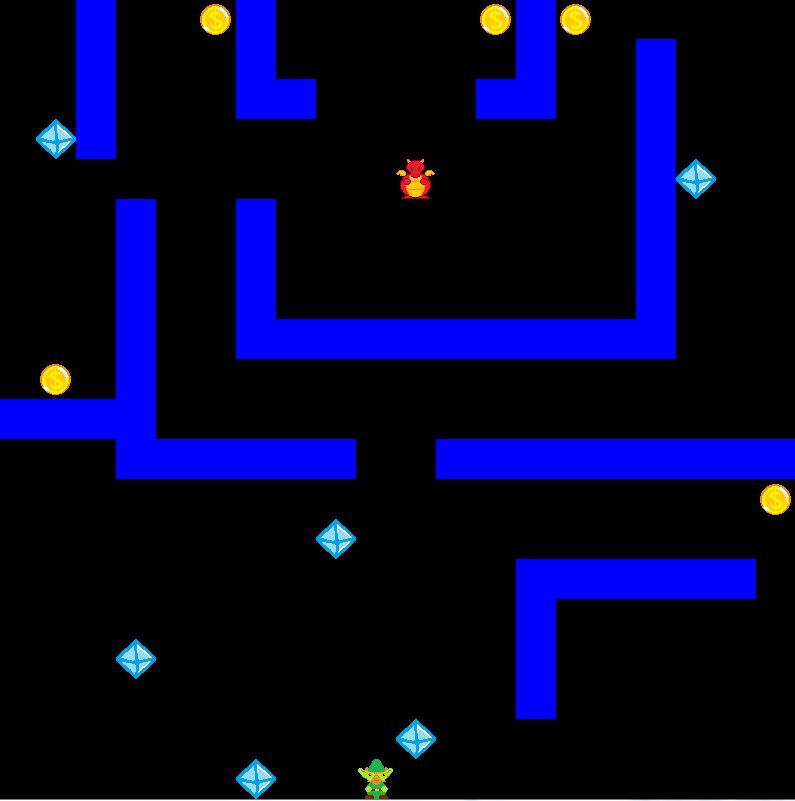
The key features implemented in this release included the majority of the user interface and important elements of gameplay such as picking up objects and the Dragon’s idle movement. There was a single large screen controlled by the Map class. The Map contained a collection of Tile objects for Creatures and Pickup objects to move and stand on. The layout of the Tiles and Map in this release is shown in **Figure 1.1** below. Animation threads were created to update and draw the location of each Tile on the Map and the dwarf was able to move around the Map freely with use of a Key Listener. This release was scheduled for October 9th.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | **D** |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | **T** |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | **P** |  |  |  |  |  |  |  |

***Figure 1.1 - Basic Map for Release 1***

## Second Release

The key elements implemented in the second release include the travelling salesman algorithm which affected the dwarf’s movement, the dragon’s chase function and range, and a variety of map layouts and treasure items. Whenever the dwarf picked up an item the player’s movement speed would slow down based on the weight and value of the item forcing the player to have to choose their items carefully to avoid being caught. The dragon also gained the ability to locate the player and chase after them once they either entered the dragon’s attack range or collected a single treasure item. Multiple map layouts were created and loaded in from a text file as shown in **Figures 1.2, 1.3 and 1.4** below and pick up objects with different weights and values were created in order to implement the travelling salesman algorithm. This release was scheduled for November 6th.



***Figure 1.2 - Map 2 Figure 1.3 - Map 3 Figure 1.4 - Map 4***

## Comparison with Original Project Design Document

The prototype produced looks more like a *Pac-Man* type of game whereas the full project described by the previous group looks more like a modern game. The prototype looks more like a maze game, and the project**[5]** described by the previous group was supposed to look like a room with a small portion of maze.

# Testing

## Items to be Tested

* Key Listener - in order to play the game this test is of a great importance to work at all times.
* Pick up Items - in order to be able to win or lose the game this test is also of great importance
* Dwarf’s speed - this test serves as a determinator for a small detail of the functionality of the game

## Test Specifications*)*

**2.1 - Key Listener Test**

**Description:** This test will determine whether or not the code responds to the keyboard presses in a timely manner and correctly.

**Items covered by this test:** Keyboard presses: the arrows, L and S keys.

**Requirements addressed by this test:** 10.2, 10.5, 12.2

**Environmental needs:** Hardware needed: Keyboard. For this test a different coder in the team will manually have to test this part of the code.

**Intercase Dependencies:** The whole game is dictated by the keyboard presses. This will have to be one of the first tests to be performed before moving on to another test.

**Test Procedures:** For this test different coders other than the one who implemented this feature will have to manually run tests to ensure that the key presses do what they are supposed to.

**Input Specification:** Arrow key presses in 3 variations patterns. (Example: U,D,L,R / D,R,L,U / L,D,R,U) The L (load game) and S (save game) will be tested at various spots in the game.

**Output Specifications:** When the S key is pressed the current game that is in play will be saved onto a text file to then be then loaded by theL key. The arrow keys should move the dwarf in the direction of the arrow specification.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Input: Keypress** | Up Arrow | Down Arrow | Right Arrow | Left Arrow | L Key | S key |
| **Output:** | Dwarf moves up | Dwarf moves down | Dwarf moves right | Dwarf  moves left | Loaded saved game in text file | Saves current game to text file |

***Table 2.1 - Input/Output for Test 2.1***

**Pass/Fail Criteria:** All the keys do what as the table above.

**2.2 - Pick Up Items Test**

**Description:** This test determines whether or not all functions associated with the player collecting treasure items on the map function correctly.

**Items covered by this test:** Intersection with the player object adds to the removal array list, removes the object from the total tiles array list, and player score is updated based on the item picked up.

**Requirements assessed by this test:** 10.4, 10.5

**Environmental Needs:** Hardware needed:Keyboard.

**Intercase Testing:** Test 2.1 must be completed.

**Test Procedures:** For this test a programmer other than the original writer must manually test that the items are removed from the map when picked up and that the player score changes based on the correct item value.

**Input Specification:** Player intersection with each of the treasure objects (Gold, Ruby, Diamond) will be tested on each map.

**Output Specifications:** The object that is picked up is removed from the map tile list and is no longer visible to the player and the player’s score will be updated and visible to the player.

|  |  |  |  |
| --- | --- | --- | --- |
| Input Interaction | Player with Gold | Player with Ruby | Player with Diamond |
| Output | Gold is no longer visible on the map, cannot be found in the Tile Arraylist, and Score is now 1 | Ruby is no longer visible on the map, cannot be found in the Tile Arraylist, and Score is now 5 | Diamond is no longer visible on the map, cannot be found in the Tile Arraylist and Score is now 10 |

***Table 2.2 - Input/Output for Test 2.2***

**Pass/Fail Criteria:** All maps in game must respond as described in the table

above.

**2.3 - Dwarf’s speed test**

**Description:** The test will determine whether the dwarf changes the speed after picking up different treasures.

**Items covered by this test:** The initial speed at every level, differences in speed after the pickup of coin, ruby and diamond separately and together.

**Requirements addressed by this test:** 10.4

**Environmental needs:** Hardware needed: Keyboard

**Intercase Dependencies:** Tests 2.1, 2.2 must be completed prior to this one.

**Test Procedures:** The dwarf’s speed would need to be tested first after picking up only one coin, then ruby, and then diamond. The next step is to test the speed after picking up of one treasure at each level. The step after is to test the speed after the pickup of all treasures at level 1. The last step is to test the speed after the collision with the dragon.

**Input Specification:** One gold coin; One ruby; One diamond; Together coin, ruby and diamond; Collision with diamond; Level 1, Level 2, Level 3, Level 4.

**Output Specifications:** Speed when no treasure is picked up - 40; Speed when coin is picked up - 30; Speed when ruby is picked up - 20; Speed when diamond is picked up - 10; Speed when all treasures are picked up - 5; Speed after the collision with diamond - 5 greater than the previous, 10 total.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Input:** | Coin | Ruby | Diamond | All together | Collision with dragon | Level 1/2/3/4 |
| **Output:** | Dwarf moves up | Dwarf moves down | Dwarf moves right | Dwarf  moves left | Loaded saved game in text file | Change in speed like in previous tests |

***Table 2.3 - Input/Output for Test 2.3***

**Pass/Fail Criteria:** The speed is accurate after each pick up and collision with diamond.

## Test Results

**2.1.1 - Key Listener Test Results**

**Date(s) of Execution:** This test was run for a week long each test was conducted once per day.

**Staff conducting tests:** Jazbel Lopez

**Expected Results:** Keys respond in a timely manner and do what they are specified in the table of 2.1 test.

**Actual Results:** The keys responded as specified and in a timely manner.

**Test Status:** Pass

**2.2.1 - Item Pick Up Results**

**Date(s) of Execution:** This test was run once a day for a week-long period.

**Staff conducting tests:** Alyssa Mihok

**Expected Results:** Items are removed appropriately from the map and score is updated appropriately in accordance with the table of the 2.2 test.

**Actual Results:** All items on all maps respond in the appropriate manner.

**Test Status:** Pass

**2.3.1 - Dwarf’s Speed Test Results**

**Date(s) of Execution:** This test was run once a day for a week-long period.

**Staff conducting tests:** Ivana Pavlovic

**Expected Results:** Speeds are accurate after pick up of the coin, ruby and diamond, and after the collision with the dragon.

**Actual Results:** The speed changed as expected. Speed after coin pick up - 30, ruby - 20, diamond - 10, together - 5, after the collision with the dragon - 10. The speed was affected the same at each level.

**Test Status:** Pass.

## Regression Testing

The test 2.2 had to be repeated until the pick up of the treasures worked on all different levels. Initial tests failed after changing levels.

# Inspection

## Items to be Inspected

**Inspection 1:**

For the first inspection test to be conducted will be on the game state class, map component class and key listener class. These classes are the main portion to get each level set up along with the main game play functionality.

**Inspection 2:**

An inspection must be conducted on the code for the functions that control the player’s ability to pick up items and be killed by the dragon as well as the chase feature of the dragon. These features are necessary for game play.

**Inspection 3:**

The last inspection that must be conducted is on the code for the main menu, dragon movements, dragon range, and dwarf’s speed through the game, after collecting the treasures and colliding with the dragon.

## Inspection Procedures

As part of the inspection process we had two meetings a week apart. Our initial meeting took place on November 16th. We all gathered together to decide which checklist to use. We decided that the Java Inspection Checklist created by Christopher Fox**[6]** would be the best checklist for us to use. Along with the checklist we all presented parts of the code to be inspected. A week later on November 23rd we met up and took turns discussing the results of our inspections.

## Inspection Results

**Inspection 1 results:**

**Inspection done by:** Jazbel Lopez

**Inspection date:** 11/20/2020

**Results of the inspection:**

Game state - The instructions are clear, naming conventions are relevant to what they method/variable is doing. In terms of functionality the methods open and close text files as it should and reads in each line of the text files. The Array List is then populated as it should with objects. However this class is missing a constructor to properly initialize the variables. This class is also missing some private/protected variables.

Map component - Simple yet one of the more important classes to create the maps. All methods/variables are protected and/or private so users won’t be able to interfere with the functionality of the game.

Key Control - This class is the main class for the user to control the game. This is one of the more straightforward classes, only controlling the 6 key presses the user will use to interact with the game. This class is always waiting and listening for the key press that then will translate to the dwarf movement.

**Re-inspection:** The re-inspection might be done in the future.

**Inspection 2 results:**

**Inspection done by:** Alyssa Mihok

**Inspection date:** 11/20/2020

**Results of the inspection:**

Item Pick Up - Checks for intersections between the Player tile object and other tile objects. Comments for each if statement make it clear what each part and function used is for and descriptive names are used for variables and methods. Works properly but could be more efficient by creating smaller arraylists with more specific types rather than one huge list of all tiles.

Player Death - Functions similarly to the item pick up but checks for intersections between the Player tile object and the Dragon tile object. Comments well written and descriptive variable names used so it is clear what the logic being used is. Removes player and returns to the main menu after points drop to zero or less.

Dragon’s Chase Function - Boolean values used to check if the chase function should be active, variable name for this could be more descriptive. The catch player function uses a series of if statements to move the dragon in the direction of the player that might work better as a switch statement.

**Re-inspection:** The re-inspection might be done in the future.

**Inspection 3 results:**

**Inspection done by:** Ivana Pavlovic

**Inspection date:** 11/20/2020

**Results of the inspection:**

Main menu - the buttons open the instructions and the game window as they are supposed to, and the instructions are clear.

Dragon movements - The dragon moves randomly in a diamond shaped movement as required, and moves diagonally towards the dwarf once the treasure is collected.

Dragon’s range - The dragon’s range is clearly shown, however the dragon is not chasing the dwarf as supposed to.

Dwarf’s speed - The dwarf’s speed is changed every time the treasure is picked up by the dwarf or after the dwarf’s collision with the dragon, and has the same value as the expected.

**Re-inspection:** The re-inspection might be done in the future.

# Recommendations and Conclusions

In regard to the key listener test all tests passed, however through testing we found that the user is able to click through faster than the map can update. Currently the map is updating at 250 millisecond which we thought that would be fast enough update time; the player is able to move too fast that at times the system can’t identify when a player is over a treasure and won’t pick it up and delete the piece from the map. To address this issue the team would either have to have the map update faster or have a restriction on how fast the user is able to click through the arrows.

The map’s main thread is where the player’s interactions with all other tile objects are being checked. For more efficient collision checking instead of using one large tile array list containing all of the tiles on the map, smaller lists of more specific object types like Pickup or Creature could be used to check collisions.

Another issue that we found was the dragon’s range function. Currently the range is only visible but not doing the function it is supposed to - chasing the dwarf once it’s in the range. The recommended solution would be to use the same code that is used for the dragon chasing after the dwarf would pick up the treasure. Another recommendation would be to increase the dragon range in higher levels.

Other additions to this project might be the sound effects when the dragon collides with the dwarf, when the dwarf picks up the treasures, and when the game starts and ends.

# Project Issues

## Open Issues

At the time of developing the prototype our team has it set up that a saved game and high score are saved on a text file on the local machine. As the game progresses the system will have to have enough memory to store each user’s individual saved game along with the highest score they have achieved.

## Waiting Room

An idea that should be kept in mind but held off for future releases is a multiplayer game. Currently a single player has to collect at least one treasure to move on to the next level, but with the multiplayer game all the dwarfs on the screen would have to collect almost all the treasure on the map before moving on to the next level.

As the game currently stands it doesn’t not have a net code implementation. This will be a crucial part to develop a multiplayer game. The system will also have detected features against cheating and account management. In order for the multiplayer game to work, player users have to have a way to store their ranking in the game along with a username.

## Ideas for Solutions

The original business analyst of the game thought of a way to make the game more inclusive to a wider range of people. The game has a specific color pallet however we might have some users who have color blindness and might have trouble viewing the display of the game. The game could have in the setting to change the color pallet of the game. We can have a team study what color pallets will be suitable for a wide variety of color blindness. These color pallets can be tested through volunteers to see which color pallets would work best.

## Project Retrospective

During the development of this project our team met once a week. Though during our meeting, we would catch each other up with each other’s progress and assign new tasks. We found that on some occasions we would have to meet up more than once a week. In the future more frequent but shorter meetings may help improve our team’s dynamic.

One of the biggest advantages we did have was when groupmates had to develop similar code they would meet up to code alongside each other. We found that this helped the development team immensely since they would have someone to talk through the portion of the code they are having trouble with.

# Glossary

**Dragon range** - the area around the dragon where if the dwarf steps in the dragon is supposed to chase it.

**Tile** - any object visible on the map

# References / Bibliography

|  |  |
| --- | --- |
| **[1]** | **Robertson and Robertson, Mastering the Requirements Process.** |
| **[2]** | **A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.** |
| **[3]** | **J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.** |
| **[4]**  **[5]**  **[6]** | **M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.**  **A. Lau, C. Riviera, M. Searcy “Greedy Dwarf Project Report”**  **C. Fox “Java Inspection Checklist”, 1999** |
|  |  |

# Index

[Project Description](#_heading=h.tyjcwt) 4

[Project Deliverables](#_heading=h.lnxbz9) 8

[Testing](#_heading=h.svtci2elahu8) 9

[Inspection](#_heading=h.z337ya) 12

[Recommendations and Conclusions](#_heading=h.1ci93xb) 13

[Project Issues](#_heading=h.3whwml4) 13