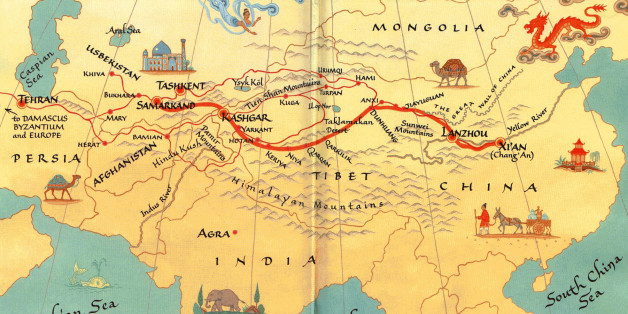
**The Silk Route Adventures**



Group 10

Prepared by:

Christopher Khoshaba,

Joel Koreth,

Christian Ramirez Gomez,

Will Davidson

April 2020

**Table of Contents**

[List of Figures 4](#_Toc70158360)

[List of Tables 5](#_Toc70158361)

[I Project Description 6](#_Toc70158362)

[1 Project Overview 6](#_Toc70158363)

[2 Project Domain 6](#_Toc70158364)

[3 Relationship to Other Documents 6](#_Toc70158365)

[4 Naming Conventions and Definitions 6](#_Toc70158366)

[4a Definitions of Key Terms 6](#_Toc70158367)

[4b UML and Other Notation Used in This Document 6](#_Toc70158368)

[4c Data Dictionary for Any Included Models 7](#_Toc70158369)

[II Project Deliverables 7](#_Toc70158370)

[5 First Release 8](#_Toc70158371)

[6 Second Release 9](#_Toc70158372)

[7 Comparison with Original Project Design Document 10](#_Toc70158373)

[III Testing 11](#_Toc70158374)

[8 Items to be Tested 11](#_Toc70158375)

[9 Test Specifications 11](#_Toc70158376)

[10 Test Results 20](#_Toc70158377)

[11 Regression Testing 25](#_Toc70158378)

[IV Inspection 25](#_Toc70158379)

[12 Items to be Inspected 25](#_Toc70158380)

[13 Inspection Procedures 25](#_Toc70158381)

[14 Inspection Results 26](#_Toc70158382)

[15 Recommendations and Conclusions 26](#_Toc70158383)

[V Project Issues 26](#_Toc70158384)

[16 Open Issues 26](#_Toc70158385)

[17 Waiting Room 27](#_Toc70158386)

[18 Ideas for Solutions 27](#_Toc70158387)

[19 Project Retrospective 28](#_Toc70158388)

[VI Glossary 28](#_Toc70158389)

[VII References / Bibliography 28](#_Toc70158390)

[VIII Index 28](#_Toc70158391)

### ****List of Figures****

[Figure 1 – R1 Kingdom Selection Screen 8](https://d.docs.live.net/3b42350a1c2b7989/Documents/UIC/5thSemester/CS440/440-Group-10-Spring-2021/Code/group10CodingProjectFinalReport%20-%20Christian.docx#_Toc70158392)

[Figure 2 – R1 Game Mode Screen 8](https://d.docs.live.net/3b42350a1c2b7989/Documents/UIC/5thSemester/CS440/440-Group-10-Spring-2021/Code/group10CodingProjectFinalReport%20-%20Christian.docx#_Toc70158393)

[Figure 3 – R1 Name Entry Screen 8](https://d.docs.live.net/3b42350a1c2b7989/Documents/UIC/5thSemester/CS440/440-Group-10-Spring-2021/Code/group10CodingProjectFinalReport%20-%20Christian.docx#_Toc70158394)

[Figure 4 – R1 Home Screen 8](https://d.docs.live.net/3b42350a1c2b7989/Documents/UIC/5thSemester/CS440/440-Group-10-Spring-2021/Code/group10CodingProjectFinalReport%20-%20Christian.docx#_Toc70158395)

[Figure 5 – R1 UI 8](https://d.docs.live.net/3b42350a1c2b7989/Documents/UIC/5thSemester/CS440/440-Group-10-Spring-2021/Code/group10CodingProjectFinalReport%20-%20Christian.docx#_Toc70158396)

[Figure 6 – R2 Broken Wagon Event 9](https://d.docs.live.net/3b42350a1c2b7989/Documents/UIC/5thSemester/CS440/440-Group-10-Spring-2021/Code/group10CodingProjectFinalReport%20-%20Christian.docx#_Toc70158397)

[Figure 7 - R2 Midgame Alert 9](https://d.docs.live.net/3b42350a1c2b7989/Documents/UIC/5thSemester/CS440/440-Group-10-Spring-2021/Code/group10CodingProjectFinalReport%20-%20Christian.docx#_Toc70158398)

[Figure 8 - R2 Gift Event 9](https://d.docs.live.net/3b42350a1c2b7989/Documents/UIC/5thSemester/CS440/440-Group-10-Spring-2021/Code/group10CodingProjectFinalReport%20-%20Christian.docx#_Toc70158399)

[Figure 9 - R2 Sandstorm Event 9](https://d.docs.live.net/3b42350a1c2b7989/Documents/UIC/5thSemester/CS440/440-Group-10-Spring-2021/Code/group10CodingProjectFinalReport%20-%20Christian.docx#_Toc70158400)

[Figure 10 – R2 Bandit Attack Event 9](https://d.docs.live.net/3b42350a1c2b7989/Documents/UIC/5thSemester/CS440/440-Group-10-Spring-2021/Code/group10CodingProjectFinalReport%20-%20Christian.docx#_Toc70158401)

[Figure 11 – R2 Last 5 Turns Alert 9](https://d.docs.live.net/3b42350a1c2b7989/Documents/UIC/5thSemester/CS440/440-Group-10-Spring-2021/Code/group10CodingProjectFinalReport%20-%20Christian.docx#_Toc70158402)

[Figure 13 – R2 Game Results Screen 10](https://d.docs.live.net/3b42350a1c2b7989/Documents/UIC/5thSemester/CS440/440-Group-10-Spring-2021/Code/group10CodingProjectFinalReport%20-%20Christian.docx#_Toc70158403)

[Figure 14 – R2 Bandit Attack Result 10](https://d.docs.live.net/3b42350a1c2b7989/Documents/UIC/5thSemester/CS440/440-Group-10-Spring-2021/Code/group10CodingProjectFinalReport%20-%20Christian.docx#_Toc70158404)

[Figure 12 - R2 UI 10](https://d.docs.live.net/3b42350a1c2b7989/Documents/UIC/5thSemester/CS440/440-Group-10-Spring-2021/Code/group10CodingProjectFinalReport%20-%20Christian.docx#_Toc70158405)

### ****List of Tables****

[Table 1 - Buy and Sell Test 12](#_Toc70158406)

[Table 2 - Ability to Buy and Sell Test 13](#_Toc70158407)

[Table 3 - Build Post Test 14](#_Toc70158408)

[Table 4 - Starting Post Bonus Test 15](#_Toc70158409)

[Table 5 - Starting Money Bonus Test 15](#_Toc70158410)

[Table 6 - Starting Sell Bonus Test 16](#_Toc70158411)

[Table 7 - Fight Results Test 17](#_Toc70158412)

[Table 8 - Drought Test 18](#_Toc70158413)

[Table 9 - Harvest Test 18](#_Toc70158414)

[Table 10 - Gift Test 19](#_Toc70158415)

[Table 11 - Movement Test 20](#_Toc70158416)

# Project Description

## Project Overview

The project is about developing a game called the Silk Route. The Silk Route is a Computer game based in Java. This game is meant to provide consumers with a strategy game, based in the time of the Roman Empire, that focuses on maximizing profits. The game further takes inspiration from Role-Playing games to add further layers of depth to the strategy. To encourage adaptation, the game further includes numerous elements of randomness. Randomness is meant to prevent the same strategy from always being played.

## Project Domain

PC Gamers are the target demographic for this project, and the game attempts to cater to the historical and strategy game niche subsection of the population. The game is set in antiquity, but the focus of the game is not historical accuracy, but rather the maintenance of a general historical theme. The strategy elements of the game are meant to cater to people who enjoy quick thinking, and ever-changing optimal strategies.

## Relationship to Other Documents

The Silk Route Adventures Project Report Prepared by Sukhmani Khehra, Brook Habtegiorgis, Ipsit Patra, Daniel Moren was used as a basis for the project's development.

## Naming Conventions and Definitions

### Definitions of Key Terms

Silk Route: Used to refer to the Silk Route game. Unless otherwise stated, this is the default meaning of the term, and in other cases, the meaning will be provided through context.

RNG: Random Number Generator. Used to refer to randomness within the game.

Strategy: Used to refer to how a player makes decisions when presented with choice within the game.

Historical: Meant to refer to a theme, not necessarily historical accuracy.

### UML and Other Notation Used in This Document

UML present within this document adheres to general UML guidelines. Refer to UML Distilled by Fowler for further elaboration.

### Data Dictionary for Any Included Models

Data Structures

1. Classes
   1. BanditEvent
   2. EventScreen
   3. GameGUI
   4. HexData
   5. HexType
   6. Hexagon
   7. Map
   8. MapData
   9. Point
2. Basic Data Structures
   1. Hash Map (Hash Tables)
   2. Lists
   3. Arrays
   4. Enum
3. Values within Game
   1. Wealth/Cash (Double Value)
   2. Action Points (Integer Value)

# Project Deliverables

For this project we created the basic game play for a turn-based game based on the Silk Road. For this game players can move to different countries and buy and sell goods in each country. There are also random events that pop up that can either hurt or help the player. The players when signing into the game can create a username and choose an avatar to represent them.

## First Release

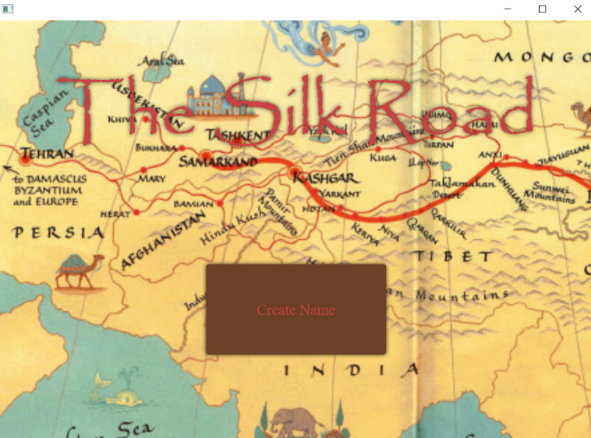
**he first release for the project was on February 26. For this part the basics for logging into the game were created. The player would enter their name, choose a game mode, and then choose their starting country on the map. For the map the player was able to move around and buy and sell in each different country. Action points were put in place to limit how far the player could move or how many items they could buy or sell in one turn. For buying there were some issues where the player could buy more items then they could afford, and they would go into negative for their wallet. Besides that, every other aspect worked properly.

Figure 1 – R1 Kingdom Selection Screen

Figure 2 – R1 Game Mode Screen

Figure 3 – R1 Name Entry Screen

Figure 4 – R1 Home Screen

**

Figure 5 – R1 UI

## Second Release

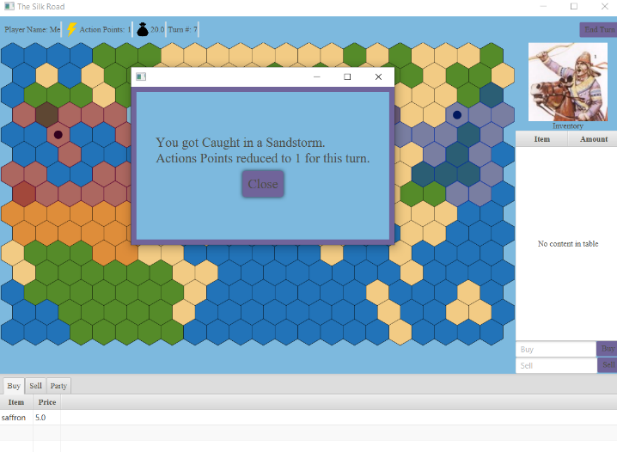
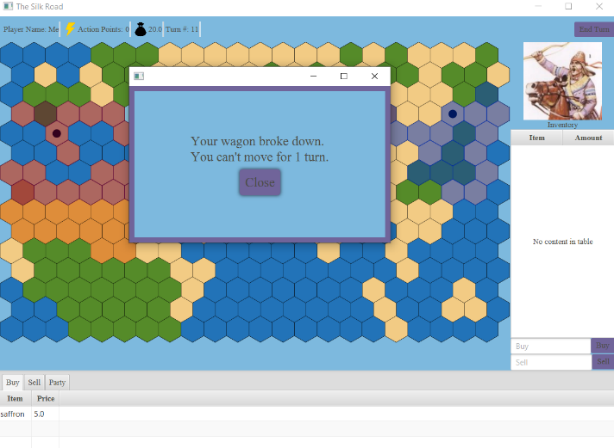
The second release for the project was on April 2. For this release the first main thing addressed was fixing the buying problems from the last release. Now the game wouldn’t let the player buy more items then they could afford. For new additions an avatar selector mode was added to let the player choose an avatar for the top right of their screen. Another new addition was implementing random events. For these random events the outcomes could be positive or negative for the player. One random event had it so the players would have to decide and depending on what they choose the result could be positive or negative.

Figure 6 – R2 Broken Wagon Event

Figure 7 - R2 Midgame Alert

Figure 8 - R2 Gift Event

Figure 9 - R2 Sandstorm Event



Figure 10 – R2 Bandit Attack Event

Figure 11 – R2 Last 5 Turns Alert

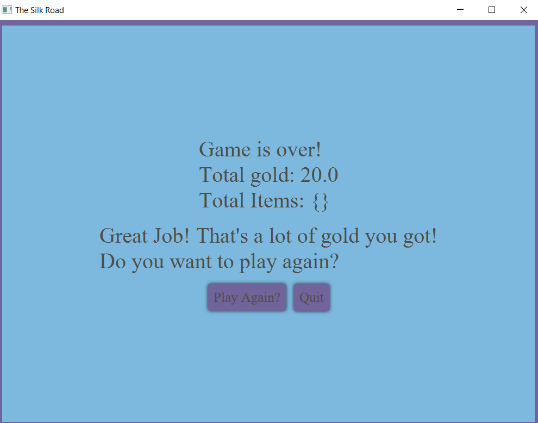
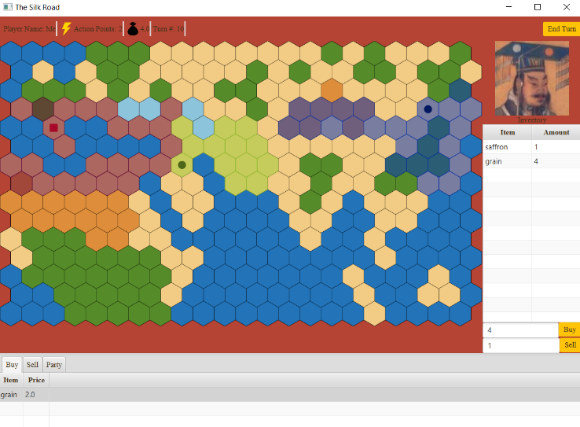


Figure 13 – R2 Game Results Screen

Figure 14 – R2 Bandit Attack Result

Figure 12 - R2 UI

## Comparison with Original Project Design Document

The original description of the project wanted to create a turn-based game based on the silk road that is historically accurate where the player would be a merchant and go to different countries to trade recourses. For this aspect our group was able to accurately create this. The countries used for the game and the main recourses for each country are historically accurate. Where our project differs is in the learning aspect. For the original description of the project the idea was that playing the game would help to teach the player about the silk road’s history. There would be certain challenges and mini games for the player to help teach them about the silk road. The project created was not able to completely deliver on this. Instead of mini games too this project uses random events for player interaction.

Another aspect of the original description that was changed was the log in feature. The original wanted to have a database to save the log in information for the player, but for the current project it was decided to just have the player enter their name every time for simplicity reasons. This database would also contribute to a “Play with Friend” option which this project was not able to implement.

# Testing

## Items to be Tested

**Item ID#T – Trader Class**

**Description:** Check for the main functions of a trader.

**Item ID#C – Combat and Mercenary Class**

**Description:** Check all possible combat results between mercenary and trader.

**Item ID#E – Event Classes**

**Description:** Check each event type and the result in game data.

**Item ID#L – Location Data Class**

**Description:** Check for correct file input.

## Test Specifications

**ID#T01 - Constructor Test**

**Description:** Verify correct construction of class.

**Items covered by this test:** Item ID#T – Trader Class

**Requirements addressed by this test:** Not Applicable.

**Environmental needs:** Test as Maven Project.

**Intercase Dependencies:** Not Applicable.

**Test Procedures:** Command: “mvn test”

**Input Specification:** strings = {"Christian", "Christopher", "Joel", "Will"}

**Output Specifications:** {"Christian", "Christopher", "Joel", "Will"}

**Pass/Fail Criteria:** Should pass the automated test by having the correct name.

**ID#T02 - Action Test**

**Description:** Verify actions points are subtracting from Trader correctly.

**Items covered by this test:** Item ID#T – Trader Class

**Requirements addressed by this test:** Not Applicable.

**Environmental needs:** Test as Maven Project.

**Intercase Dependencies:** Not Applicable.

**Test Procedures:** Command: “mvn test”

**Input Specification:** ints = {3, 2, 1}

**Output Specifications:** {0,1,2}

**Pass/Fail Criteria:** Should pass the automated test by subtracting one point.

**ID#T03 - Buy and Sell Test**

**Description:** Buy and sell a good to see if all the functions are working together correctly.

**Items covered by this test:** Item ID#T – Trader Class

**Requirements addressed by this test:** Not Applicable.

**Environmental needs:** Test as Maven Project.

**Intercase Dependencies:** Not Applicable.

**Test Procedures:** Command: “mvn test”

**Input Specification:** City File: 3Cities.json

**Output Specifications:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Quantity** | **Remaining Gold** | **Quantity After Selling** | **Final Gold** |
| 0 | 10.0 | 0 | 10.0 |
| 1 | 8.0 | 0 | 2008.0 |
| 5 | 0.0 | 0 | 10000.0 |

Table 1 - Buy and Sell Test

**Pass/Fail Criteria:** Should pass the automated test by buying goods in one city at one price and then selling in another city at a higher price. The increase in gold should be correct.

**ID#T04 - Ability to Buy and Sell Test**

**Description:** Tests the Boolean functions used to check if you can buy or sell beforehand.

**Items covered by this test:** Item ID#T – Trader Class

**Requirements addressed by this test:** Not Applicable.

**Environmental needs:** Test as Maven Project.

**Intercase Dependencies:** Not Applicable.

**Test Procedures:** Command: “mvn test”

**Input Specification:** City File: 3Cities.json

**Output Specifications:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Item Quantity** | **City** | **Can Buy** | **Can Sell** |
| 10 | CityA | true | true |
| 11 | CityA | false | false |
| 1 | CityO | false | false |

Table 2 - Ability to Buy and Sell Test

**Pass/Fail Criteria:** Should be able to buy when you have enough gold and only be able to sell when you have the correct amount of that good.

**ID#T05 - Build Post Test**

**Description:** Tests the function that builds the Post the Trader will own.

**Items covered by this test:** Item ID#T – Trader Class

**Requirements addressed by this test:** Not Applicable.

**Environmental needs:** Test as Maven Project.

**Intercase Dependencies:** Not Applicable.

**Test Procedures:** Command: “mvn test”

**Input Specification:** City File: 3Posts.json

**Output Specifications:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Item Quantity** | **Gold** | **City** | **Can Build** |
| 3 | 0 | CityA | False |
| 0 | 5.0 | CityA | False |
| 3 | 5.0 | CityA | True |
| 0 | 50.0 | CityO | False |
| 3 | 5.0 | CityO | False |
| 3 | 50.0 | CityO | True |

Table 3 - Build Post Test

**Pass/Fail Criteria:** You should only be able to build a post when you have enough materials and gold as described by the input file.

**ID#T06 - Starting Post Bonus Test**

**Description:** Tests the functionality of the Trader’s starting bonus that reduces the cost of building a post.

**Items covered by this test:** Item ID#T – Trader Class

**Requirements addressed by this test:** Not Applicable.

**Environmental needs:** Test as Maven Project.

**Intercase Dependencies:** Not Applicable.

**Test Procedures:** Command: “mvn test”

**Input Specification:** Bonus: Imperial\_Legacy, File: 2Posts.json

**Output Specifications:**

|  |  |
| --- | --- |
| **Input Bonus** | **Remaining Gold** |
| Imperial\_Legacy | 3.75 |
| None | 5.0 |

Table 4 - Starting Post Bonus Test

**Pass/Fail Criteria:** The cost of building a post should go down by 25% in gold based on the input file compared to a trader without it.

**ID#T07 - Starting Money Bonus Test**

**Description:** Tests the functionality of the Trader’s starting bonus that increases the starting gold.

**Items covered by this test:** Item ID#T – Trader Class

**Requirements addressed by this test:** Not Applicable.

**Environmental needs:** Test as Maven Project.

**Intercase Dependencies:** Not Applicable.

**Test Procedures:** Command: “mvn test”

**Input Specification:** Bonus: Economic\_Giant

**Output Specifications:**

|  |  |
| --- | --- |
| **Input Bonus** | **Starting Gold** |
| Economic\_Giant | 15.0 |
| None | 10.0 |

Table 5 - Starting Money Bonus Test

**Pass/Fail Criteria:** Trader should start with 15 gold, compared to 10 without this bonus.

**ID#T08 - Starting Sell Bonus Test**

**Description:** Tests the functionality of the Trader’s starting bonus that increases the gold received from selling by 2%.

**Items covered by this test:** Item ID#T – Trader Class

**Requirements addressed by this test:** Not Applicable.

**Environmental needs:** Test as Maven Project.

**Intercase Dependencies:** Not Applicable.

**Test Procedures:** Command: “mvn test”

**Input Specification:** Bonus: Middlemen\_of\_the\_Silk\_Road, File: 3Cities.json

**Output Specifications:**

|  |  |
| --- | --- |
| **Input Bonus** | **Gold After Sale** |
| Middlemen\_of\_the\_Silk\_Road | 10200.0 |
| None | 10000.0 |

Table 6 - Starting Sell Bonus Test

**Pass/Fail Criteria:** Trader should be making 2% more when they sell in a city compared to the prices set in the input file.

**ID#C01 – Luck Test**

**Description:** Tests the functionality of escaping battle though your luck skill points.

**Items covered by this test:** Item ID#C – Combat and Mercenary Class

**Requirements addressed by this test:** Not Applicable.

**Environmental needs:** Test as Maven Project.

**Intercase Dependencies:** Not Applicable.

**Test Procedures:** Command: “mvn test”

**Input Specification:** int = {0,100}

**Output Specifications:** CombatResult = {Victory, Escape }

**Pass/Fail Criteria:** Trader should always escape if their luck is at 100 points. If it is always at 0 points they should win. Values in between cant be tested automatically since they are random.

**ID#C02 – Fight Results Test**

**Description:** Tests the functionality of when you fight a mercenary or a bandit in the bandit event.

**Items covered by this test:** Item ID#C – Combat and Mercenary Class

**Requirements addressed by this test:** Not Applicable.

**Environmental needs:** Test as Maven Project.

**Intercase Dependencies:** Not Applicable.

**Test Procedures:** Command: “mvn test”

**Input Specification:** Different Mercenaries with different attack and defense points.

**Output Specifications:**

|  |  |  |
| --- | --- | --- |
| **Input Attack** | **Input Defense** | **CombatResult** |
| 1 | 1 | Tie |
| 2 | 1 | Loss |
| 0 | 0 | Victory |

Table 7 - Fight Results Test

**Pass/Fail Criteria:** Trader always starts with 1 attack and 1 defense. Theyb are compared by how much damage and the person who does the most damage will win the fight.

**ID#E01 – Drought Test**

**Description:** Test the functionality of the effect of a Drought Event.

**Items covered by this test:** Item ID#E – Event Classes

**Requirements addressed by this test:** Not Applicable.

**Environmental needs:** Test as Maven Project.

**Intercase Dependencies:** Not Applicable.

**Test Procedures:** Command: “mvn test”

**Input Specification:** Item to buy from a city.

**Output Specifications:**

|  |  |  |
| --- | --- | --- |
| **Input Item** | **Price Before** | **Price After** |
| silk | 1.0 | 1.25 |
| tea | 2.0 | 2.5 |

Table 8 - Drought Test

**Pass/Fail Criteria:** Prices in a city should increase by 25% after the event.

**ID#E02 – Harvest Test**

**Description:** Test the functionality of the effect of a Plentiful Harvest Event.

**Items covered by this test:** Item ID#E – Event Classes

**Requirements addressed by this test:** Not Applicable.

**Environmental needs:** Test as Maven Project.

**Intercase Dependencies:** Not Applicable.

**Test Procedures:** Command: “mvn test”

**Input Specification:** Item to buy from a city.

**Output Specifications:**

|  |  |  |
| --- | --- | --- |
| **Input Item** | **Price Before** | **Price After** |
| silk | 1.0 | 0.75 |
| tea | 2.0 | 1.50 |

Table 9 - Harvest Test

**Pass/Fail Criteria:** Prices in a city should decrease by 25% after the event.

**ID#E03 – Destroyed City Test**

**Description:** Test the functionality of the effect of a City Destroyed Event.

**Items covered by this test:** Item ID#E – Event Classes

**Requirements addressed by this test:** Not Applicable.

**Environmental needs:** Test as Maven Project.

**Intercase Dependencies:** Not Applicable.

**Test Procedures:** Command: “mvn test”

**Input Specification:** Not Applicable.

**Output Specifications:** The city “defaultPlace” should return null and the LocationData should now return 2 cities instead of 3.

**Pass/Fail Criteria:** There should now be 2 cities and the last one should not exist in the data anymore.

**ID#E04 – Gift Test**

**Description:** Test the functionality of the effect of the Gift Event.

**Items covered by this test:** Item ID#E – Event Classes

**Requirements addressed by this test:** Not Applicable.

**Environmental needs:** Test as Maven Project.

**Intercase Dependencies:** Not Applicable.

**Test Procedures:** Command: “mvn test”

**Input Specification:** ints = {0,1,2,10}

**Output Specifications:**

|  |  |  |
| --- | --- | --- |
| **Input Number of Gifts** | **Gold Before** | **Gold After** |
| 0 | 10.0 | 10.0 |
| 1 | 10.0 | 20.0 |
| 2 | 10.0 | 30.0 |
| 10 | 10.0 | 110.0 |

Table 10 - Gift Test

**Pass/Fail Criteria:** The input decides how many times the event is done. This must translate to 10.0 extra gold per event.

**ID#E05 – Movement Test**

**Description:** Test the functionality of the effect of Events that affect player movement.

**Items covered by this test:** Item ID#E – Event Classes

**Requirements addressed by this test:** Not Applicable.

**Environmental needs:** Test as Maven Project.

**Intercase Dependencies:** Not Applicable.

**Test Procedures:** Command: “mvn test”

**Input Specification:** Event to be put into effect.

**Output Specifications:**

|  |  |  |
| --- | --- | --- |
| **Event** | **Actions Before** | **Actions After** |
| LocalSandStorm | 3 | 1 |
| LocalBrokenWagon | 3 | 0 |

Table 11 - Movement Test

**Pass/Fail Criteria:** The number of available action points must be at 1 after a Sandstorm event and at 0 after a Broken Wagon event.

## Test Results

**ID#T01 - Constructor Test**

**Date(s) of Execution:** Automated Tests were run once a week on Fridays at the end of each sprint.

**Staff conducting tests:** Christopher Khoshaba, Joel Koreth, Christian Ramirez Gomez, Will Davidson

**Expected Results:** Expected results were stated in the Test Specifications.

**Actual Results:** Results matched expected output.

**Test Status:** Pass.

**ID#T02 - Action Test**

**Date(s) of Execution:** Automated Tests were run once a week on Fridays at the end of each sprint.

**Staff conducting tests:** Christopher Khoshaba, Joel Koreth, Christian Ramirez Gomez, Will Davidson

**Expected Results:** Expected results were stated in the Test Specifications.

**Actual Results:** Results matched expected output.

**Test Status:** Pass.

**ID#T03 - Buy and Sell Test**

**Date(s) of Execution:** Automated Tests were run once a week on Fridays at the end of each sprint.

**Staff conducting tests:** Christopher Khoshaba, Joel Koreth, Christian Ramirez Gomez, Will Davidson

**Expected Results:** Expected results were stated in the Test Specifications.

**Actual Results:** Results matched expected output.

**Test Status:** Pass.

**ID#T04 - Ability to Buy and Sell Test**

**Date(s) of Execution:** Automated Tests were run once a week on Fridays at the end of each sprint.

**Staff conducting tests:** Christopher Khoshaba, Joel Koreth, Christian Ramirez Gomez, Will Davidson

**Expected Results:** Expected results were stated in the Test Specifications.

**Actual Results:** Results matched expected output.

**Test Status:** Pass.

**ID#T05 - Build Post Test**

**Date(s) of Execution:** Automated Tests were run once a week on Fridays at the end of each sprint.

**Staff conducting tests:** Christopher Khoshaba, Joel Koreth, Christian Ramirez Gomez, Will Davidson

**Expected Results:** Expected results were stated in the Test Specifications.

**Actual Results:** Results matched expected output.

**Test Status:** Pass.

**ID#T06 - Starting Post Bonus Test**

**Date(s) of Execution:** Automated Tests were run once a week on Fridays at the end of each sprint.

**Staff conducting tests:** Christopher Khoshaba, Joel Koreth, Christian Ramirez Gomez, Will Davidson

**Expected Results:** Expected results were stated in the Test Specifications.

**Actual Results:** Results matched expected output.

**Test Status:** Pass.

**ID#T07 - Starting Money Bonus Test**

**Date(s) of Execution:** Automated Tests were run once a week on Fridays at the end of each sprint.

**Staff conducting tests:** Christopher Khoshaba, Joel Koreth, Christian Ramirez Gomez, Will Davidson

**Expected Results:** Expected results were stated in the Test Specifications.

**Actual Results:** Results matched expected output.

**Test Status:** Pass.

**ID#T08 - Starting Sell Bonus Test**

**Date(s) of Execution:** Automated Tests were run once a week on Fridays at the end of each sprint.

**Staff conducting tests:** Christopher Khoshaba, Joel Koreth, Christian Ramirez Gomez, Will Davidson

**Expected Results:** Expected results were stated in the Test Specifications.

**Actual Results:** Results matched expected output.

**Test Status:** Pass.

**ID#C01 – Luck Test**

**Date(s) of Execution:** Automated Tests were run once a week on Fridays at the end of each sprint.

**Staff conducting tests:** Christopher Khoshaba, Joel Koreth, Christian Ramirez Gomez, Will Davidson

**Expected Results:** Expected results were stated in the Test Specifications.

**Actual Results:** Results matched expected output.

**Test Status:** Pass.

**ID#C02 – Fight Results Test**

**Date(s) of Execution:** Automated Tests were run once a week on Fridays at the end of each sprint.

**Staff conducting tests:** Christopher Khoshaba, Joel Koreth, Christian Ramirez Gomez, Will Davidson

**Expected Results:** Expected results were stated in the Test Specifications.

**Actual Results:** Results matched expected output.

**Test Status:** Pass.

**ID#E01 – Drought Test**

**Date(s) of Execution:** Automated Tests were run once a week on Fridays at the end of each sprint.

**Staff conducting tests:** Christopher Khoshaba, Joel Koreth, Christian Ramirez Gomez, Will Davidson

**Expected Results:** Expected results were stated in the Test Specifications.

**Actual Results:** Results matched expected output.

**Test Status:** Pass.

**ID#E02 – Harvest Test**

**Date(s) of Execution:** Automated Tests were run once a week on Fridays at the end of each sprint.

**Staff conducting tests:** Christopher Khoshaba, Joel Koreth, Christian Ramirez Gomez, Will Davidson

**Expected Results:** Expected results were stated in the Test Specifications.

**Actual Results:** Results matched expected output.

**Test Status:** Pass.

**ID#E03 – Destroyed City Test**

**Date(s) of Execution:** Automated Tests were run once a week on Fridays at the end of each sprint.

**Staff conducting tests:** Christopher Khoshaba, Joel Koreth, Christian Ramirez Gomez, Will Davidson

**Expected Results:** Expected results were stated in the Test Specifications.

**Actual Results:** Results matched expected output.

**Test Status:** Pass.

**ID#E04 – Gift Test**

**Date(s) of Execution:** Automated Tests were run once a week on Fridays at the end of each sprint.

**Staff conducting tests:** Christopher Khoshaba, Joel Koreth, Christian Ramirez Gomez, Will Davidson

**Expected Results:** Expected results were stated in the Test Specifications.

**Actual Results:** Results matched expected output.

**Test Status:** Pass.

**ID#E05 – Movement Test**

**Date(s) of Execution:** Automated Tests were run once a week on Fridays at the end of each sprint.

**Staff conducting tests:** Christopher Khoshaba, Joel Koreth, Christian Ramirez Gomez, Will Davidson

**Expected Results:** Expected results were stated in the Test Specifications.

**Actual Results:** Results matched expected output.

**Test Status:** Pass.

## Regression Testing

The above tests should all be repeated as new code is being added to the program. This is because these tests are verifying the basic functions of the game. Therefore, they must always keep working as more code is added. The are all run together with the command “mvn test” and take a few seconds to complete.

# Inspection

## Items to be Inspected

**Item ID#T – Trader Class**

**Description:** Look at the main functions of a trader.

**Item ID#L – Location Data Class**

**Description:** Look at the class that loads the JSON file.

**Item ID#M – Map Movement Logic Classes**

**Description:** Look at map movement code

**Item ID#G – Silk Road Game Class**

**Description:** Look at the class that holds everything for the game logic.

## Inspection Procedures

Inspection Checklist -

1. Class Scope
2. Unit Test Coverage
3. Library Usage – Security and Performance

Inspections are to be done in a mix of during meetings and in person. The primary focus is on members of the team to inspect their classes and have the results be brought up in meetings for any questions with other members and discussions on design choices. 2 meetings were held in this manner. The

## Inspection Results

|  |  |
| --- | --- |
| Class Scope | All tested classes were sufficiently encapsulated in their scope and did not have any overreach. |
| Unit Test Coverage | Trader Class and Location Data Class have sufficient unit test coverage. Map Movement Class and Silk Road Game Class have some coverage. |
| Library Usage – Security and Performance | Libraries have been included through which allows the team to update libraries easily.   * Json-simple library hasn’t been updated in a almost a decade. This can present in security issues and can be relatively slow to newer JSON parsing libraries. * JUNIT library version is outdated. * OpenJFX library version is outdated. |

## Recommendations and Conclusions

Unit test coverage for some classes such as Map Movement Logic Classes and Silk Road Game Class were found to be lacking and certain sections should be abstracted away to facilitate this. The json-simple library used in Location Data class is concerning for security reasons as the library hasn’t been updated in almost a decade and JSON parsers have been shown to be vulnerable to various attacks. There needs to be more care in keeping libraries up to date and dealing with possible regressions because of this.

# Project Issues

## Open Issues

* + - 1. The game is meant to have an end, but it is still undetermined how the game will end.
      2. Method of content distribution of the software is still undetermined. Current platforms under consideration include Steam and Good old Games.

## Waiting Room

A big part of the project that should be included for the future is the ability to play against another person or in single player have a computer AI to play against. The original intention was for this to be part of the game but there was not enough time to implement this. For this feature the game would consider several different statistics for each player (the computer being counted as a player) and compare them at the end to determine a winner. These stats could be amount of gold, diversity of items, number of items, and so on. This would take time to implement and test so it should be saved for a future release.

Another feature to implement later would be the ability to hire mercenaries to work for you and be able to build your own trading posts in certain locations. For this feature the mercenaries would be able to help with certain random encounters and maybe could be used for interactions with other players. For the trading posts it would be a good way for the player to earn additional gold and gain certain resources. This might take some time to accomplish so it can be saved for a future release.

Being able to change the attributes for the player was another part of the game to be implemented for the future. This would make certain encounters more engaging since there’s more consequences this way for the player’s actions. As of now the player’s attributes are equal to all the bandits, they encounter so there’s no consequences as of now. The player could use gold to add to certain attributes they would want to increase. The ability to include this in the game is already present so this should be implemented by the next release.

The last implementation the project needs is for the inclusion of global events. These would be random events that would go so far as to affect the map for the player or maybe even some of the empires. This was very close to being done so this should be able to be implemented by the next release.

## Ideas for Solutions

As mentioned above, the game currently has not method to complete. Multiple ideas were brainstormed for this problem. For example, players could gain points depending on how many achievements they earned in the game. Each achievement would have a different point value that can be awarded at the end to the player to see how well they performed. This would mean a set number of turns would need to be enforced. Another method to win the game would depend on the amount of gold the Trader earned. Both methods could be used to compare score on a high score list.

The second issue that had to do with the method of releasing the game could be solved different ways. Another platform such as Steam could be used to distribute it. This could be helpful in allowing players to find it, but it also means the platform would need a cut of the profit. Another solution could be to just sell in an independent website, but this would require more work for the marketing team.

## Project Retrospective

*Content*

We were able to conduct meetings at least once a week or more. These meetings were invaluable in planning our sprints, checking our progress, and for guiding each other. There were some difficulties in the beginning with setting up each team member’s environment regarding getting the right JavaFX version and Java IDE. It was more work to guide each other asynchronously for these issues.

*Considerations*

There were certain bottlenecks in development, especially in the early stages. There were points where one part of the project was necessary to be completed in order for other team members to progress. As a result of this, there were some slowdowns while things like map movement was defined. This didn’t mesh well with JIRA as tickets were stuck in progress till completion.

# Glossary

Silk Route: Used to refer to the Silk Route game. Unless otherwise stated, this is the default meaning of the term, and in other cases, the meaning will be provided through context.

RNG: Random Number Generator. Used to refer to randomness within the game.

Strategy: Used to refer to how a player makes decisions when presented with choice within the game.

Historical: Meant to refer to a theme, not necessarily historical accuracy.

# References / Bibliography

|  |  |
| --- | --- |
| [1] | J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012. |
| [2] | A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013. |
| [3] | Robertson and Robertson, Mastering the Requirements Process. |
| [4] | M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004. |
| [5] | S. Khehra, B. Habtegiorgis, I. Patra and D. Moren, "The Silk Route Adventures Project Report," 2015. |
| [6] | "Implementation of Hex Grids," [Online]. Available: https://www.redblobgames.com/grids/hexagons/implementation.html. |
| [7] | L. Gupta, "Java read json and write json example – JSON.simple," [Online]. Available: https://howtodoinjava.com/java/library/json-simple-read-write-json-examples/. |

# Index

Glossary 28

Index 28

Inspection 25

List of Figures 4

List of Tables 5

Project Deliverables 7

Project Description 6

Project Issues 26

References / Bibliography 28

Table of Contents 2

Testing 11