***Testing and Inspection Report***

**

***Testing and Inspection for “History Arcade”***

***Prepared by:***

***Michael Lemus***

***Daniel Rios***

***Muhammad Zeeshan***

**Table of Contents**

[List of Figures 4](#_Toc38668180)

[I Project Description 5](#_Toc38668181)

[1 Project Overview 5](#_Toc38668182)

[2 Project Domain 5](#_Toc38668183)

[3 Relationship to Other Documents 5](#_Toc38668184)

[4 Naming Conventions and Definitions 6](#_Toc38668185)

[4a Definitions of Key Terms 6](#_Toc38668186)

[4b UML and Other Notation Used in This Document 6](#_Toc38668187)

[4c Data Dictionary for Any Included Models 6](#_Toc38668188)

[II Testing 8](#_Toc38668189)

[5 Items to be Tested 8](#_Toc38668190)

[6 Test Specifications 9](#_Toc38668191)

[7 Test Results 20](#_Toc38668192)

[8 Regression Testing 24](#_Toc38668193)

[III Inspection 24](#_Toc38668194)

[9 Items to be Inspected 24](#_Toc38668195)

[10 Inspection Procedures 25](#_Toc38668196)

[11 Inspection Results 26](#_Toc38668197)

[IV Recommendations and Conclusions 29](#_Toc38668198)

[V Project Issues 30](#_Toc38668199)

[12 Open Issues 30](#_Toc38668200)

[13 Waiting Room 30](#_Toc38668201)

[14 Ideas for Solutions 30](#_Toc38668202)

[15 Project Retrospective 31](#_Toc38668203)

[VI Glossary 32](#_Toc38668204)

[VII References / Bibliography 32](#_Toc38668205)

[VIII Index 33](#_Toc38668206)

[IX Appendix 34](#_Toc38668207)

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

[Figure 1: Class Diagram 7](#_Toc38668061)

[Figure 2: Class Diagram: Inheritance 8](#_Toc38668062)

[Figure 3: Java Checklist (Level 1 Inspection) 34](#_Toc38668063)

**List of Tables**

[Table 1: Java Checklist 35](#_Toc38668001)

# Project Description

## Project Overview

History arcade is a turn-based quiz game that can be played against a human or an AI. The questions are linked to the region that the player is in in order to obtain the territories and collect points. This game is interactive and should engage the user on challenging questions and opponents

This game was developed using javafx and is run as a java project. User join as an anonymous user and is able to play against an ai (single player) or play against another friend (multiplayer). To begin the game, the user selects the number of players, number of rounds (optional or randomly set), and the map they want to play. The locations are randomly set for the players with player 1 always starting first. Players are free to wander the map, even territories that are owned by other players. If the territory is owned by the same player, a turn is lost, essentially nothing will change. Each country is worth 1 point the game is driven by a timer in order to avoid long waiting time. The game ends once the set number of rounds has passed.

## Project Domain

The domain of the project is entertainment and education. This game is meant for classroom-like settings where it can be used as a tool to facilitate learning of important historical topics. Therefore, the features most important to the game’s success are historical accuracy and its entertainment value.

That being said, the information presented in the game environment should be as accurate as possible to ensure that people playing the game learn information that is true. The map the gameplay takes place on should be historically accurate for the context of the event being learned about.

Secondly, the game should also be fun for students in order to maintain their interest while communicating factual information about particular events in history. The game should be engaging enough to be entertaining while in a classroom setting. The engagement factor will come from the competitive aspect of the gameplay.

## Relationship to Other Documents

**group28FinalReport:** Previous group’s documentation of the conceptualized/suggested design of History Arcade that our group based our project on.

**History Arcade Scenarios 1-2:** Descriptions of features that were objectives for the group to accomplish or strive to complete. Features were based on the ideas suggested by the previous group.

**iceScrum Task Board:** Contains logs of what work was accomplished during particular sprints and who had done the work.

## Naming Conventions and Definitions

### Definitions of Key Terms

**Round:** A round is completed once every player in the game has taken a turn.

**PC:** Computing device that requires a wired or Wi-Fi connection to the internet (Desktops and laptops with any operating system.)

**Quiz:** The set of topical trivia questions that are linked with each individual map.

**Trivia:** Questions related to a particular place and time of historical significance.

**Map:** Map represents the geographical area which player selects while creating a new game. It is the ‘game board’ of the game.

**Conquer:** When a player answers a question correctly and occupies some part of the map, then he is said to have conquered a territory and gains a point.

**Territory:** Regions that characters in the game can navigate to on the map and claim.

### UML and Other Notation Used in This Document

This document generally follows the Version 2.0 OMG UML standard, as described by Fowler in [4].

### Data Dictionary for Any Included Models

**Map files** are stored as json files containing specifications on how to generate a particular map.  
**Locations** are territories contained in the map to be generated.  
**Connections** are a list that specifies the allowable moves between territories in the map.  
**Trivia** is a list of questions and answers tied to specific Locations on the map.  
An example of how map files’ contents are specified is shown below:

|  |
| --- |
| {  "Locations”: [  {  "Region”: "String",  "ID”: int  }  ],    "Connections”: [  {  "fromRegionID”: int,  "fromRegionName”: "String",  "toRegionID”: int,  "toRegionName”: "String"  }  ],    "Trivia”: [  {  "Question”: "String",  "Answer”: "String",  "RegionID”: int  }  ]  }    Figure 1: Class Diagram    Figure 2: Class Diagram: Inheritance |

# Testing

## Items to be Tested

1. Reading number of players
2. Reading number of rounds
   1. Strings and within limits
3. Creating a multiplayer game
   1. Ip number as string
   2. Port number as an integer ONLY.
4. Joining a multiplayer game
   1. Ip number as string
   2. Port number as an integer ONLY.
5. Choosing a map
6. Selecting the correct answer/vs not selecting the answer
7. Ensuring Timer works correctly
   1. When it reaches 0 sends a message
   2. Font changes when less than 5 seconds
   3. Resets font after each round
   4. Disabling submit option
8. Quit buttons work correctly
9. Questions are linked to region
10. Player movements
    1. Correct placement of the character

## Test Specifications

**ID#1T - Number of players 1**

**Description:** Ensure the system recognizes the number of players selected (1).

**Items covered by this test:** Section 5Item 1

**Requirements addressed by this test:**  Ensuring single player functionality.

**Environmental needs:** A java ide (IntelliJ/Eclipse), json-simple-1.1.jar, AnimateFX-1.2.1.jar

**Intercase Dependencies:**  The server has started on 5555. Map selection was made.

**Test Procedures:**

1. Start up the IDE (Eclipse/IntelliJ)

2. Start Application

3. Select the number of players (1)

4. Select the map

5. Click Start Game

**Input Specification:**

1. Select the number of players (1)

2. Select the map to play in.

**Output Specifications:**

Game should display with the correct map. Game should also be playing against *Glados* the AI which can be seen after the first player (you) have made your first move.

**Pass/Fail Criteria:**

The game successfully started to play against Glados after the move the first player made. If the game errors out or expects another player to join, the test fails.

**2T - Number of players 2**

**Description:**  Ensure the system recognizes the number of players selected (2).

**Items covered by this test:** Section 5 Item 1

**Requirements addressed by this test:**  Ensuring multiplayer functionality.

**Environmental needs:** A java ide (IntelliJ/Eclipse), json-simple-1.1.jar, AnimateFX-1.2.1.jar

**Intercase Dependencies:** The server has started on 5555. Map selection was made.

**Test Procedures:**

1. Start up the IDE (Eclipse/IntelliJ)

2. Start Application

3. Select the number of players (2)

4.Select the map

5. Click Start Game

**Input Specification:**

1. Select the number of players (2)

2. Select the map to play in.

**Output Specifications:**

Game should be displayed with the correct map but should tell the user that they are waiting for the other player to join.

**Pass/Fail Criteria:**

The game successfully started and is waiting for the second player to join. Otherwise it fails.

**3T - Number of Rounds**

**Description:** System can recognize an INTEGER as round input.

**Items covered by this test:** Section 5 item 2

**Requirements addressed by this test:**  Ensuring round input optionality.

**Environmental needs:** A java ide (IntelliJ/Eclipse), json-simple-1.1.jar, AnimateFX-1.2.1.jar

**Intercase Dependencies:** The server has started on 5555. Map selection was made.

**Test Procedures:**

1. Start up the IDE (Eclipse/IntelliJ)

2. Start Application

3. Select the number of players (1)

4. Input a set number of rounds as an integer

5.Select the map

6. Click Start Game

**Input Specification:**

1. Select the number of players (2)

2. Select the map to play in.

3. Input the number of rounds to input (will only recognize integers)

**Output Specifications:**

Game should be displayed with the correct map. The game will then play with the input number of rounds against Glados.

**Pass/Fail Criteria:**

The game successfully starts and plays with the specified number of rounds. Otherwise, it fails.

**4T - Creating a multiplayer game**

**Description:** A player can successfully create a multiplayer game that another player can join.

**Items covered by this test:** Section 5 Item 3

**Requirements addressed by this test:**  Ensuring multiplayer functionality.

**Environmental needs:** A java ide (IntelliJ/Eclipse), json-simple-1.1.jar, AnimateFX-1.2.1.jar

**Intercase Dependencies:** The server has started on 5555. Map Selection was made

**Test Procedures:**

1. Start up the IDE (Eclipse/IntelliJ)

2. Start Application

3. Select the number of players (2)

4. Input a set number of rounds as an integer

5. Select the map

6. Input port number

7. Input IP Address

8. Click Create Game

**Input Specification:**

1. Select the number of players (2)

2. Select the map to play in.

**Output Specifications:**

Game should be displayed with the correct map but should tell the user that they are waiting for the other player to join. After the second player joins, the game should inform the player that the game can now start.

**Pass/Fail Criteria:**

The game successfully started and is waiting for the second player to join. Otherwise, it fails.

**5T - Joining a multiplayer move**

**Description:** A player can successfully join a multiplayer game that another player created.

**Items covered by this test:** Section 5 Item 4

**Requirements addressed by this test:**  Ensuring multiplayer functionality.

**Environmental needs:** A java ide (IntelliJ/Eclipse), json-simple-1.1.jar, AnimateFX-1.2.1.jar

**Intercase Dependencies:** The server has started on 5555. Multiplayer game was created.

**Test Procedures:**

1. Start up the IDE (Eclipse/IntelliJ)

2. Start Application

3. Select the number of players (2)

4. Input a set number of rounds as an integer

5. Select the map

6. Input port number

7. Input IP Address

8. Click join Game

**Input Specification:**

1. Select the number of players (2)

2. Select the map to play in.

**Output Specifications:**

Game should be displayed with the correct map and since the second player has joined, the game shall now begin with the first player being able to make their first move.

**Pass/Fail Criteria:** The game successfully started and is now proceeding with the first player being able to make their move. Otherwise, it fails.

**6T - Choosing a map**

**Description:** If the player selects the map, the game shall display and use the map.

**Items covered by this test:** Section 5 Item 5

**Requirements addressed by this test:** Ensuring map optionality.

**Environmental needs:** A java ide (IntelliJ/Eclipse), json-simple-1.1.jar, AnimateFX-1.2.1.jar

**Intercase Dependencies:** The server has started on 5555.Number of players was chosen.

**Test Procedures:**

1. Start up the IDE (Eclipse/IntelliJ)

2. Start Application

3. Select the number of players (1 or 2)

4. Select the map (any map)

5. Start the game

**Input Specification:**

1. Select the number of players (1)

2. Select the map to play in.

**Output Specifications:**

Game should be displayed with the correct map. The game has started and is either waiting for another player or is playing against Glados.

**Pass/Fail Criteria:**

The game successfully started, and the correct map is shown. Otherwise, it fails.

**7T - Selecting the correct answer vs not selecting the correct answer**

**Description:** The system (server) should evaluate the clients input and inform the user if they got the correct answer or not.

**Items covered by this test**: Section 5 Item 6

**Requirements addressed by this test:** Main functionality of the game; ensuring main game functionality.

**Environmental needs:** A java ide (IntelliJ/Eclipse), json-simple-1.1.jar, AnimateFX-1.2.1.jar

**Intercase Dependencies:** The server has started on 5555. Number of players have been selected. The game has begun

**Test Procedures:**

1. Start up the IDE (Eclipse/IntelliJ)

2. Start Application

3. Select the number of players (1)

4. Select the map (any map)

5. Start the game

6. Follow directions provided from server.

7. When asked to answer, provide the correct answer/ or wrong answer and check for results.

**Input Specification:**

1. Select the number of players (1)

2. Select the map to play in.

**Output Specifications:** The system should inform the user of their response and whether or not the answer provided was correct or not**.**

**Pass/Fail Criteria:** The system should always choose the correct answer and inform the user if they got it right or wrong. Otherwise, it fails,

**8T - Ensuring timer works correctly**

**Description:** The timer should work as described: if the time left is less than 5 seconds it should emphasize that to the user, etc. (See Section 5 Item 7 for more information).

**Items covered by this test:** Section 5 Item 7

**Requirements addressed by this test:** Having the game run continuously, increasing player engagement.

**Environmental needs:** A java ide (IntelliJ/Eclipse), json-simple-1.1.jar, AnimateFX-1.2.1.jar

**Intercase Dependencies:** The server has started on 5555. Number of players have been selected. The game has begun

**Test Procedures:**

1. Start up the IDE (Eclipse/IntelliJ)

2. Start Application

3. Select the number of players (1)

4. Select the map (any map)

5. Start the game

6. Follow directions provided from server.

7. When asked to answer, wait, and check the Timer and see if all of the requirements are met. (See above).

**Input Specification:**

1. Select the number of players (1)

2. Select the map to play in.

**Output Specifications:** The timer correctly works as explained.

**Pass/Fail Criteria:** The timer should always restart after every question and work as described. Otherwise it fails.

**9T - Quit Buttons work correctly**

**Description:** The quit button on the main menu closes the application but the one on the main game screen returns the user to the main menu.

**Items covered by this test:** Section 5 Item 8

**Requirements addressed by this test:** NA

**Environmental needs:** A java ide (IntelliJ/Eclipse), json-simple-1.1.jar, AnimateFX-1.2.1.jar

**Intercase Dependencies:** The server has started on 5555. Number of players have been selected. The game has begun. Or if on main menu screen, you are able to proceed without doing anything.

**Test Procedures:**

1. Start up the IDE (Eclipse/IntelliJ)

2. Start Application

3. Select the number of players (1 or 2)

4. Select the map (any map)

5. Start the game

6. Click Quit game. (Make sure the screen shown is the main menu screen after that).

7. Once in the main menu, click quit game. (This should close the application),

**Input Specification:**

1. Select the number of players (1)

2. Select the map to play in.

**Output Specifications:** The quit button on the main game screen returns you to the main menu while the quit button on the main menu screen exits the application.

**Pass/Fail Criteria:** The quit button works as described while not erroring out some other part of the code. Otherwise, it fails.

**10T - Questions are linked to region**

**Description:** On every map, the questions linked to the characters movement should be tied to a specific question to avoid repetitive questions.

**Items covered by this test:** Section 5 Item 9

**Requirements addressed by this test:** The system should provide questions to the users in order to conquer territories.

**Environmental needs:** A java ide (IntelliJ/Eclipse), json-simple-1.1.jar, AnimateFX-1.2.1.jar

**Intercase Dependencies:** The server has started on 5555. Number of players have been selected. The game has begun.

**Test Procedures:**

1. Start up the IDE (Eclipse/IntelliJ)

2. Start Application

3. Select the number of players (1 or 2)

4. Select the map (WWI/WWII)

5. Start the game

6. Play a couple of rounds to ensure there are different questions in each region.

7. Repeat for WWII.

**Input Specification:**

1. Select the number of players (1)

2. Select the map to play in.

**Output Specifications:**  The system should display the correct map and retrieve questions from the region that they are currently in.

**Pass/Fail Criteria:** The questions are linked to the specific region and the time period (WWI/WWII). Otherwise it fails.

**11T - Player movements**

**Description:** The system recognizes the messages received from the server and moves the characters accordingly.

**Items covered by this test:** Section 5 Item 10

**Requirements addressed by this test:** System engages user by displaying character moves.

**Environmental needs:** A java ide (IntelliJ/Eclipse), json-simple-1.1.jar, AnimateFX-1.2.1.jar

**Intercase Dependencies:** The server has started on 5555. Number of players have been selected. The game has begun.

**Test Procedures:**

1. Start up the IDE (Eclipse/IntelliJ)

2. Start Application

3. Select the number of players (1)

4. Select the map (WWI/WWII)

5. Start the game

6. Play a couple of rounds to ensure that if you provide a location, your character moves.

7. Repeat for WWII.

**Input Specification:**

1. Select the number of players (1)

2. Select the map to play in.

**Output Specifications:** The system should update the characters locations accordingly. If the player chooses not to move, the player will remain in the same spot.

**Pass/Fail Criteria:** If the characters movement is not recognized by the system when they input the correct input, the test will fail. If the system is able to recognize users input and move the character accordingly, the test passes.

## Test Results

**1TT - Number of Players 1**

**Date(s) of Execution:** 4/23/20

**Staff conducting tests:** Muhammad Zeeshan

**Expected Results:**  Game should be displayed with the correct map. The game will then play against Glados.

**Actual Results:** Game was tested with WWI map. The game displayed the WWI map. Then, proceeded to make sure the game ran for a single player against the AI Glados.

**Test Status:**  Pass

**2TT - Number of Players 2**

**Date(s) of Execution:** 4/23/20

**Staff conducting tests:** Muhammad Zeeshan

**Expected Results:**  Game should be displayed with the correct map. The game will then wait for a second player. Once the second player has joined, the game will start and indicate the player’s turn.

**Actual Results:** Game was tested with WWI map. The game displayed the WWI map. Then, proceeded to make sure the game waited for another player. Once a new player had joined the same server, the game proceeded to run with Player 1 having the first turn.

**Test Status:**  Pass

**3TT - Number of Rounds**

**Date(s) of Execution:** 4/22/20

**Staff conducting tests:** Michael Lemus

**Expected Results:**  Game should be displayed with the correct map.The game will then play with the input number of rounds against Glados.

**Actual Results:** Game was tested with WWI map. The game displayed the WWI map. Then, proceeded to make sure the game ran for the specified amount of rounds.

**Test Status:**  Pass

**4TT - Creating a multiplayer game**

**Date(s) of Execution:** 4/22/20

**Staff conducting tests:** Michael Lemus

**Expected Results:** Game should be displayed with the correct map but should tell the user that they are waiting for the other player to join.

**Actual Results:** The game successfully started and is waiting for the second player to join.

**Test Status:** Pass

**5TT - Joining a multiplayer game**

**Date(s) of Execution:**4/22/20

**Staff conducting tests:** Michael Lemus

**Expected Results:**  Game should be displayed with the correct map and since the second player has joined, the game shall now begin with the first player being able to make their first move.

**Actual Results:**  The game successfully started and was followed by the first player being able to make their move.

**Test Status:** Pass

**6TT1 - Choosing a map WWI**

**Date(s) of Execution:** 4/22/20.

**Staff conducting tests:** Michael Lemus

**Expected Results:** Game should be displayed with the WWI map. The game has started and is either waiting for another player or is playing against Glados.

**Actual Results:**  The game successfully started and the correct map is shown.

**Test Status:** Pass

**6TT2 - Choosing a map WWII**

**Date(s) of Execution:** 4/22/20.

**Staff conducting tests:** Michael Lemus

**Expected Results:** Game should be displayed with the WWII map. The game has started and is either waiting for another player or is playing against Glados.

**Actual Results:**  The game successfully started and the correct map is shown.

**Test Status:** Pass

**6TT3 - Choosing a map Civil War**

**Date(s) of Execution:** 4/22/20.

**Staff conducting tests:** Michael Lemus

**Expected Results:** Game should be displayed with the Civil War map. The game has started and is either waiting for another player or is playing against Glados.

**Actual Results:**  The game successfully started and the correct map is shown.20

**Test Status:** Pass

**7TT - Selecting the correct answer vs not selecting the correct answer**

**Date(s) of Execution:** 4/23/20

**Staff conducting tests:** Muhammad Zeeshan

**Expected Results:**  If the correct answer is inputted, the player receives a point on the scoreboard and the output window tells the players that the territory has been captured. If an incorrect answer is selected, no territory is captured and the output window tells the player that the answer is false.

**Actual Results:** Game was tested with WWI map. The game displayed the WWI map and asked a question linked to the region. Once the question was answered correctly, the output window said that the region was captured and the scoreboard reflected the correct score. When I answered incorrectly, the output window told me the answer was false.

**Test Status:**  Pass

**8TT - Ensuring timer works correctly**

**Date(s) of Execution:** 4/23/20

**Staff conducting tests:** Muhammad Zeeshan

**Expected Results:**  If the time left is less than 5 seconds it should emphasize that to the user by increasing in size and pulsing.

**Actual Results:** Game was tested with WWI map. The game displayed the WWI map and asked a question linked to the region. When 5 seconds were left, the timer grew in size and pulsed in animation to signify that the time was running out.

**Test Status:**  Pass

**9TT - Quit Buttons work correctly**

**Date(s) of Execution:** 4/23/20

**Staff conducting tests:** Muhammad Zeeshan

**Expected Results:**  The quit button on the main menu closes the application but the one on the main game screen returns the user to the main menu.

**Actual Results:** Game was tested with WWI map. The game displayed the WWI map and asked a question linked to the region. When the quit button was clicked on the game screen, the user was brought back to the main menu. When the quit button was clicked on the main menu, the application exited successfully.

**Test Status:**  Pass

**10TT - Questions are linked to region**

**Date(s) of Execution:** 4/23/20

**Staff conducting tests:** Daniel Rios

**Expected Results:**  The question received on the game screen should be about the region that the user is occupying on the selected map.

**Actual Results:** Game was tested with WWI map. The game displayed the WWI map and asked a question linked to the region. The region selected was Italy and the question was “What side did Italy fight on in WWI?” This question pertains to Italy and is therefore linked to the correct region.

**Test Status:**  Pass

**11TT - Player movements**

**Date(s) of Execution:** 4/23/20

**Staff conducting tests:** Daniel Rios

**Expected Results:**  If the user selects a country to move to, the avatar is placed on that country in the map and the output window outputs that the player has moved to that country.

**Actual Results:** Game was tested with WWI map. The game displayed the WWI map and asked a question linked to the region. After answering the question, the user selected to move to Russia. The avatar was then placed on Russia in the map and the output window said the following: “Player 1 has moved to Russia.”

**Test Status:**  Pass

## Regression Testing

Not Applicable

# Inspection

## Items to be Inspected

Item being inspected: Round number input box

Item being inspected: Player positioning functions

Item being inspected: Location.java

## Inspection Procedures

Item being inspected: Round number input

Meetings: One meeting where the results were discussed electronically.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Yes | No | Comment |
| Check that it properly responds to invalid inputs (Words) |  |  |  |
| Check that it properly responds to invalid inputs (Zero) |  |  |  |
| Check that it properly responds to invalid inputs (Negative Numbers) |  |  |  |
| Check that it properly responds to invalid inputs (Beyond Integer Range) |  |  |  |
| Check that it accepts positive integers between 2 and 100 |  |  |  |
| Defaults to 5 if no input is given |  |  |  |

Item being inspected: Player positioning functions

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Yes | No | Comment |
| JavaDoc comments present and clearly explain parameters, returns, and method purpose |  |  |  |
| Player Icons are positioned above their respective country when a player is in that region |  |  |  |
| Method name is meaningful |  |  |  |
| Method visibility is appropriate for its usage |  |  |  |

Item being inspected: Location.java

Checklist available in Appendix, made by Pezze & Young [6]

## Inspection Results

Item being inspected: Round number input

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Yes | No | Comment |
| Check that it properly responds to invalid inputs (Words) | ✔ |  | Outputs an error message and allows the user to correct their mistake. |
| Check that it properly responds to invalid inputs (Zero) | ✔ |  | Outputs an error message and notifies the user that the number of rounds should be between 2 and 100. |
| Check that it properly responds to invalid inputs (Negative Numbers) |  | ✔ | Outputs an error message and notifies the user that the number of rounds should be between 2 and 100 but proceeded to main game screen. \*Resolved-Check below for details |
| Check that it properly responds to invalid inputs (Beyond Integer Range) |  | ✔ | Outputs an error message and notifies the user that the number of rounds should be between 2 and 100 but proceeded to main game screen. \*Resolved-Check below for details |
| Check that it accepts positive integers between 2 and 100 | ✔ |  | System successfully recognized integer numbers and proceeded to play the game for the corresponding number of rounds. |
| Defaults to 5 if no input is given | ✔ |  | Game was run and it lasted for 5 rounds. |

Item Inspected: Round number input

**Who did the inspection**: Michael, Daniel

**Time/Date:** 4/24/20 - 6pm

**What was discovered:** A couple bugs involving acceptable input range

**Resolution:** Muhammad fixed the issues discovered during inspection.

Item being inspected: Player positioning functions

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Yes | No | Comment |
| JavaDoc comments present and clearly explain parameters, returns, and method purpose |  | ✔ | Return is not documented and method purpose is not explained |
| Player Icons are positioned above their respective country when a player is in that region |  | ✔ | Found an issue with methods and Luxembourg region. Player icon would be placed off the map. |
| Method name is meaningful | ✔ |  | Indicates what the methods’ purpose is clearly. |
| Method visibility is appropriate for its usage |  | ✔ | Visibility is public when should be private since they are only being called within the same class file. |

Item Inspected: Player positioning functions

**Who did the inspection:** Daniel, Muhammad

**Time/Date:** 4/23/20 - 8pm

**What was discovered:** Bug caused by a typo in the method and the game map files.

**Resolution:** The bug was fixed by removing the typo in both the method and map files.

|  |  |  |  |
| --- | --- | --- | --- |
| File items: Are the following items included and consistent? | Yes | No | Comments |
| Author and current maintainer identity |  | ✓ | There are no notes where someone can distinguish who authored the document. |
| Cross-reference to design entity |  |  | Not Applicable |
| Overview of package structure, if the class is the principal entry point of a package |  |  | Not Applicable |
| Import Section: Are the following requirements satisfied? | Yes | No | Comments |
| Brief comment on each import with the exception of the standard set: java.io.\*, java.util.\* |  |  | Not Applicable |
| Each imported package corresponds to a dependence in the design documentation |  |  | Not Applicable |
| Class Declaration: Are the following requirements satisfied? | Yes | No | Comments |
| The constructor is explicit (if the class is not static) | ✓ |  | Constructor was modified in order to take in values. |
| The visibility marker matched the design document |  |  | Not Applicable |
| The constructor is explicit(if the class is not static) | ✓ |  | Followed the correct procedure |
| The visibility of the class is consistent with the design document |  |  | Not Applicable |
| Class: Are names compliant with the following rules? | Yes | No | Comments |
| Class or interface:CapitalizedWithEachInternalWordCapitalized | ✓ |  |  |
| Field name:capsAfterFirstWord.  Name must be meaningful outside of context | ✓ |  |  |
| Idiomatic Methods: Are names compliant with the following rules? | Yes | No | Comments |
| Method name: capsAfterFirstWord  Local VAriables: capsAfterFirstWord  Name may be short (e.g., i for integer) if scope of declaration and us is less than 30 lines | ✓ |  | All methods followed the naming convention described. |
| Factory method for x: newX | ✓ |  | All methods followed the naming convention described. |
| Getter for attributeX: getX(); | ✓ |  | All getter methods followed the naming convention described. |
| Setter for attribute X void setX; | ✓ |  | All setter methods followed the naming convention described. |

Item Inspected: Location.java

**Who did the inspection:** Michael, Muhammad

**Time/Date:** 4/22/20 - 5pm

**What was discovered:** There was a failure where the author was not mentioned.

**Resolution:** The name of the author was appended to the code.

# Recommendations and Conclusions

Item being inspected: Round number input box

**Results:** Item did not pass its tests and inspection but issues that were found have been resolved. Further testing and inspection may be required later to ensure no other issues crop up.

Item being inspected: Player positioning functions

**Results:** Item did not pass inspection but the issue found was resolved. Further inspection needed later to check if proper documentation has been added.

Item being inspected: Location.java

**Results:** Item did not pass inspection but the issue was resolved. There should be a continuation of good practices moving forward.

# Project Issues

## Open Issues

The previous group had requirements regarding accounts and multiplayer interactions that our game currently has not addressed. There would need to be a database created for the purposes of storing login data and player statistics. It would have to either be stored on the server running the game or another separate server to handle that information. Possibly the inclusion of a publicly available leaderboard for highest player scores.

There was also a requirement that the game client end be done on browsers using javascript but we took another route with our development of the game. Therefore the question about how the client end of the game will be given to users as a product is still not clear. Some options are developing the client end as a mobile app or a desktop application. Possibly developing it for both types of hardware is something to consider which opens up a discussion about if crossplay will be an option for users.

## Waiting Room

There were some ideas that we would have implemented if we could. For example, we would have added coloring to the maps that correspond to the players if we had better maps. In order to accomplish this, we would need someone to develop a colorable map that is from that time period.  We would have also liked to add a waiting area for players, where players could see leaderboards and invite players from around the world. Another great feature to be added is adding 4 player functionality. Then, players can work as teams or against one another for a better challenge. This would make the game more engaging as they would need a more effective strategy than against one other player.

## Ideas for Solutions

* Separate database server for purposes such as storing player login information and multiplayer rankings
  + Maybe use JDBC library for connecting to and querying an SQL database with the tables for Login and Score information
  + A public webpage to display top scores which is updated every hour with the latest scores would also be nice
* Moving from location to location via clicking on the map
  + Maybe create cleaner maps that are less compressed and have buttons that appear on locations available for the player to move to
* Changing answer input from typing response to multiple choice
  + Possibly a drop down list of options or pop-up window with 4 buttons to be selected from

## Project Retrospective

* Our swap to different UI library midway into project development:

We had some difficulty deciding what library to use for constructing the user interface. Initially, we thought that JavaSwing would be best but we decided to switch to JavaFX instead later. Only one of us was familiar with JavaSwing while two of us were familiar with JavaFX. Also, JavaFX had much a nicer polished look to it. The combination of these two facts made us decide to switch.

However, this switch to JavaFX was a little bit of a setback since we had to build the UI from scratch again. This resulted in a bit of delay with work on other features of the project we wanted to add such as, multiple choice selection inputs.

In hindsight, it might have been better to weigh the merits of each library before selecting one instead of quickly trying to get the project off the ground. It would have resulted in less delays on other tasks needing to be worked on later. Nonetheless, the decision to switch to JavaFX was a good one since the majority of the group knew how to work with it and the UI created was much more polished.

* Our handling of exceptions could have been better:

Throughout our program, we tried to make it a point to catch possible exceptions from code that could encounter problems. However, there were some areas where our catch blocks were catching exceptions that were too general and the response generated by the catch block misled us away from fixing the source of the problem.

This problem resulted in us being unable to locate the source of some errors for extended periods of time. It resulted in wasted time and certain problems going unaddressed.

In future, we should be more specific with the types of exceptions our catch blocks are looking for in order to prevent unexpected problems from being overlooked. Also, it’s generally better practice to use more specific exception types.

# Glossary

**Round:** A round is completed once every player in the game has taken a turn.

**PC:** Computing device that requires a wired or Wi-Fi connection to the internet (Desktops and laptops with any operating system.)

**Quiz:** The set of topical trivia questions that are linked with each individual map.

**Trivia:** Questions related to a particular place and time of historical significance.

**Map:** Map represents the geographical area which player selects while creating a new game. It is the ‘game board’ of the game.

**Conquer:** When a player answers a question correctly and occupies some part of the map, then he is said to have conquered a territory and gains a point.

**Territory:** Regions that characters in the game can navigate to on the map and claim.

# 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] | M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004. |
| [5]  [6] | |  |  | | --- | --- | |  | Group 28: Rajan Bhandari, Shanmathi Krithivasan, Viswas Kuruvalli, Kyle Almryde, Brent Campbell, “Development Report for ‘History Arcade’” | |  | Mauro Pezze, Michal Young, “Software Testing and Analysis: Process, Principles and Techniques,” 1st Edition, John Wiley & Sons Inc., 2008 | |

# Index

Bibliography 3, 27

Conclusions 2, 25

Connections 7

Conquer 7, 27

Dictionary 2, 7

Domain 6

Glossary 2, 27

group28FinalReport 6

iceScrum 6

Inspection 24

Issues 2, 25

Key Terms 2, 7

Locations 7

Map files 7

Overview 6

PC 7, 27

Procedures 2, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 25

Quiz 7, 27

References 3, 27

Regression 2, 24

Results 2, 20, 21, 22, 23, 24, 25

Retrospective 2, 26

Round 7, 27

Scenarios 6

Solutions 2, 26

Specifications 10

Territory 7, 27

Testing 10

Trivia 7, 8, 27

UML 2, 7, 28

# Appendix

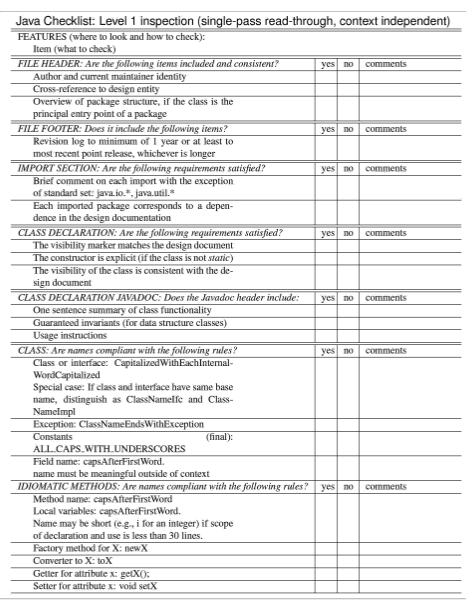
Java Checklist used for inspection in section III:

Figure 3: Java Checklist (Level 1 Inspection)

|  |  |  |  |
| --- | --- | --- | --- |
| File items: Are the following items included and consistent? | Yes | No | Comments |
| Author and current maintainer identity |  |  |  |
| Cross-reference to design entity |  |  |  |
| Overview of package structure, if the class is the principal entry point of a package |  |  |  |
| Import Section: Are the following requirements satisfied? | Yes | No | Comments |
| Brief comment on each import with the exception of the standard set: java.io.\*, java.util.\* |  |  |  |
| Each imported package corresponds to a dependence in the design documentation |  |  |  |
| Class Declaration: Are the following requirements satisfied? | Yes | No | Comments |
| The constructor is explicit (if the class is not static) |  |  |  |
| The visibility marker matched the design document |  |  |  |
| The constructor is explicit(if the class is not static) |  |  |  |
| The visibility of the class is consistent with the design document |  |  |  |
| Class: Are names compliant with the following rules? | Yes | No | Comments |
| Class or interface:CapitalizedWithEachInternalWordCapitalized |  |  |  |
| Field name:capsAfterFirstWord.  Name must be meaningful outside of context |  |  |  |
| Idiomatic Methods: Are names compliant with the following rules? | Yes | No | Comments |
| Method name: capsAfterFirstWord  Local VAriables: capsAfterFirstWord  Name may be short (e.g., i for integer) if scope of declaration and us is less than 30 lines |  |  |  |
| Factory method for x: newX |  |  |  |
| Getter for attributeX: getX(); |  |  |  |
| Setter for attribute X void setX; |  |  |  |

Table 1: Java Checklist