‘XYZ’ System Design and Specification

|  |  |
| --- | --- |
| Team Name / Number: | Team 12 |
| Team Members: | Sadie Coffield, Annie Traill, Iain Mckenzie, Bernard Hoeger Jerez |

# Introduction / Problem Statement

What we plan to design is Selkies and Munros, a Scottish twist on the classic game Snakes and Ladders. Selkies and Munros is a game traditionally played on a board that is played by two or more players who take turns rolling a die in order to know how many spaces they can move across the grid. The objective of the game is to avoid the obstacles and get to the top left of the board first. This document will describe how we will create the Selkies and Munros project.

**Requirements**

## Functional Requirements

R1. The system shall have a menu containing a start, load, help and exit option.

R2. When the game is initialized a 10x10 grid should be displayed.

R2.1. The Selkies and Munros should be displayed in the correct positions on the grid each time the grid is generated.

R3. The system should take 2 users.

R4. The system should display who the current player is when it is their turn.

R5. The system shall generate a dice roll for each player on each turn.

R5.1 The system shall move the player along the number grid according to their dice roll.

R6. The system should introduce the additional feature “whisky boost”.

R6.1 If the user gets a whisky boost the system shall move the user a further five spaces.

R7 When a player lands on a Selkie or Munro the game shall move the player to the correct location

Rationale: Selkie’s taking the player down the grid (towards the starting square) and Munros taking the player up towards the goal (final square)

R8 The game shall implement a few basic rules, which include: the player cannot go up a Selkie or down a Munro, the player must go up a Munro or down a Selkie (when applicable) and the user must move the number of squares that they rolled.

R9 When deciding a winner, the system Should take the first player who gets their character to position [0][0] in the array.

## Non-functional Requirements

NR1 The system Should run on windows 10 onwards.

NR2 The system shall have an intuitive Layout.

NR3 The system Shall be Created using the Java programming language.

NR4 The System Shall be limited to 4 Selkies and Munros that are possible on the board in each game.

NR5 The System Shall be limited to 3 Activity Squares on the board in each game.

# User Interface

A white background with black text

Description automatically generated

A diagram of a diagram

Description automatically generated with medium confidence

A grid with text on it

Description automatically generated

# Use Cases

**Start of the game** **Alternatives**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | System | Generates menu with the options to start a new game or to load a game that has been saved |  |
| 2 | User | Starts new game | A1 |
| 3 | System | Generates a new 10x10 numbered grid with Selkies and Munros randomly placed throughout the grid |  |
| 4 | System | Displays who the current player is. |  |
| 5 | System | Displays options for user to play the next turn, pause the game or exit the game. |  |

**A1 – Load game** **Alternatives**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | User | Loads a game that has been saved | A2 |
| 2 | System | Generates the grid that had been saved |  |
| 3 | System | Returns to step 4 in the ‘Start of the game’ flow |  |

**A2 – Help option**

|  |  |  |
| --- | --- | --- |
| 1 | User | Selects help option |
| 2 | System | Displays the rules of the game |
| 3 | System | Returns to step 1 in the ‘Start of the game’ flow |

**A3 – Exit**

|  |  |  |
| --- | --- | --- |
| 1 | User | Exits the game |
| 2 | System | Ends the program |

**Each turn of the game** **Alternatives**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | User | Enters 1 to generate a dice roll and move their character along the grid | A4 |
| 2 | System | Generates a random dice roll each time 1 is entered |  |
| 3 | System | Displays what random number was generated | A8 |

**A4 - Game is paused**  **Alternatives**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | User | Selects the pause option during the game | A5 |
| 2 | System | Displays a menu containing resume, save and help options. |  |
| 3 | User | Resumes the current game by selecting the resume option | A6 |

**A5 – Exit game**

|  |  |  |
| --- | --- | --- |
| 1 | User | Selects the exit option during the game |
| 2 | System | Exits the current game |
| 3 | System | Returns to step 1 in ‘Start of game’ flow |

**A6 – User saves the game**  **Alternatives**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | User | Selects the save option | A7 |
| 2 | System | It saves the users current positions on the grid by writing them to file. |  |
| 3 | System | Returns to step 2 in the ‘Game is paused’ flow |  |

**A7 – Help option**

|  |  |  |
| --- | --- | --- |
| 1 | User | Selects help option |
| 2 | System | Displays the rules of the game |
| 3 | System | Returns to step 2 in the ‘Game is paused’ flow |

**A8 – User lands on selkie**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | System | Moves player to the bottom of the selkie if they land on a selkie | A9 |
| 2 | System | Returns to step 1 in the ‘Each turn of game’ flow |  |

**A9 – User lands on Munro**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | System | Moves player to the top of the Munro if they land on a Munro | A10 |
| 2 | System | Returns to step 1 in the ‘Each turn of game’ flow |  |

**A10 – User lands on whisky boost**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | System | Moves the player a further five spaces if they land on a whisky boost. |  |
| 2 | System | Returns to step 1 in the ‘Each turn of game’ flow |  |

# Classes

## Candidate Classes

|  |  |  |
| --- | --- | --- |
| Candidate Class | Accept/Reject? | Reason for rejection |
| board | Accept |  |
| die | Reject | The dice roll is generated within the player class |
| Selkies | Accept |  |
| Munros | Accept |  |
| player | Accept |  |
| Whiskey Boost | Reject | The different types of spaces are displayed within the board class |
| spaces | Reject | The different types of spaces are displayed within the board class |
| Menu | Accept |  |
| Grid | Reject | Duplication of board |
| user | Reject | Duplication of player |
| Name | Reject | Duplication of User/player |
| Help button | Reject | Would be part of the menu class |
| Load Button | Reject | Would be part of the menu class |
| Save button | Reject | Would be part of the menu class |
| Exit button | Reject | Would be part of the menu class |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Class Descriptions including Responsibilities, Fields, and Methods

The Player Class is used to keep track of the players dice roll and whether it is their turn or not. There are no fields within this class. The generateDiceRoll() method returns a random integer between 1 and 6. The generateP1DiceRoll() method assigns a random number generated to player 1. The generateP2DiceRoll() assigns a random number generated to player 2. The playerTurn() method alternates between each player, displaying an option to play their turn, pause the game or exit the game each time.

The Board class displays the board as well as displaying and moving the objects on the board (players, selkies, munros). The fields in this class are the board array, updatedPlayer1Col, updatedPlayer1Row, updatedPlayer2Col, updatedPlayer2Row, currentPlayer1Col, currentPlayer1Row, currentPlayer2Col, and currentPlayer2Row. The initialiseBoard() method initializes each space on the board to be empty. The displayBoard() method displays the grid to the user. The displayActivitySquares() method displays the selkies, munros, and whisky boosts on the board. The initializePlayerLocation() method the players location to the bottom left of the board. The updateP1BoardPos() method moves player 1 along the board according on what number they roll and what spaces they land on. The updateP2BoardPos() method moves player 2 along the board according on what number they roll and what spaces they land on. The saveGame() method writes the current position of each player to file. The loadGame() method reads in each players positions from the previously saved game.

The Menu class displays the menu to the user and processes the user’s choices. The main method runs program. The displayMenu() method displays the main menu to the user. The displayPauseMenu() method displays the pause menu to the user. The processUserChoice() method processes the users choice and runs the appropriate methods. The displayRules() method displays the rules of the game to the user.

The Selkies class stores the positions that players will be moved to if they land on a selkie. The fields in this class are selkie1PlayerPos\_X, selkie1PlayerPos\_Y, selkie2PlayerPos\_X, selkie2PlayerPos\_Y, selkie3PlayerPos\_X, selkie3PlayerPos\_Y, selkie4PlayerPos\_X, selkie4PlayerPos\_Y. The methods in this class return these fields to decide what position on the board the player will be moved to depending on what selkie they land on.

The Munros class stores the positions that players will be moved to if they land on a munro. The fields in this class are munro1PlayerPos\_X, munro1PlayerPos\_Y, munro2PlayerPos\_X, munro2PlayerPos\_Y, munro3PlayerPos\_X, munro3PlayerPos\_Y, munro4PlayerPos\_X, munro4PlayerPos\_Y. The methods in this class return these fields to decide what position on the board the player will be moved to depending on what munro they land on.

## Class Diagram

# Activity Diagrams / Pseudocode

Pseudocode for a roll of a dice and moving the player to that square

Method generateDiceRoll () {

Use the Random class to generate random number between 1 and 6.

Stores it in a variable called randomNumber

Return randomNumber

}

Psuedocode for landing on a Selkie

Method updateP1BoardPos() {

If currentP1BoardPos.contains(“selkie”)

{

updatedPlayer1Col= selkies.getSelkie1PlayerPos\_X;

updatedPlayer1Row = selkies.getSelkie1PlayerPos\_Y;

}

}

Psuedocode for landing on a Munro

Method updatePlayer1BoardPos() {

If currentP1BoardPos.contains(“munro”)

{

updatedPlayer1Col = munro.getMunro1PlayerPos\_X;

updatedPlayer1Row = munro.getMunro1PlayerPos\_Y;

}

}

Psuedocode for landing on a Whisky Boost

Method updatePlayer1BoardPos() {

If currentP1BoardPos.contains(“whisky boost”)

{

updatedPlayer1Col = updatedPlayer1Col + 5;

}

}

Pseudocode for alternating player turn

Method playerTurn() {

currentPlayer = 1;

if userChoice = 1

{

If currentPlayer = 1

{

“Current player is “ + currentPlayer;

displayActivitySquares();

updateP1BoardPos();

displayBoard();

currentPlayer = currentPlayer + 1;

}

Else

{

“Current player is “ + currentPlayer;

displayActivitySquares();

updateP2BoardPos();

displayBoard();

currentPlayer = currentPlayer – 1;

}

}

}