**Script**

Our team decided to design the game, Nine men Morris. It consists of a board containing several points that are placed on squares and lines connecting the points together. The objective of the game is to fill the three slots present on every line in a straight line (only vertical and horizontal lines are permissible, diagonal lines are not permitted). Each player has 9 stones to fill these slots and whenever a line is completely filled, which means all 3 slots have been connected, the player can choose which one of the opponent’s stone can be removed from. Once a player has placed all the stones on the board, then a player can move the stones on adjacent points in the line. When a player has less than 3 stones remaining on the board the player has lost the game.

**Show a demo of the actual game.**

**DEMO OF APPLICATION!!**

Features present from this iteration:

* New stones can be placed on the board
* Move stones on the board to adjacent points
* Remove opponent’s stone after building a line connecting 3 stones
* Artificial intelligence can place stones on the board at random
* The game can determine the winner by evaluating who has less than 3 stones present on the board on the last round

**Class Diagram:**

Points:

To start with, the most basic of our classes was the Points Class. This class essentially created the location for all the slots, which are the visible points on the board, and all other objects created in the future classes that will be discussed in a bit. Each point has its own x and y coordinates on the plane. This is the building block for all future classes.

Stone:

One of the most major components of the game is created using this class. The stones are created as instances of this class in which each stone is assigned a location using the seLocation method as well as associated with an owner, which is an instance of the player class. The point class is also used for a stone at any moment when the stone is being moved or placed on the board.

Once a stone is set on a point, that point becomes associated with a player and a stone through using the separate classes Player and Stone.

The methods getOccupiedPlayer and getOccupiedStone will allow us to determine the respective stone and player occupying the point, if any.

Line:

The line class forms a line connecting three points. This class is heavily related to the points class as the location is set for each point in the line which then makes up the line and sets its slot locations on the board. Each instance of this class will have three instance variables (three points on the line), whether the line is filled or not and an array containing a slot location for each slot/point. This line has a method called isLineFilled which can determine if a slot (visible point) on the line is filled with stone and checks if a line has all three slots filled. This class along with point are then used to create the next class Square.

Square:

Four lines are then used to create the three squares present on the board. There is a method to access each line as well as return the final array that stores all the lines making up the square as well as returning the name of each line. There are two constructors present for this class, the first constructor creates 4 new lines and their points and sets the 4 line into a list. The second constructor takes the 4 lines already made and stores them in a list.

Board:

The board class uses a mixture of square and line which in turn uses the points class in order to setup the game board for Nine men morris. An array of squares is used to store the 3 squares present on the board, the outer, mid and inner most squares as well as the 4 lines connecting the squares together. It also returns all the points of the lines which will be needed in the other classes further into the design to determine whether the line is filled and who is winning.

Player:

This class is the parent class of the children classes human player and AI player. Each time a player is created, it automatically gets an arraylist of 9 stones that can be set on the board. It has basic functions of setting a new stone on the board and moving the stone on the board. This allows for this class to be connected to the stones class which in turn means its connected to points class.

Human Player:

This is the children class of Player and it is meant to represent the human player who will be playing the game. The player will give their input into which stone they want to move, onto which point and which stone they want remove of the opponent’s.

AI Player:

This represents the computer player. The player will go through a series of methods to determine which move is the best to make at every turn. It also decides automatically which stone to remove depending on the game going on, on the board.

Game:

The class creates the board game, which then creates all the points, lines, squares, stones, players (both human and AI). It also has the ability to determine whether a player is valid to place a new stone, move an existing stone or remove an opponent’s stone. It also checks if a point is occupied on the board as well as to determine if the point you want to move an already placed stone to is adjacent or not. It has access to all the players therefore, it checks and removes an opponent’s stone if the player forms a line.

TextApplication:

This class will prompt user input to start or quit the game. Once the game is started, the game class will be called in order to play a game. You can start a new game as many times as you please until you input the word ‘quit’ as that is how the user prompt is designed in this case. The game is based on a while loop that keeps track of the amount of stones each player will have on the board and as soon as one player has less than three stones on the board, the game will automatically end. This will determine who the winner of the game is.

**Challenges:**

The main challenge was coordinating with other team members to make sure a certain procedure that everyone agreed to was being followed while also trying to keep up with constant changes being made to the code in order to converge on the end point of the game together. The constant updates and understanding each others logic without having to meet was a major challenge and making sure. The time pressure in which all of this was being conducted was a huge challenge as well but all of this was resolved through increased communication (through several meetings and problem solving together).

**Future Plans:**

Have a proper GUI set up for the user to interact with. This can be seen through a proper image of the board being set up as well as the user being able to click on the screen in order for the stones to move. Visual images of the stones moving or being removed are going to be available.