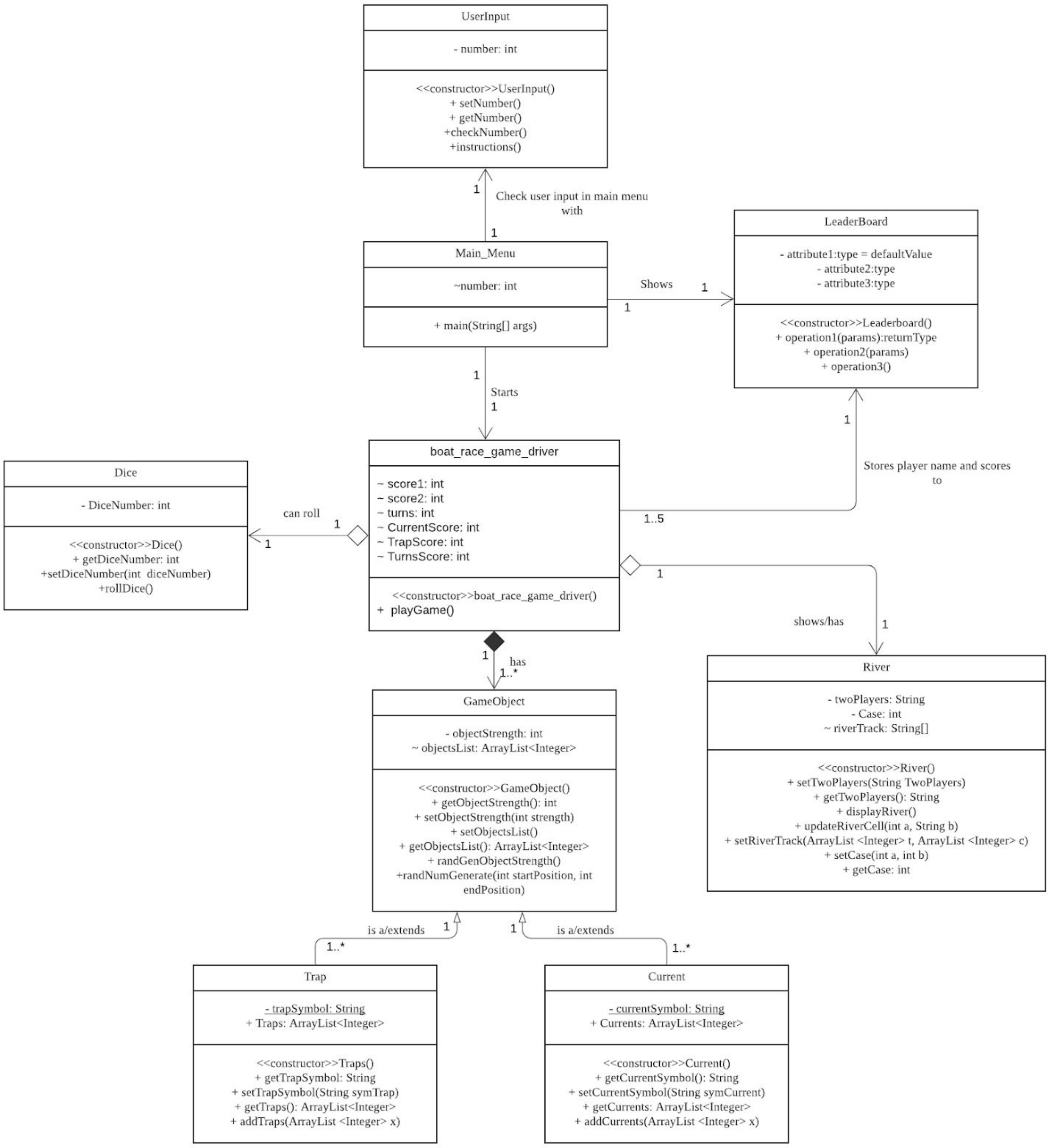
OOPF Group 3 Report (Boat Game)

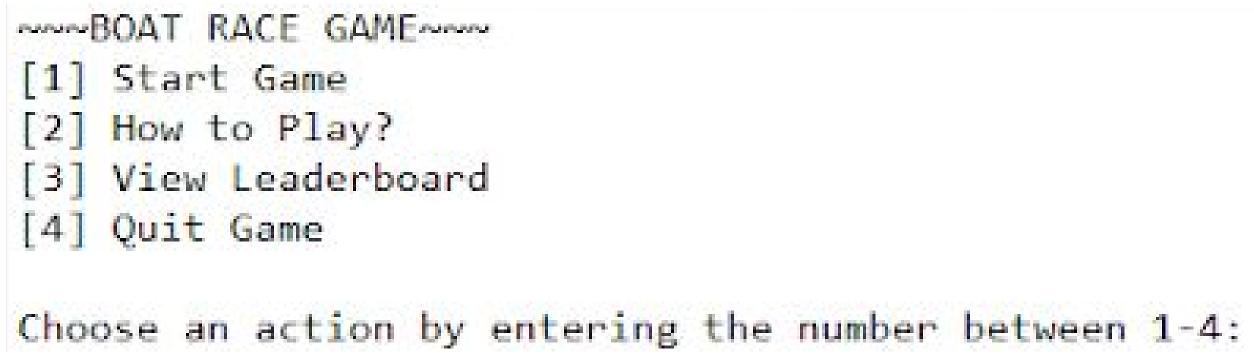
Design: UML Diagram


UserInput
- number: int
<<constructor>>UserInput()
+ setNumber()
+ getNumber()
+checkNumber()
+instructions()
1
Check user input in main menu with
1
Main_Menu
~number: int
+ main(String[] args)
Shows
1
1
LeaderBoard
- attribute1:type = defaultValue
- attribute2:type
- attribute3:type
<<constructor>>Leaderboard()
+ operation1(params):returnType
+ operation2(params)
+ operation3()
1
Starts
1
1
Stores player name and scores to
1..5
Dice
- DiceNumber: int
<<constructor>>Dice()
+ getDiceNumber: int
+setDiceNumber(int diceNumber)
+rollDice()
can roll
1
1
boat_race_game_driver
~ score1: int
~ score2: int
~ turns: int
~ CurrentScore: int
~ TrapScore: int
~ TurnsScore: int
<<constructor>>boat_race_game_driver()
+ playGame()
1
shows/has
1
River
- twoPlayers: String
- Case: int
~ riverTrack: String[]
<<constructor>>River()
+ setTwoPlayers(String TwoPlayers)
+ getTwoPlayers(): String
+ displayRiver()
+ updateRiverCell(int a, String b)
+ setRiverTrack(ArrayList <Integer> t, ArrayList <Integer> c)
+ setCase(int a, int b)
+ getCase: int
1
has
1..*
GameObject
- objectStrength: int
~ objectsList: ArrayList<Integer>
<<constructor>>GameObject()
+ getObjectStrength(): int
+ setObjectStrength(int strength)
+ setObjectsList()
+ getObjectsList(): ArrayList<Integer>
+ randGenObjectStrength()
+randNumGenerate(int startPosition, int endPosition)
is a/extends
1
1..*
1
is a/extends
1..*
Trap
- trapSymbol: String
+ Traps: ArrayList<Integer>
<<constructor>>Traps()
+ getTrapSymbol: String
+ setTrapSymbol(String symTrap)
+ getTraps(): ArrayList<Integer>
+ addTraps(ArrayList <Integer> x)
Current
- currentSymbol: String
+ Currents: ArrayList<Integer>
<<constructor>>Current()
+ getCurrentSymbol(): String
+ setCurrentSymbol(String symCurrent)
+ getCurrents: ArrayList<Integer>
+ addCurrents(ArrayList <Integer> x)

Added Functionality: Main Menu

For this project, our group has implemented a **main menu** which acts as an interactive user interface so that the user can choose whether they want to start the game, close the program, or view the instructions.

****The menu will appear as below:***

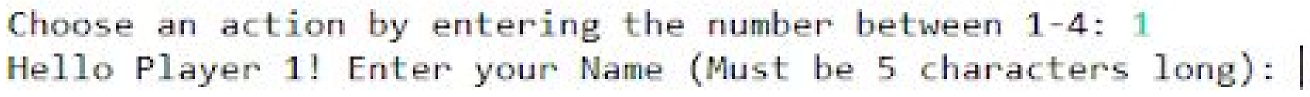


```
~~~~BOAT RACE GAME~~~~
[1] Start Game
[2] How to Play?
[3] View Leaderboard
[4] Quit Game

Choose an action by entering the number between 1-4:
```

The aim here is to make the program more versatile by giving the user more options rather than just playing the game straight away. The main menu here adds the benefit of allowing the user to understand the game beforehand as well as restart the game if there is a game over rather than executing the program from the beginning again.

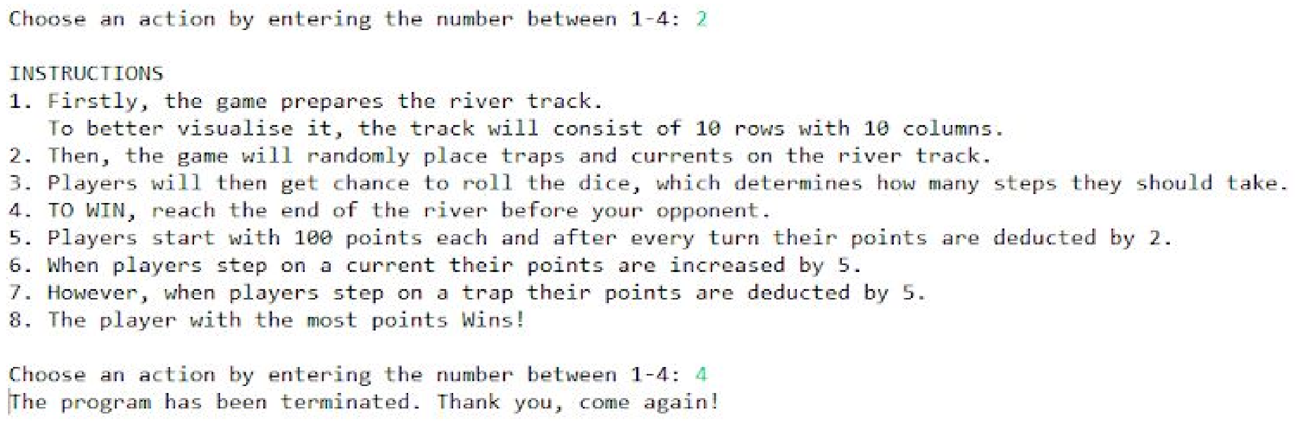
Each number will make the program execute a different task. It should be noted that after every execution other than '4' which ends the program, the user will be prompted again on what they want to do. This includes starting a new game, reading the instructions, viewing the updated scoreboard, or ending the program.

If the user enters ‘1’, the game will start:



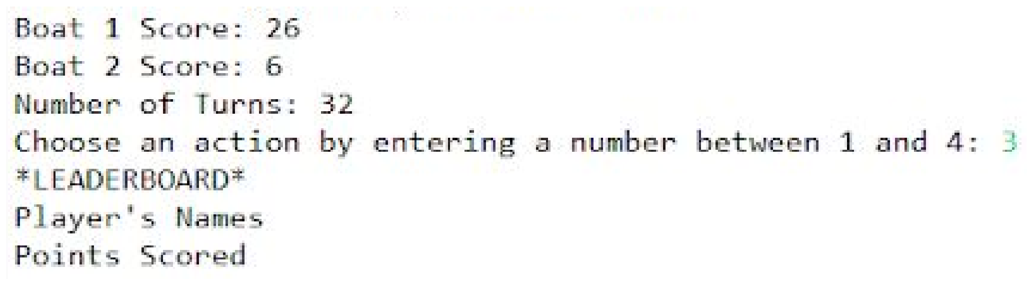
```
Choose an action by entering the number between 1-4: 1
Hello Player 1! Enter your Name (Must be 5 characters long):
```

If the user enters ‘2’, the instructions will be shown:



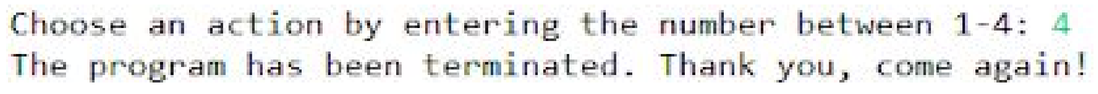
```
Choose an action by entering the number between 1-4: 2

INSTRUCTIONS
1. Firstly, the game prepares the river track.
   To better visualise it, the track will consist of 10 rows with 10 columns.
2. Then, the game will randomly place traps and currents on the river track.
3. Players will then get chance to roll the dice, which determines how many steps they should take.
4. TO WIN, reach the end of the river before your opponent.
5. Players start with 100 points each and after every turn their points are deducted by 2.
6. When players step on a current their points are increased by 5.
7. However, when players step on a trap their points are deducted by 5.
8. The player with the most points Wins!

Choose an action by entering the number between 1-4: 4
The program has been terminated. Thank you, come again!
```

If the user enters ‘3’, the top 5 scores will be shown:



```
Boat 1 Score: 26
Boat 2 Score: 6
Number of Turns: 32
Choose an action by entering a number between 1 and 4: 3
*LEADERBOARD*
Player's Names
Points Scored
```

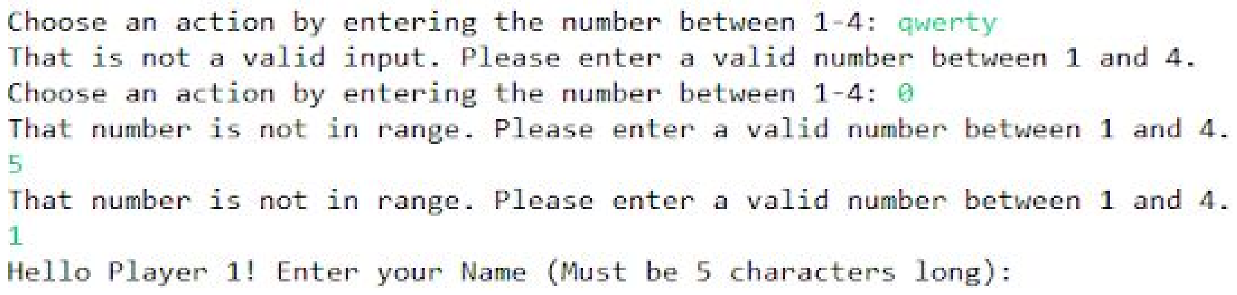
If the user enters ‘4’, the program will end:



```
Choose an action by entering the number between 1-4: 4
The program has been terminated. Thank you, come again!
```

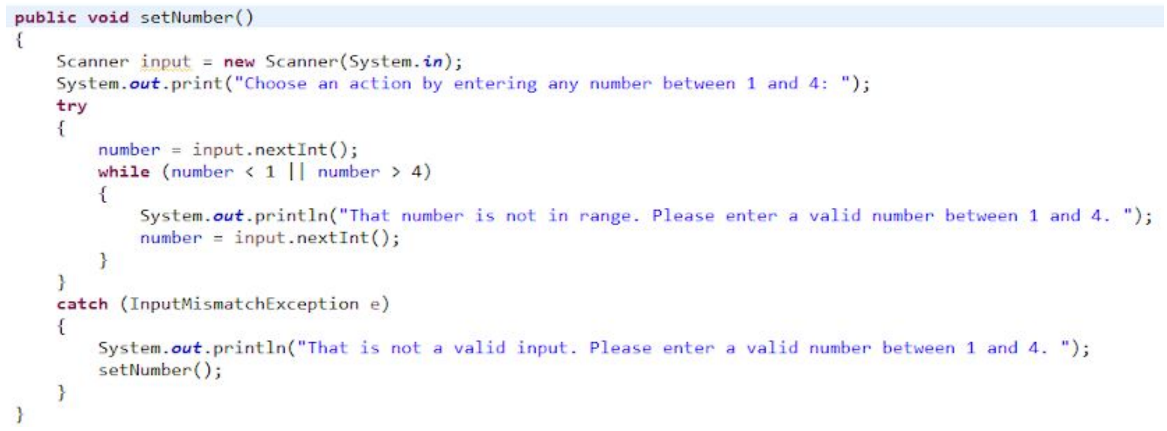
Error handling is used to ensure that the data type of the user input is valid. Furthermore, the input range is checked using while loops as well to ensure that the input is within a set range which in this case is 1 to 4.

<u>*Input validation is done by ensuring that the data is within the set range as well as checking the input's data type:</u>



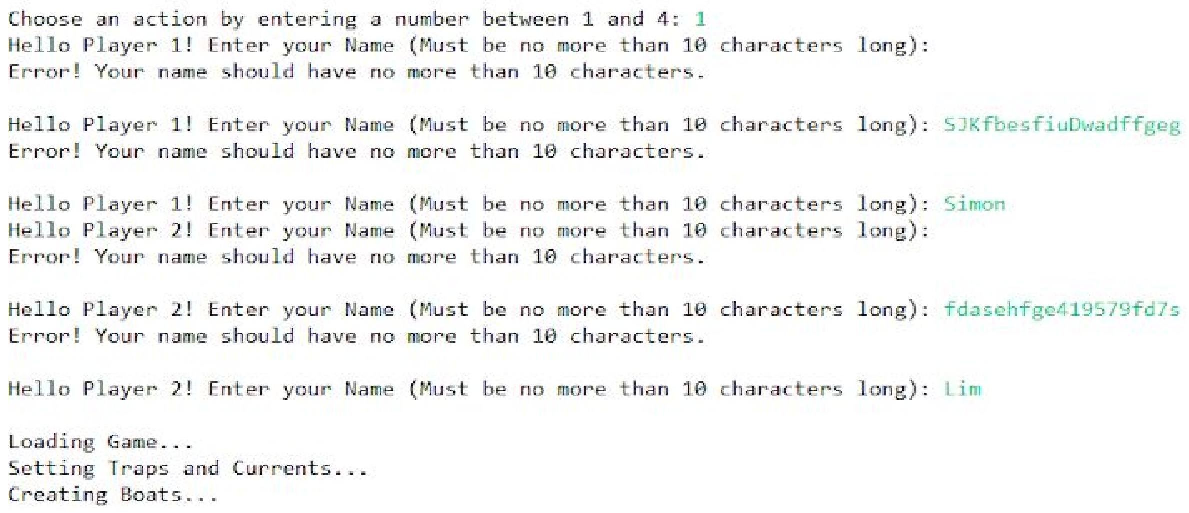
```
Choose an action by entering the number between 1-4: qwerty
That is not a valid input. Please enter a valid number between 1 and 4.
Choose an action by entering the number between 1-4: 0
That number is not in range. Please enter a valid number between 1 and 4.
5
That number is not in range. Please enter a valid number between 1 and 4.
1
Hello Player 1! Enter your Name (Must be 5 characters long):
```

<u>*Code used for input validation as well as exception handling:</u>



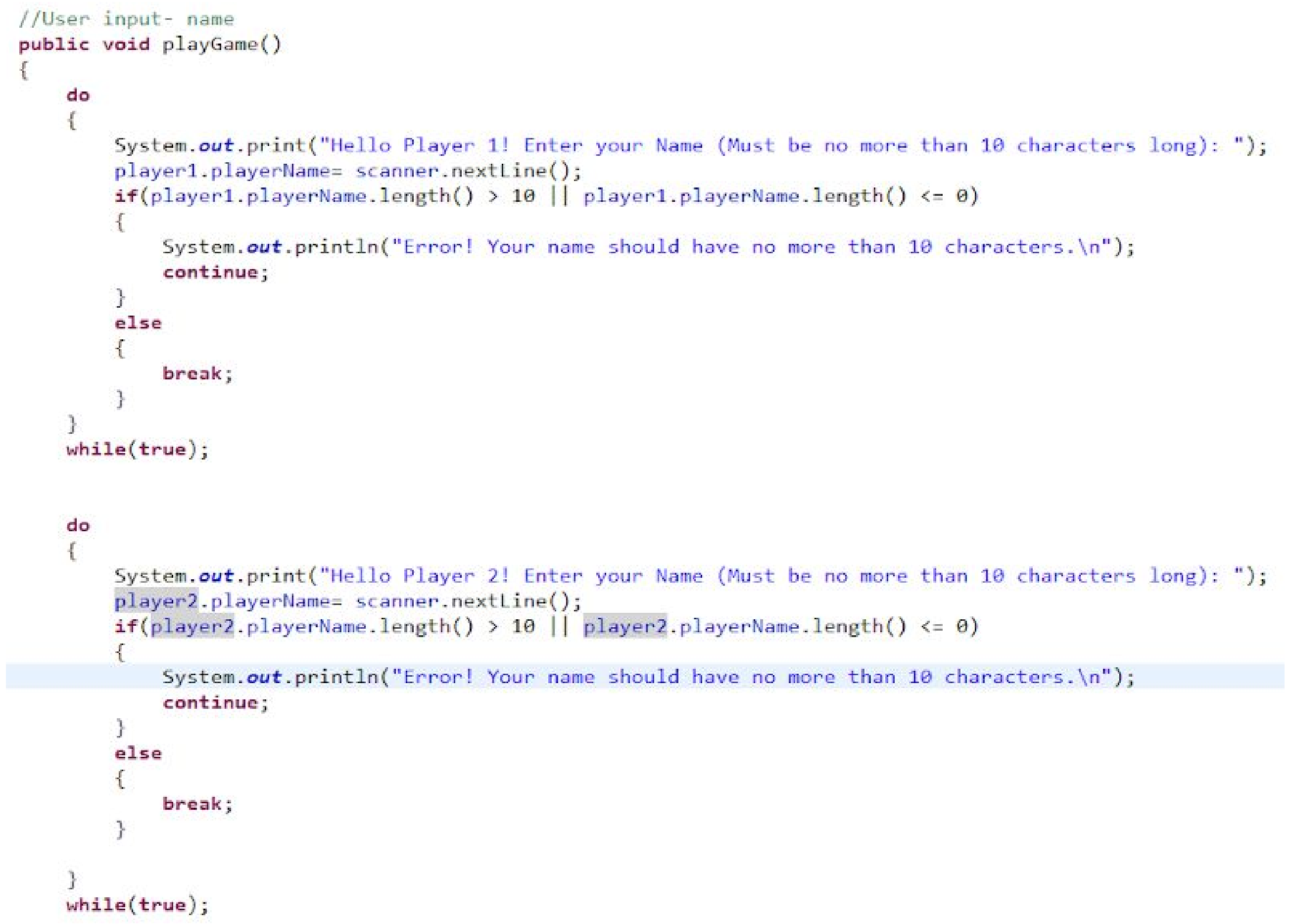
```java
public void setNumber()
{
    Scanner input = new Scanner(System.in);
    System.out.print("Choose an action by entering any number between 1 and 4: ");
    try
    {
        number = input.nextInt();
        while (number < 1 || number > 4)
        {
            System.out.println("That number is not in range. Please enter a valid number between 1 and 4. ");
            number = input.nextInt();
        }
    }
    catch (InputMismatchException e)
    {
        System.out.println("That is not a valid input. Please enter a valid number between 1 and 4. ");
        setNumber();
    }
}
```

<u>*Same for input validation used to ensure names are not too long:</u>



```
Choose an action by entering a number between 1 and 4: 1
Hello Player 1! Enter your Name (Must be no more than 10 characters long):
Error! Your name should have no more than 10 characters.

Hello Player 1! Enter your Name (Must be no more than 10 characters long): SJKfbesfiuDwadffgeg
Error! Your name should have no more than 10 characters.

Hello Player 1! Enter your Name (Must be no more than 10 characters long): Simon
Hello Player 2! Enter your Name (Must be no more than 10 characters long):
Error! Your name should have no more than 10 characters.

Hello Player 2! Enter your Name (Must be no more than 10 characters long): fdasehfge419579fd7s
Error! Your name should have no more than 10 characters.

Hello Player 2! Enter your Name (Must be no more than 10 characters long): Lim

Loading Game...
Setting Traps and Currents...
Creating Boats...
```

<u>Respective code:</u>

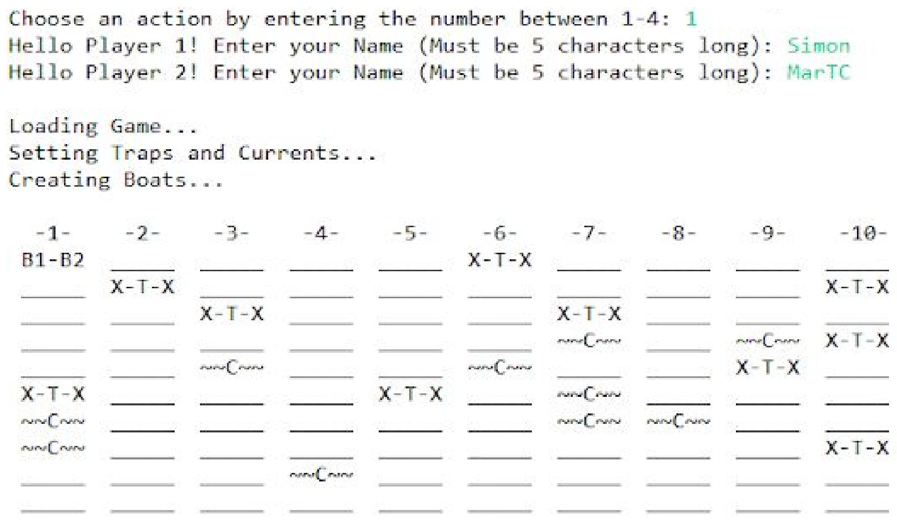


```java
//User input- name
public void playGame()
{
    do
    {
        System.out.print("Hello Player 1! Enter your Name (Must be no more than 10 characters long): ");
        player1.playerName= scanner.nextLine();
        if(player1.playerName.length() > 10 || player1.playerName.length() <= 0)
        {
            System.out.println("Error! Your name should have no more than 10 characters.\n");
            continue;
        }
        else
        {
            break;
        }
    }
    while(true);


    do
    {
        System.out.print("Hello Player 2! Enter your Name (Must be no more than 10 characters long): ");
        player2.playerName= scanner.nextLine();
        if(player2.playerName.length() > 10 || player2.playerName.length() <= 0)
        {
            System.out.println("Error! Your name should have no more than 10 characters.\n");
            continue;
        }
        else
        {
            break;
        }

    }
    while(true);
```

Testing: Program Working as Intended

Once the users enter their names, the game begins.

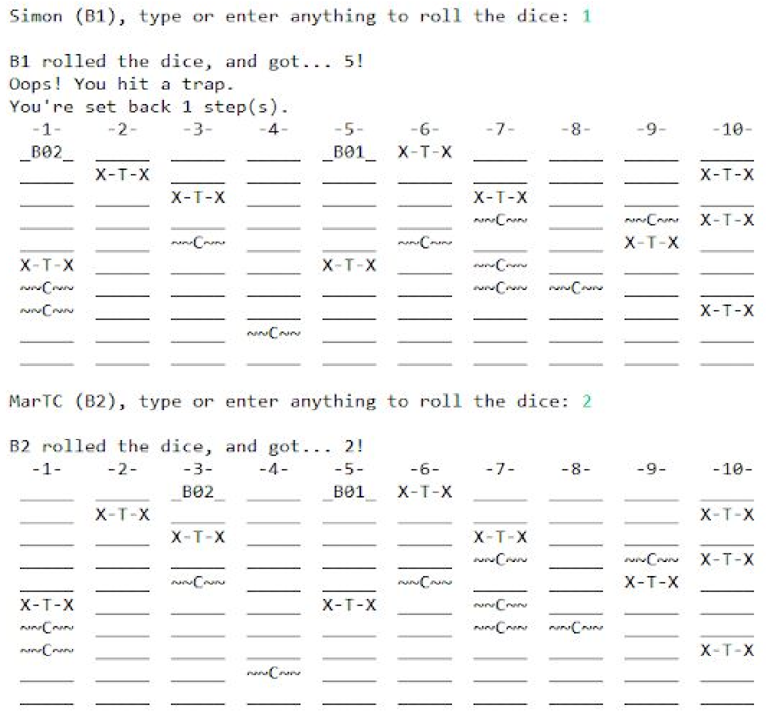
The grid of the river is shown as below:



```
Choose an action by entering the number between 1-4: 1
Hello Player 1! Enter your Name (Must be 5 characters long): Simon
Hello Player 2! Enter your Name (Must be 5 characters long): MarTC

Loading Game...
Setting Traps and Currents...
Creating Boats...

 -1-    -2-    -3-    -4-    -5-    -6-    -7-    -8-    -9-    -10-
B1-B2  _____  _____  _____  _____  X-T-X  _____  _____  _____  _____
_____  X-T-X  _____  _____  _____  _____  _____  _____  _____  X-T-X
_____  _____  X-T-X  _____  _____  _____  X-T-X  _____  _____  _____
_____  _____  _____  _____  _____  _____  ~~C~~  _____  ~~C~~  X-T-X
_____  _____  ~~C~~  _____  _____  ~~C~~  _____  _____  X-T-X  _____
X-T-X  _____  _____  _____  X-T-X  _____  ~~C~~  _____  _____  _____
~~C~~  _____  _____  _____  _____  _____  ~~C~~  ~~C~~  _____  _____
~~C~~  _____  _____  _____  _____  _____  _____  _____  _____  X-T-X
_____  _____  _____  ~~C~~  _____  _____  _____  _____  _____  _____
_____  _____  _____  _____  _____  _____  _____  _____  _____  _____
```

The first player gets to roll the dice first, the boat will move based on the dice's number. Then, the second player will also roll the dice and move their boat accordingly. This is repeated until one of the players reaches the end.

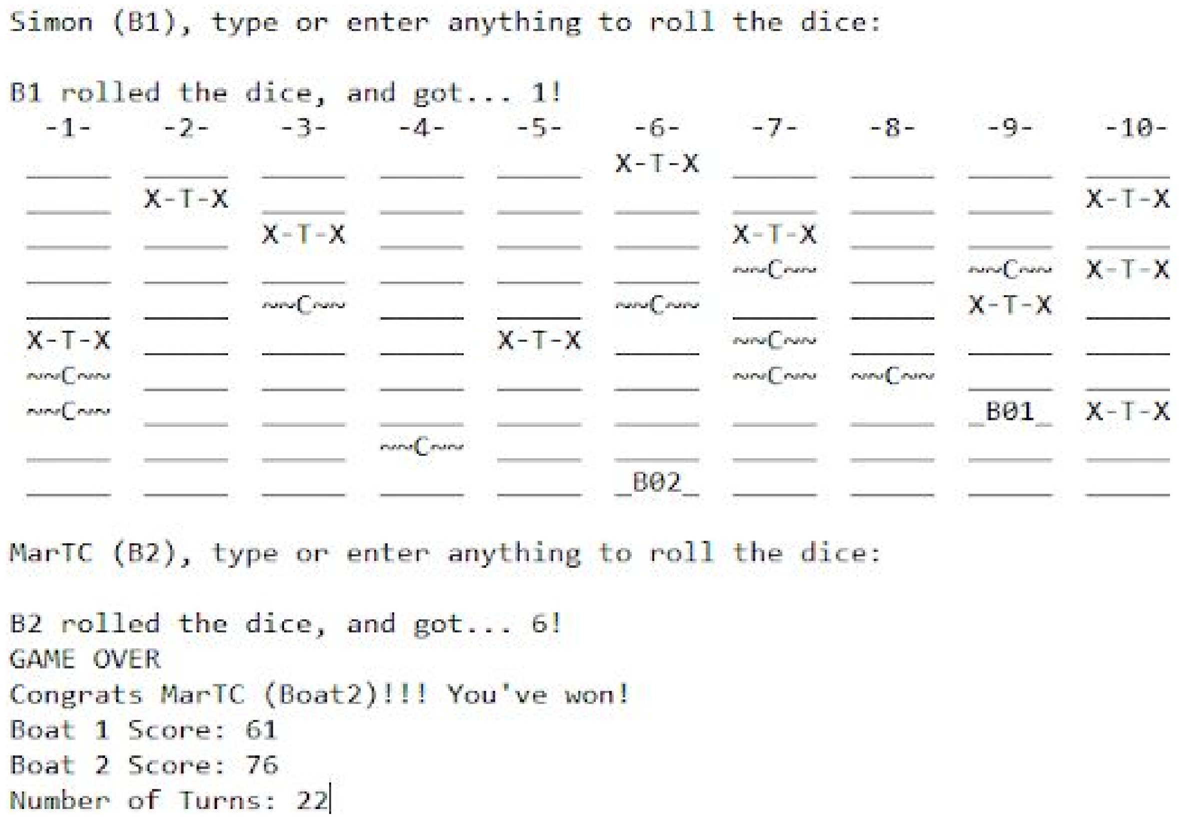
An example is shown below:



```
Simon (B1), type or enter anything to roll the dice: 1

B1 rolled the dice, and got... 5!
Oops! You hit a trap.
You're set back 1 step(s).
 -1-    -2-    -3-    -4-    -5-    -6-    -7-    -8-    -9-    -10-
_B02_  _____  _____  _____  _B01_  X-T-X  _____  _____  _____  _____
_____  X-T-X  _____  _____  _____  _____  _____  _____  _____  X-T-X
_____  _____  X-T-X  _____  _____  _____  X-T-X  _____  _____  _____
_____  _____  _____  _____  _____  _____  ~~C~~  _____  ~~C~~  X-T-X
_____  _____  ~~C~~  _____  _____  ~~C~~  _____  _____  X-T-X  _____
X-T-X  _____  _____  _____  X-T-X  _____  ~~C~~  _____  _____  _____
~~C~~  _____  _____  _____  _____  _____  ~~C~~  ~~C~~  _____  _____
~~C~~  _____  _____  _____  _____  _____  _____  _____  _____  X-T-X
_____  _____  _____  ~~C~~  _____  _____  _____  _____  _____  _____
_____  _____  _____  _____  _____  _____  _____  _____  _____  _____

MarTC (B2), type or enter anything to roll the dice: 2

B2 rolled the dice, and got... 2!
 -1-    -2-    -3-    -4-    -5-    -6-    -7-    -8-    -9-    -10-
_____  _____  _B02_  _____  _B01_  X-T-X  _____  _____  _____  _____
_____  X-T-X  _____  _____  _____  _____  _____  _____  _____  X-T-X
_____  _____  X-T-X  _____  _____  _____  X-T-X  _____  _____  _____
_____  _____  _____  _____  _____  _____  ~~C~~  _____  ~~C~~  X-T-X
_____  _____  ~~C~~  _____  _____  ~~C~~  _____  _____  X-T-X  _____
X-T-X  _____  _____  _____  X-T-X  _____  ~~C~~  _____  _____  _____
~~C~~  _____  _____  _____  _____  _____  ~~C~~  ~~C~~  _____  _____
~~C~~  _____  _____  _____  _____  _____  _____  _____  _____  X-T-X
_____  _____  _____  ~~C~~  _____  _____  _____  _____  _____  _____
_____  _____  _____  _____  _____  _____  _____  _____  _____  _____
```

Game-winning turn: Once one of the players reaches the end, the user is shown the winner’s name, boat, score. The score is then recorded in the leaderboard if it is among the top 5 scores recorded.

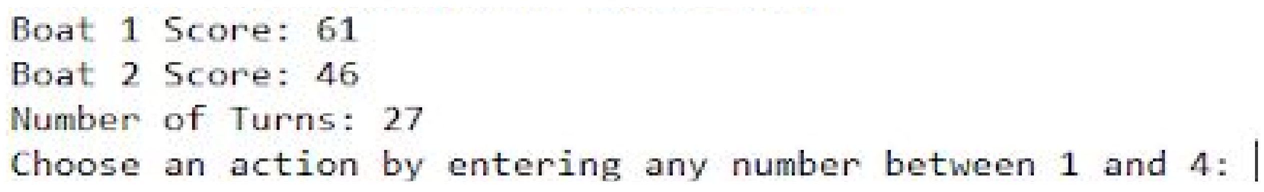
<u>An example is shown below:</u>



```
Simon (B1), type or enter anything to roll the dice:

B1 rolled the dice, and got... 1!
 -1-    -2-    -3-    -4-    -5-    -6-    -7-    -8-    -9-   -10-
_____  _____  _____  _____  _____  X-T-X  _____  _____  _____  _____
_____  X-T-X  _____  _____  _____  _____  _____  _____  _____  X-T-X
_____  _____  X-T-X  _____  _____  _____  X-T-X  _____  _____  _____
_____  _____  _____  _____  _____  _____  ~~C~~  _____  ~~C~~  X-T-X
_____  _____  ~~C~~  _____  _____  ~~C~~  _____  _____  X-T-X  _____
X-T-X  _____  _____  _____  X-T-X  _____  ~~C~~  _____  _____  _____
~~C~~  _____  _____  _____  _____  _____  ~~C~~  ~~C~~  _____  _____
~~C~~  _____  _____  _____  _____  _____  _____  _____  _B01_  X-T-X
_____  _____  _____  ~~C~~  _____  _____  _____  _____  _____  _____
_____  _____  _____  _____  _____  _B02_  _____  _____  _____  _____

MarTC (B2), type or enter anything to roll the dice:

B2 rolled the dice, and got... 6!
GAME OVER
Congrats MarTC (Boat2)!!! You've won!
Boat 1 Score: 61
Boat 2 Score: 76
Number of Turns: 22
```

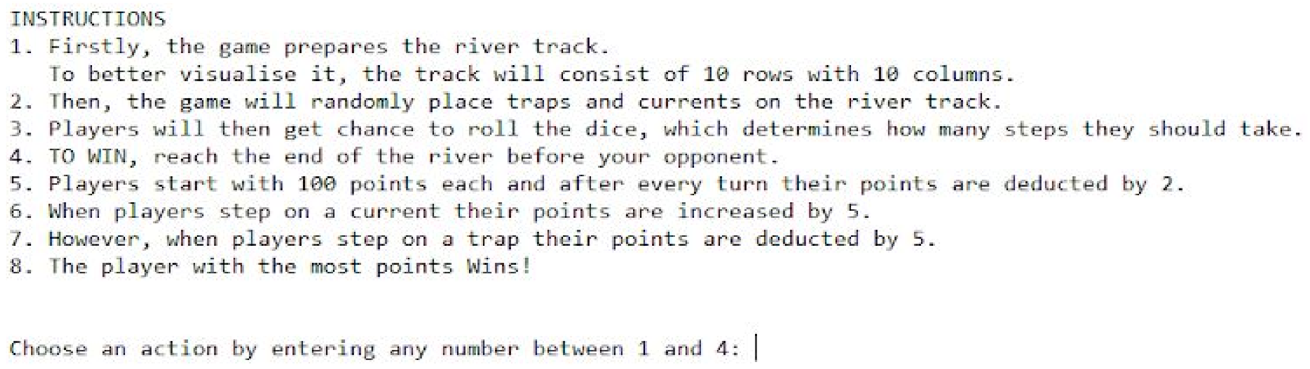
It can be observed that once the game ends, the user is prompted on whether they want to start a new game, relook at the instructions, check the updated leaderboard, or end the program.

<u>*The user interface pops up again once the game ends:</u>



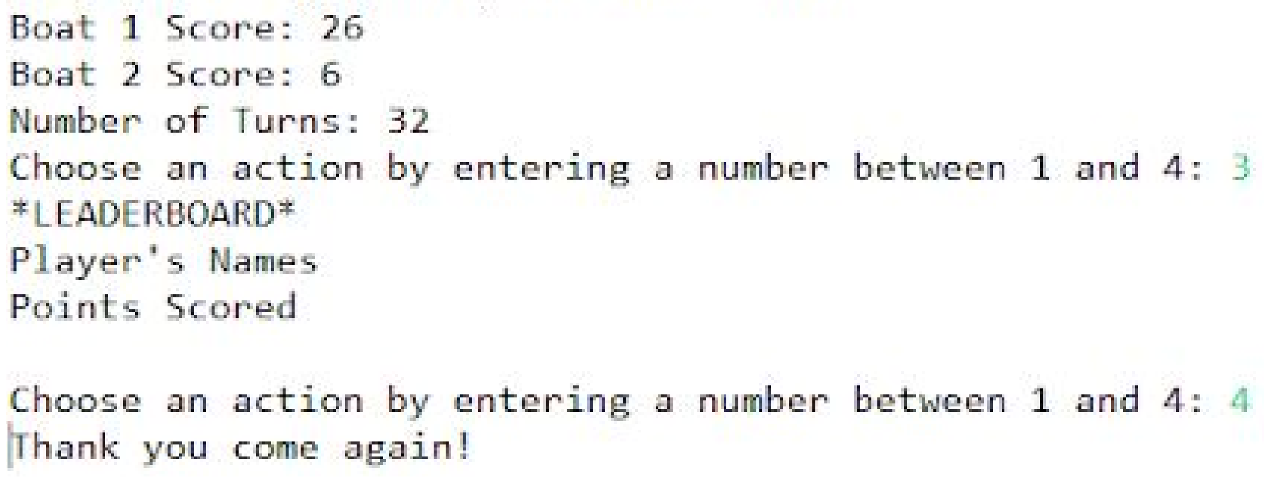
```
Boat 1 Score: 61
Boat 2 Score: 46
Number of Turns: 27
Choose an action by entering any number between 1 and 4:
```

<u>*The user interface pops up again once the instructions are shown:</u>



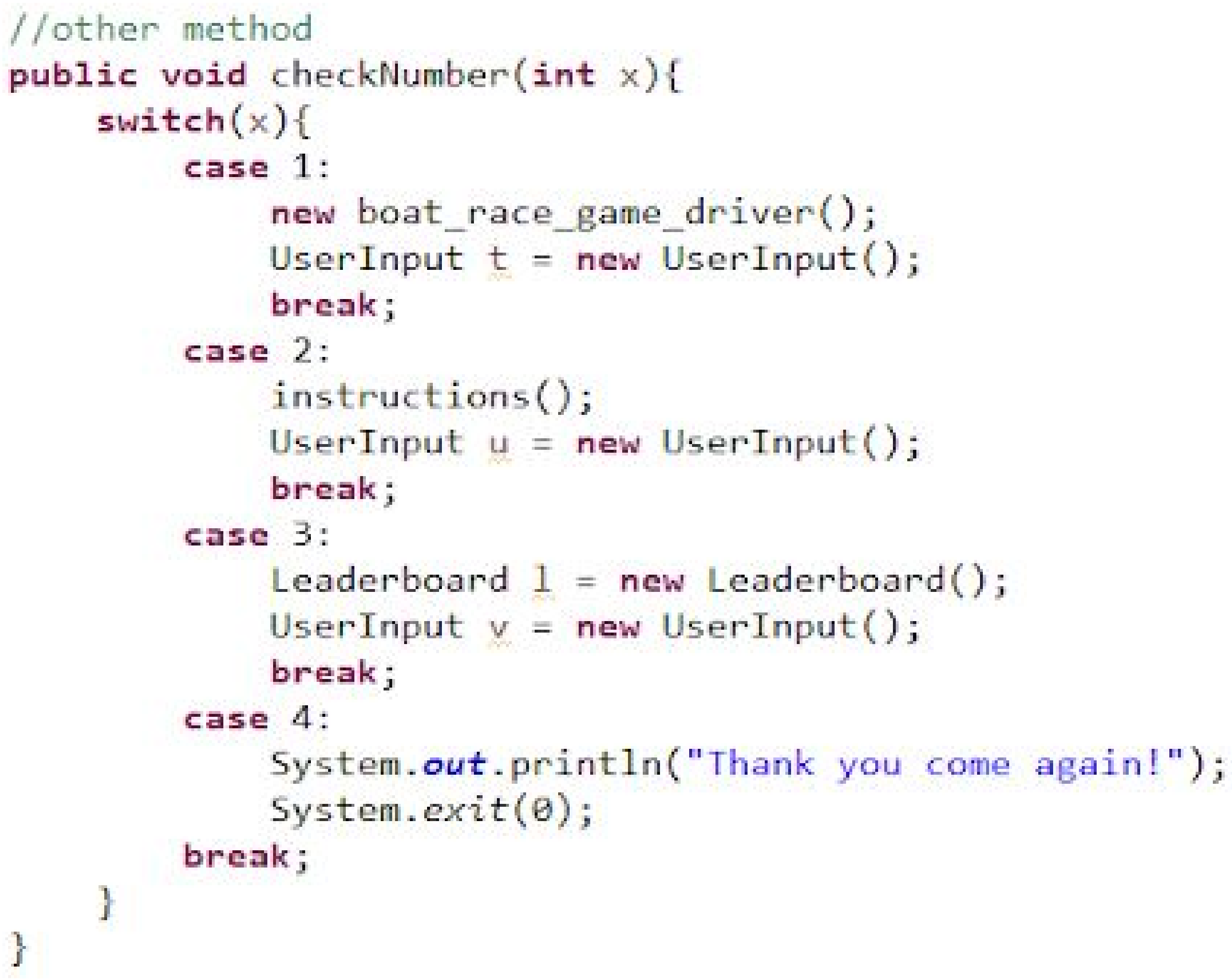
```
INSTRUCTIONS
1. Firstly, the game prepares the river track.
   To better visualise it, the track will consist of 10 rows with 10 columns.
2. Then, the game will randomly place traps and currents on the river track.
3. Players will then get chance to roll the dice, which determines how many steps they should take.
4. TO WIN, reach the end of the river before your opponent.
5. Players start with 100 points each and after every turn their points are deducted by 2.
6. When players step on a current their points are increased by 5.
7. However, when players step on a trap their points are deducted by 5.
8. The player with the most points Wins!


Choose an action by entering any number between 1 and 4:
```

<u>*The user interface pops up again once the scores are shown:</u>



```
Boat 1 Score: 26
Boat 2 Score: 6
Number of Turns: 32
Choose an action by entering a number between 1 and 4: 3
*LEADERBOARD*
Player's Names
Points Scored

Choose an action by entering a number between 1 and 4: 4
Thank you come again!
```

<u>Code used to rerun UserInput again:</u>



```java
//other method
public void checkNumber(int x){
    switch(x){
        case 1:
            new boat_race_game_driver();
            UserInput t = new UserInput();
            break;
        case 2:
            instructions();
            UserInput u = new UserInput();
            break;
        case 3:
            Leaderboard l = new Leaderboard();
            UserInput v = new UserInput();
            break;
        case 4:
            System.out.println("Thank you come again!");
            System.exit(0);
        break;
    }
}
```

Conclusion

In summary, we implemented an interactive main menu that users can play around with. In the future, we hope that we can improve and add more functions to the main menu. Other than that, certain parts that we could have improved on was the design of the game. Although there is nothing wrong with the design at this time, the game itself is relatively simple with basic functionality and a grid. If we had more knowledge and experience, it may have been possible to design it in a way where it's more interactive.To add on, not adding the leaderboard is something we were unable to do and we hope to be able to code it in the near future. To clarify, the future ambition here is to have more input options other than simply typing in text. As an example, adding items or giving the players a second chance in rolling if they hit the trap. In conclusion, other than some suggestions on improvements, the code is running properly and there are minimal to no bugs present after constant troubleshooting and fixing.