SCHOOL OF ENGINEERING AND TECHNOLOGY

COURSEWORK FOR THE

BSC (HONS) INFORMATION TECHNOLOGY; YEAR 1
BSC (HONS) COMPUTER SCIENCE; YEAR 1
BSC (HONS) INFORMATION TECHNOLOGY (COMPUTER NETWORKING AND SECURITY); YEAR 1
BSC (HONS) SOFTWARE ENGINEERING; YEAR 1

ACADEMIC SESSION 2022; SEMESTER 2,3,4

PRG1203: OBJECT ORIENTED PROGRAMMING FUNDAMENTALS

DEADLINE: 2 DECEMBER 2022 11:59PM (Friday)

INSTRUCTIONS TO CANDIDATES

- **This assignment will contribute 20% to your final grade.**
- **This is a group (maximum 5 students) assignment**

IMPORTANT

The University requires students to adhere to submission deadlines for any form of assessment. Penalties are applied in relation to unauthorized late submission of work.

Any work submitted after the deadline, or after any period of extension granted shall be marked as a Fail or awarded a zero.

Academic Honesty Acknowledgement

YASHNNI SUGUMAR, DHARSHAAN SEGARAN, LIIVENESH SUNDARA RAJU, ANEEYSA BINTI REDUAN, verify that this paper contains entirely my own work. I have not consulted with any outside person or materials other than what was specified (an interviewee, for example) in the assignment or the syllabus requirements. Further, I have not copied or inadvertently copied ideas, sentences, or paragraphs from another student. I realize the penalties *(refer student handbook undergraduate programme)* for any kind of copying or collaboration on any assignment."

Yashnni, Dharshaan, Liivenesh, Aneeysa (2/12/2022)

Group Number: Group 19

Team Members:

No	Name	Student ID	Contribution %
1	ANEEYSA BINTI REDUAN	21086525	30%
2	YASHNNI SUGUMAR	20061156	30%
3	DHARSHAAN SEGARAN	20029997	30%
4	LIIVENESH SUNDARA RAJU	20061586	10%

Marking Scheme

Criteria	Reference Marks		Marks	Remarks
Design (10%) Implement good object-oriented design in solving the problem, with high modularity, maintainability and reusability. Able to identify appropriate classes and their relationships, complete the classes with appropriate attributes and methods. The design is well presented in UML class and class relationship diagrams.	10	Excellent		
	7-9	Good		
	4-6	Average		
	1-3	Poor		
Coding (5%) Fulfil all the functionalities and align to the design you have presented in the UML diagrams. Follow the best programming practices, such as naming convention, indenting, code structure, optimisation, with appropriate exception handling. Good user-friendliness.	5	Excellent		
	4	Good		
	2	Average		
	1	Poor		
Additional Functionality (5%) Add at least one additional enhancement or functionality to your program. Explain the rationale and reasoning by providing justification that supports the decision.	5	Excellent		
	4	Good		
	2	Average		
	1	Poor		
TOTAL	20			

TABLE OF CONTENT

1.0 TASKS AND DELIVERABLES

You are required to upload the report and source code to the eLearn. Only the team leader needs to do the submission.

The submission should include:

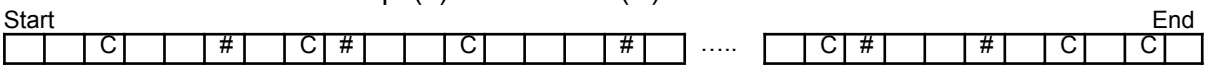
- Cover page with the team group ID, team members name and student ID
- Class and class relationship diagram for all the classes in the system
- Screenshots with description to demonstrate the test results
- Explanation or justification of the additional functionality
- Source code (upload the zipped project folder)
  Note: Make sure all the java source files (.java) are included. Submission with only the *.class file will be given zero mark*

2.0 BOAT RACING GAME

You are required to build a simple game 'Boat Race' in Java program that fulfills the below requirements. Analyze and develop the Java program as described using the Object-Oriented design. You should design your program for optimum maintainability and reusability with the best practices of object-oriented techniques you have learned. You also need to document your design using the UML class and class relationship diagrams.

The game rules:

- The game is a two-player game. At the beginning of the game, each player will be allocated a boat. During the game, the players take turns throwing the dice (you can use the random function to generate the random dice number) to decide how many steps should the boat move forward.
- The river can be visualized as a 100-column track as below, which is filled with a random number of traps(#) and currents(C).



- Once the game started, all the traps and currents will be scattered randomly in the river. Some currents are stronger than others, and so as the traps. The stronger current or trap will make the boat moves more steps forward or backward. When the boat hits the trap, the boat will need to move backward x number of steps, when the boat hits the current, it will move forward x number of steps. The boat should not be allowed to move beyond the river's boundary.

- Game will end when either player's boat reaches the end of the river. Display the location of the boats after every move.

When the game starts, display the Top 5 scores and ask the player for the name (short name with one word). You should count the total turns that each player takes in the game. When the game ended and the score of the player is within the top 5 scores, store the player's score and name in the 'TopScore.txt' text file. The list should be ordered by score in ascending order.

Tips: You can add any additional attributes to the objects in this game that you see fit

3.0 CLASSES

Total classes: 9

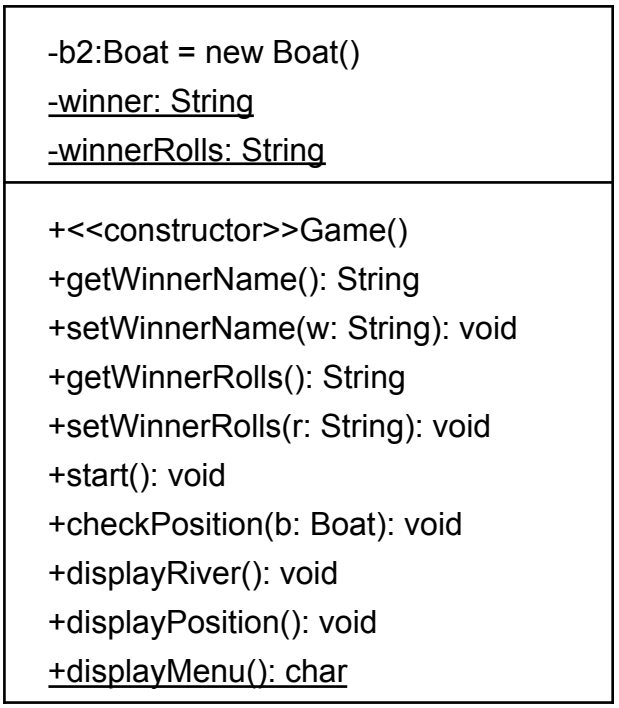
Boat
-name: String -position: int = 0 -d: Dice = new Dice()
+getName(): String +setName(int:num): void +getPosition(): int +setPosition(strength:int): void +getNumRolls(): int +rollDice(): void +reCalculate(value: int): void

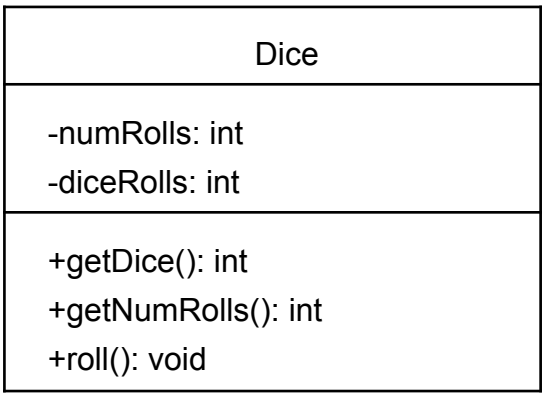
Current
+<<constructor>>Current()

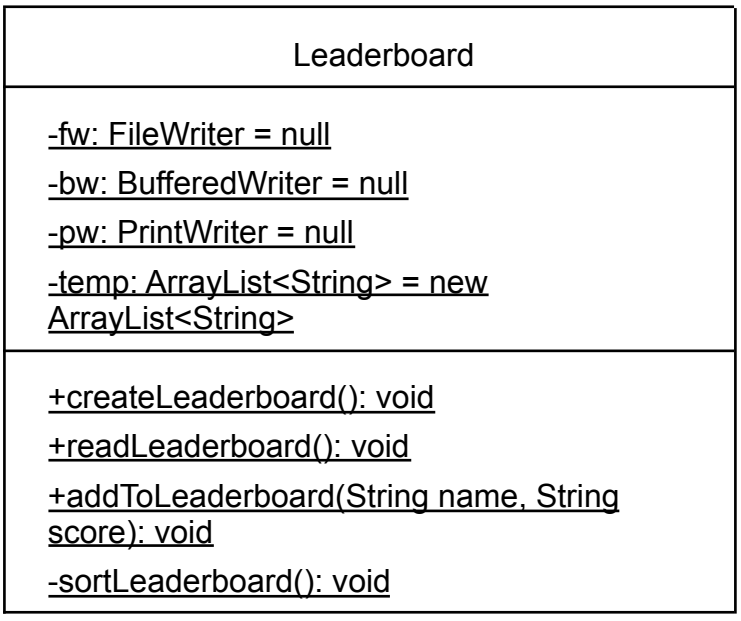
Trap
+<<constructor>>Trap()

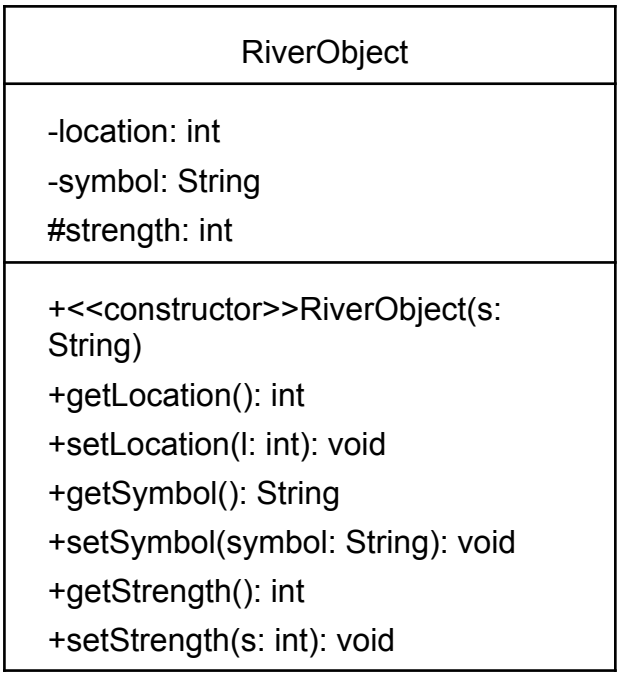
Fish
+<<constructor>>Fish()

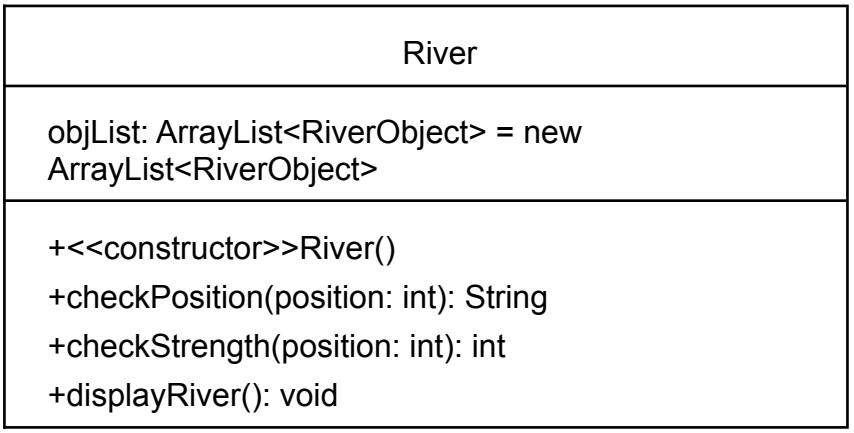
Game
-river:River -b1:Boat = new Boat()


-b2:Boat = new Boat()
-winner: String
-winnerRolls: String
+<<constructor>>Game()
+getWinnerName(): String
+setWinnerName(w: String): void
+getWinnerRolls(): String
+setWinnerRolls(r: String): void
+start(): void
+checkPosition(b: Boat): void
+displayRiver(): void
+displayPosition(): void
+displayMenu(): char

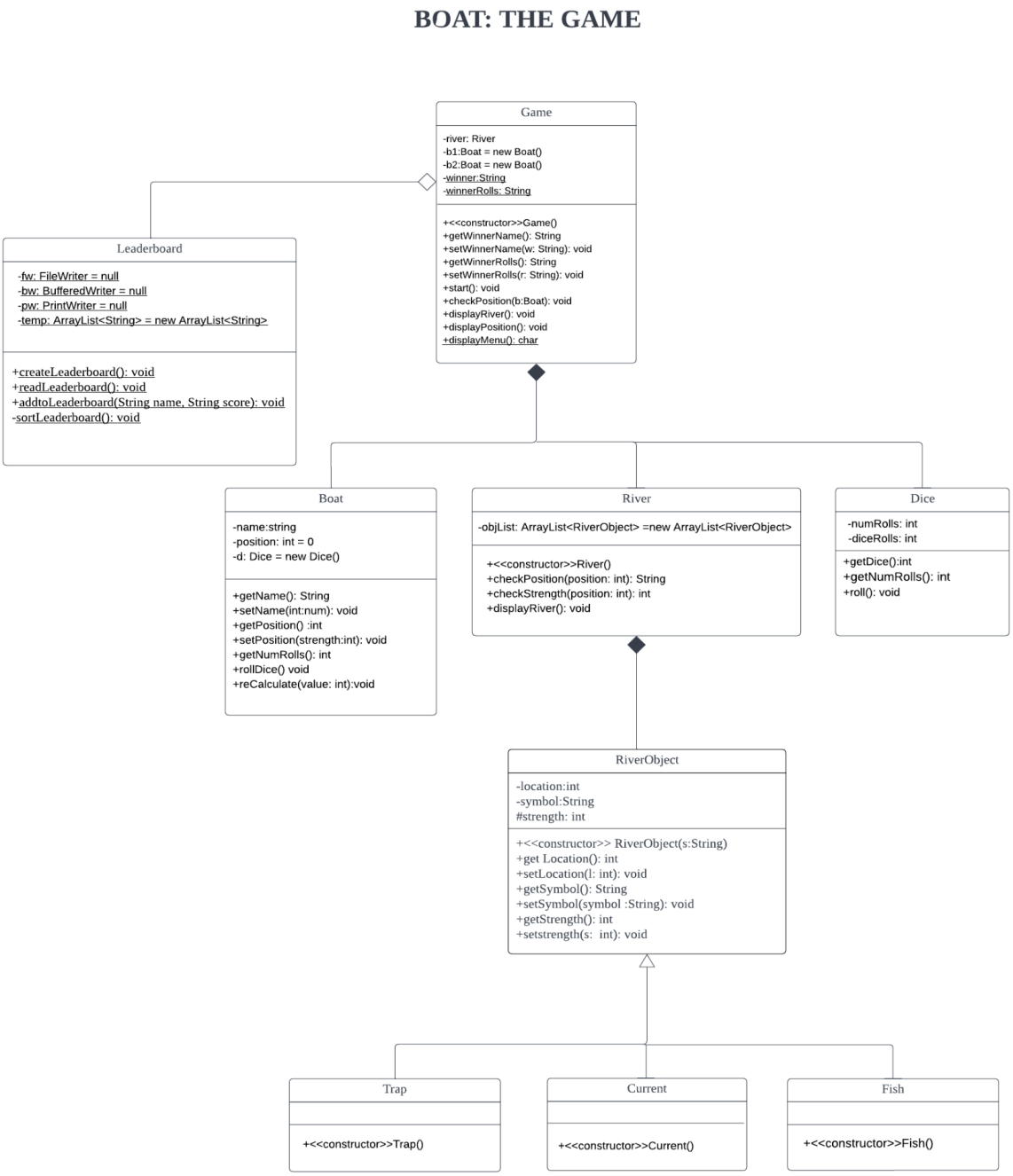

Dice
-numRolls: int
-diceRolls: int
+getDice(): int
+getNumRolls(): int
+roll(): void


Leaderboard
-fw: FileWriter = null
-bw: BufferedWriter = null
-pw: PrintWriter = null
-temp: ArrayList<String> = new ArrayList<String>
+createLeaderboard(): void
+readLeaderboard(): void
+addToLeaderboard(String name, String score): void
-sortLeaderboard(): void


RiverObject
-location: int
-symbol: String
#strength: int
+<<constructor>>RiverObject(s: String)
+getLocation(): int
+setLocation(l: int): void
+getSymbol(): String
+setSymbol(symbol: String): void
+getStrength(): int
+setStrength(s: int): void


River
objList: ArrayList<RiverObject> = new ArrayList<RiverObject>
+<<constructor>>River()
+checkPosition(position: int): String
+checkStrength(position: int): int
+displayRiver(): void

4.0 UML CLASS DIAGRAM



UML (Unified Modeling Language) is one of many ways of visualizing a software program by incorporating a collection of diagrams which is known as UML Class Diagram. As there are many types of UML diagrams, for this assignment, Class Diagram was used. Class diagrams typically consist of classes, interfaces, associations as well as collaborations. The class diagram also represents the object-oriented perspective of a system. The UML Class Diagram above subsists of the classes, attributes, and methods that were created in order to complete the “Boat Racing Game”. As shown in the diagram above, there are 9 classes that we came up with for us to use as a blueprint for the coding bit of the assignment. All 9 classes are associated with each other in order to create a relationship to ease the coding process. It was very beneficial as it kept the whole coding portion of the

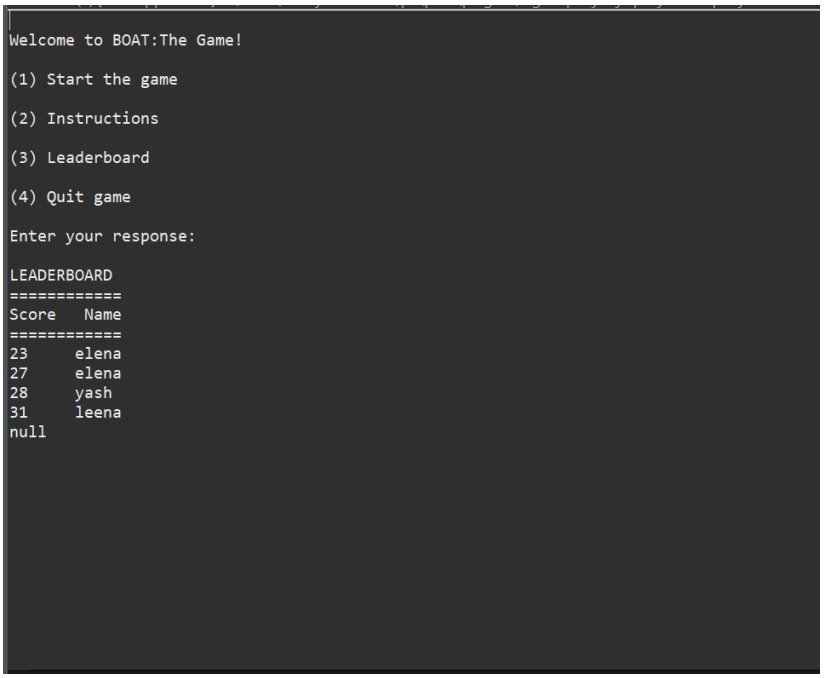
assignment well-organized and less confusing as we could always refer to the UML Class Diagram if we were thrown with confusion or doubts regarding the relationships.

5.0 ADDITIONAL FUNCTIONALITIES

Creating at least one additional functionality was one of the requirements for getting this final project completed. There are three additional functionalities that were implemented into completing the “Boat Racing Game”. By having to originate additional functionalities, creative thoughts were used on making this game unique from the rest. It also made this game extra challenging which positively affected the user experience. Following, the additional functionalities added will be elaborated in detail.

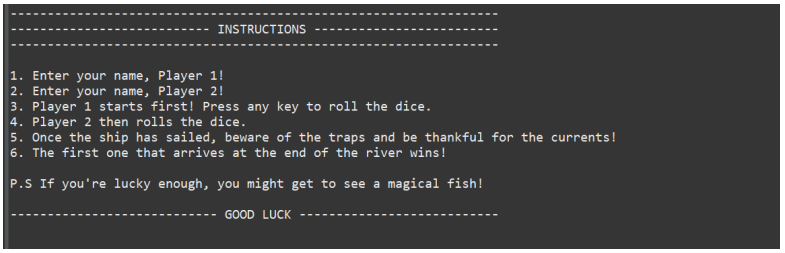
MAIN MENU - MINOR ADDITIONAL FUNCTIONALITY

The first minor additional functionality that was added is *Main Menu*. *Main Menu* is important for a game as it allows the players to make a decision if they want to start the game right away, read the instructions, take a look at the leaderboard, or quit the game. This makes the game user-friendly as well as interactive. When the player inputs the number, “***1***”, the game starts right away. This option is given to the players who are already familiar with the concept and instructions of this game so that they do not have to waste their time going through the instructions each time they want to play the game. Next, when the player inputs the number, ***“2”***, they will be prompted to the instructions page. When the player inputs the number, ***“3”***, the program will print the leaderboard of the Top 5 players arranged in ascending order. Lastly, when the user inputs the number ***“4”***, the whole program will end as the player chooses to quit the game.



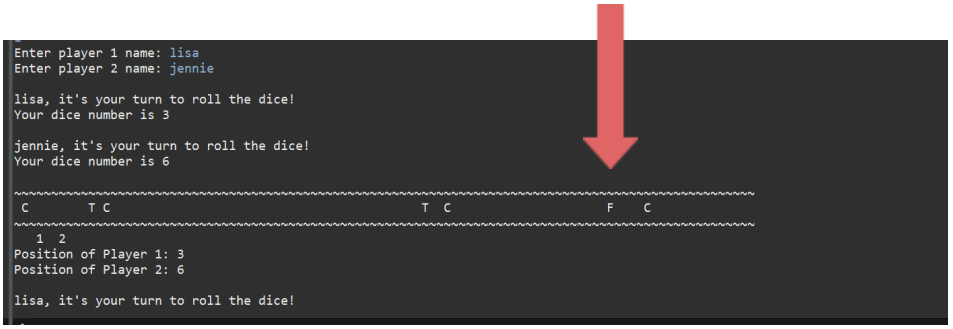
INSTRUCTIONS - MINOR ADDITIONAL FUNCTIONALITY

One of the minor additional functionalities that were added to this “Boat Racing Game" is ***instructions***. When it comes to playing a game, understanding the game ris one of the most important things as, without basic knowledge or understanding about the game, the players will have difficulty playing as they would not know what is going on. Therefore, it is mandatory to have ***instructions*** as one of the functionalities as it can be a guide for the players to understand the concept of the game before playing.



MAGICAL FISH - MAJOR ADDITIONAL FUNCTIONALITY

To make this game extra fun and interactive, a major additional functionality was added, which is the ***magical fish***. As we all know, players would always expect something different to enjoy while playing a game. To brief on the functionality of the ***magical fish***, it appears on the river together with Trap and Current. What the ***magical fish*** does is that it allows a player to roll the dice once more upon landing to increase the chance of winning. To make it more interesting, a very limited number of ***magical fish*** were added so that the probability for the players to land on it is not too high. By getting an extra roll of the dice, the player will have a higher chance of winning compared to the other player who did not land on the ***magical fish***. The positions of the ***magical fish*** are also randomized for each round which adds more fun for the players. The ***magical fish*** is represented by the letter ‘**F**’.



6.0 TEST CASES & TEST RESULTS

MAIN MENU

SCENARIO 1: PLAYER CHOOSES NUMBER 1

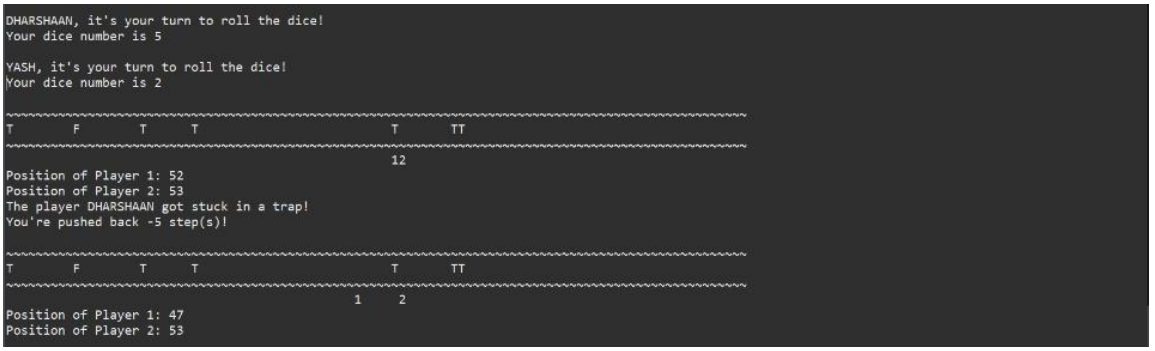
1) Player 1 and 2 insert their name

2) Player 1 rolls the dice by pressing enter key

3) Player 2 rolls the dice by pressing enter key

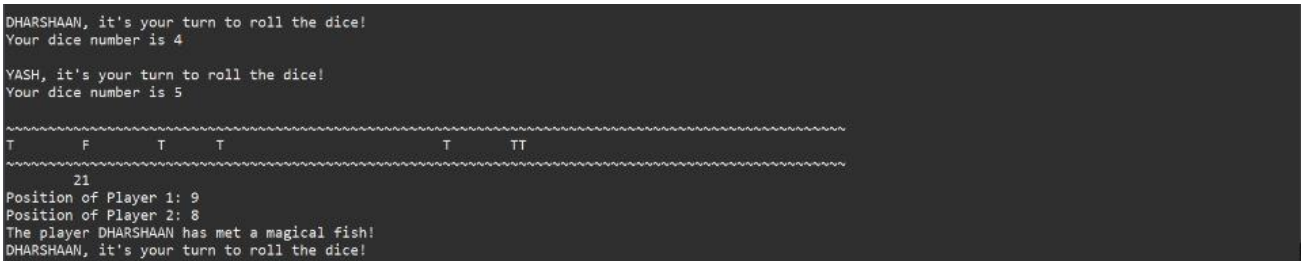
4) Player 1 and 2 moving on the river

5) If players get stuck in a trap

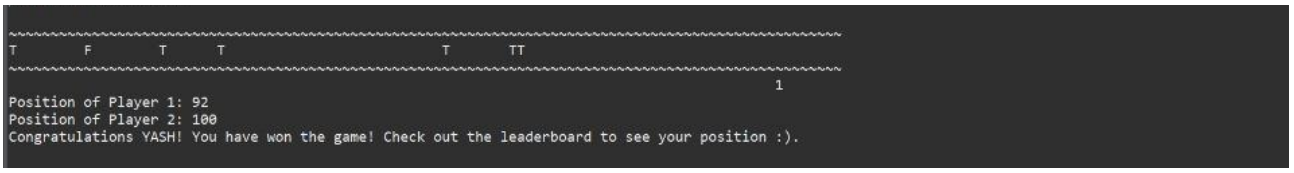


6) If players get carried forward in a current

7) If players meet a magical fish



8) Player reaches the end of the river and wins the game



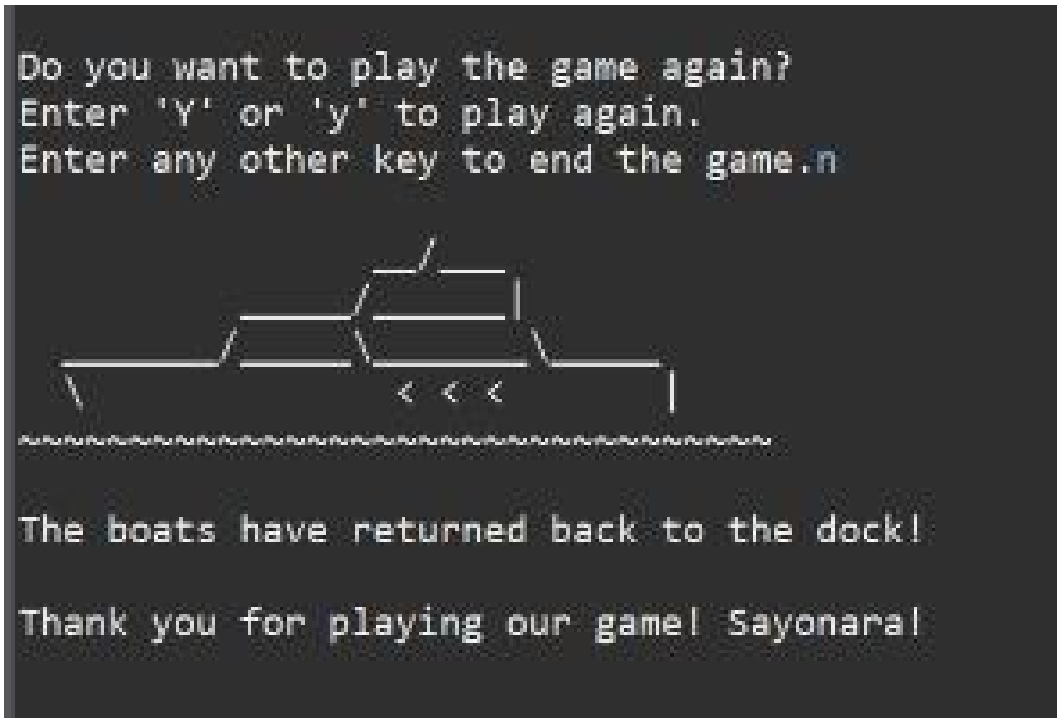
9) Player that wins the game gets a position on the Leaderboard

10) Game asks whether they want to play again

11) **If the players enter ‘Y’ (Yes, they want to play again)**

```
Do you want to play the game again?
Enter 'Y' or 'y' to play again.
Enter any other key to end the game.y

Welcome to BOAT:The Game!

(1) Start the game

(2) Instructions

(3) Leaderboard

(4) Quit game

Enter your response:

LEADERBOARD
============
Score   Name
============
23      elena
27      elena
28      yash
31      leena
35      yash
```

12) If the players enter an input other than ‘Y’ (No, they want to end the game)

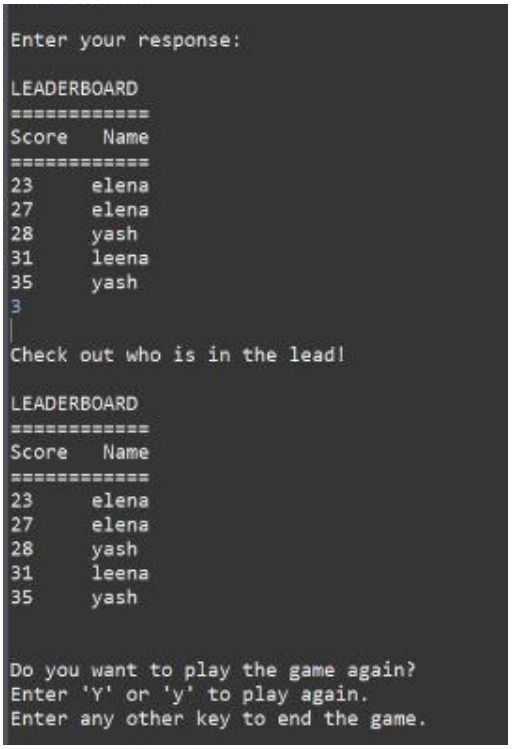


SCENARIO 2: PLAYER CHOOSES NUMBER 2

1) **The players will see the instructions**

SCENARIO 3: PLAYER CHOOSES NUMBER 3

1) **The players will see the leaderboard for the TOP 5 players (arranged in ascending order)**



SCENARIO 4: PLAYER CHOOSES NUMBER 4

1) **The whole program will quit**

```
4
That's a pity :( . It's alright, you can always play again next time! Adios, mi amigo!
```

SCENARIO 5: PLAYER CHOOSES OTHER THAN 1, 2, 3 OR 4

1) **A message will be printed asking the user to key in a valid response**

```
Welcome to BOAT:The Game!

(1) Start the game

(2) Instructions

(3) Leaderboard

(4) Quit game

Enter your response:

LEADERBOARD
============
Score   Name
============
29      YASH
33      YASH
null
null
null
hello
Enter a valid response!:
```

7.0 SCENARIO SUMMARY

To summarize the “Boat Racing Game”, once the game starts, the players will get to see the display of the “**Main Menu**” page. Players will then have to decide on what they are going to type in for the input as there are four possible inputs that they can put in. If the player wants to start the game, the player has to input **1**. The players will then have to fill up their usernames in “**Player 1**” and “**Player 2**” sections. Then the game will start by asking the players to press ***“ENTER”*** in order to roll the dice. The game goes on and if any one of the players land on ***TRAP***, they will move back a few steps as not every ***TRAP*** has the same number of steps. Then if the players land on ***CURRENT,*** the players will be pushed a few steps ahead according to the system as it is also set to be randomized each time with a different number of steps. Lastly, if the players land on a ***MAGICAL FISH*** they are given a second chance to roll the dice again. This can be both good or bad for the players according to whatever situation they are in as if they get to re-roll the dice and land on ***TRAP*** they will move back a few steps. The winner's name will then be printed on the Leaderboard once they reach the end with the score beside their name. Once the game ends, the player can either enter “**Y**” or “**y**” to go back to the main menu or they can enter any other key to quit the game. If they quit the game, then the system prints out a message for the players and the program will end. On the **Main Menu**, the players also have an option to view the game instructions by inputting **2** or view the leaderboard by inputting **3**. If the players want to quit the game, they can input **4** to end the program.