SUNWAY UNIVERSITY

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: 15 JULY 2022 11:59PM

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

“I AISHA SOFIA BINTI NAJIDI ………………, CHAN XUAN YING …………………, CHONG KAR YAN …………………, EMILY TENG JIE QI ………………, JUSTIN PHANG SHENG XUAN ………………… (student name). 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.” [illegible] ………………, [illegible] ………………, [illegible] ……. ……, [illegible] ………………, [illegible] ……………
(Student’s signature / Date)

Group Number: **G6**

Team Members:

No	**Name**	**Student ID**	**Contribution (%)**
1	AISHA SOFIA BINTI NAJIDI	20065231	20
2	CHAN XUAN YING	21041686	20
3	CHONG KAR YAN	20072310	20
4	EMILY TENG JIE QI	20054607	20
5	JUSTIN PHANG SHENG XUAN	20066502	20

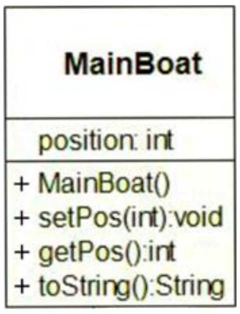
<u>Marking Scheme</u>

Criteria	Reference Marks		Marks	Remarks
<u>Design (10%)</u> 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		
<u>Coding (5%)</u> 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			

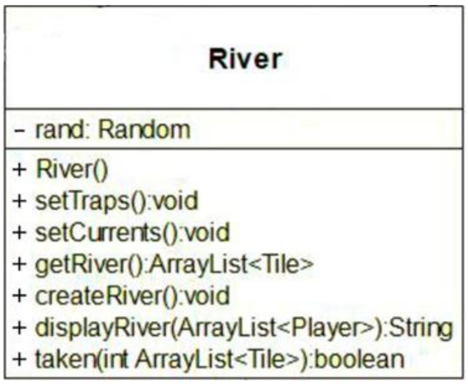
Table of Contents

UML Class Diagram

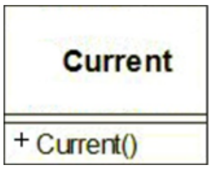
Main Boat Class:



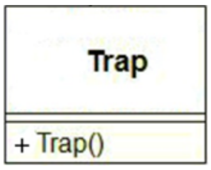
River Class:


River
- rand: Random
+ River()
+ setTraps():void
+ setCurrents():void
+ getRiver():ArrayList<Tile>
+ createRiver():void
+ displayRiver(ArrayList<Player>):String
+ taken(int ArrayList<Tile>):boolean

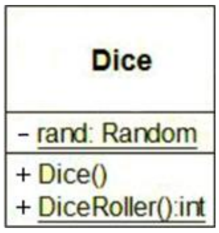
Current Class:



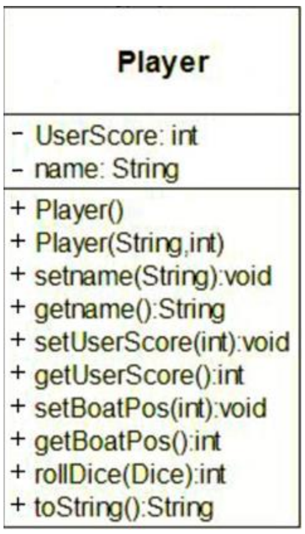
Trap Class:



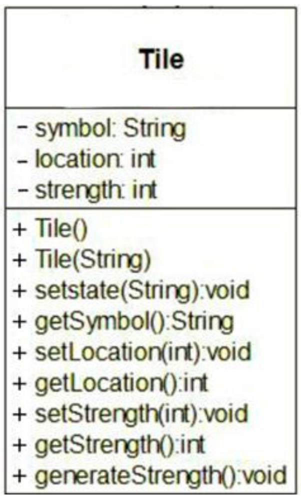
Dice class:



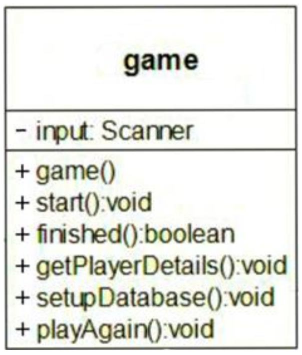
Player Class:



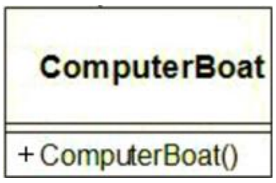
Tile Class:



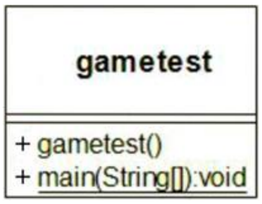
Game Class:



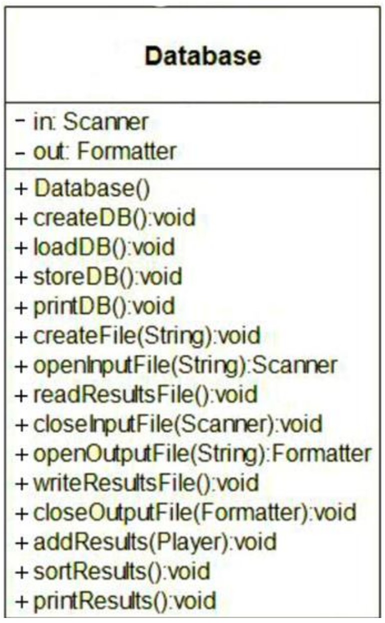
Computer Boat Class:



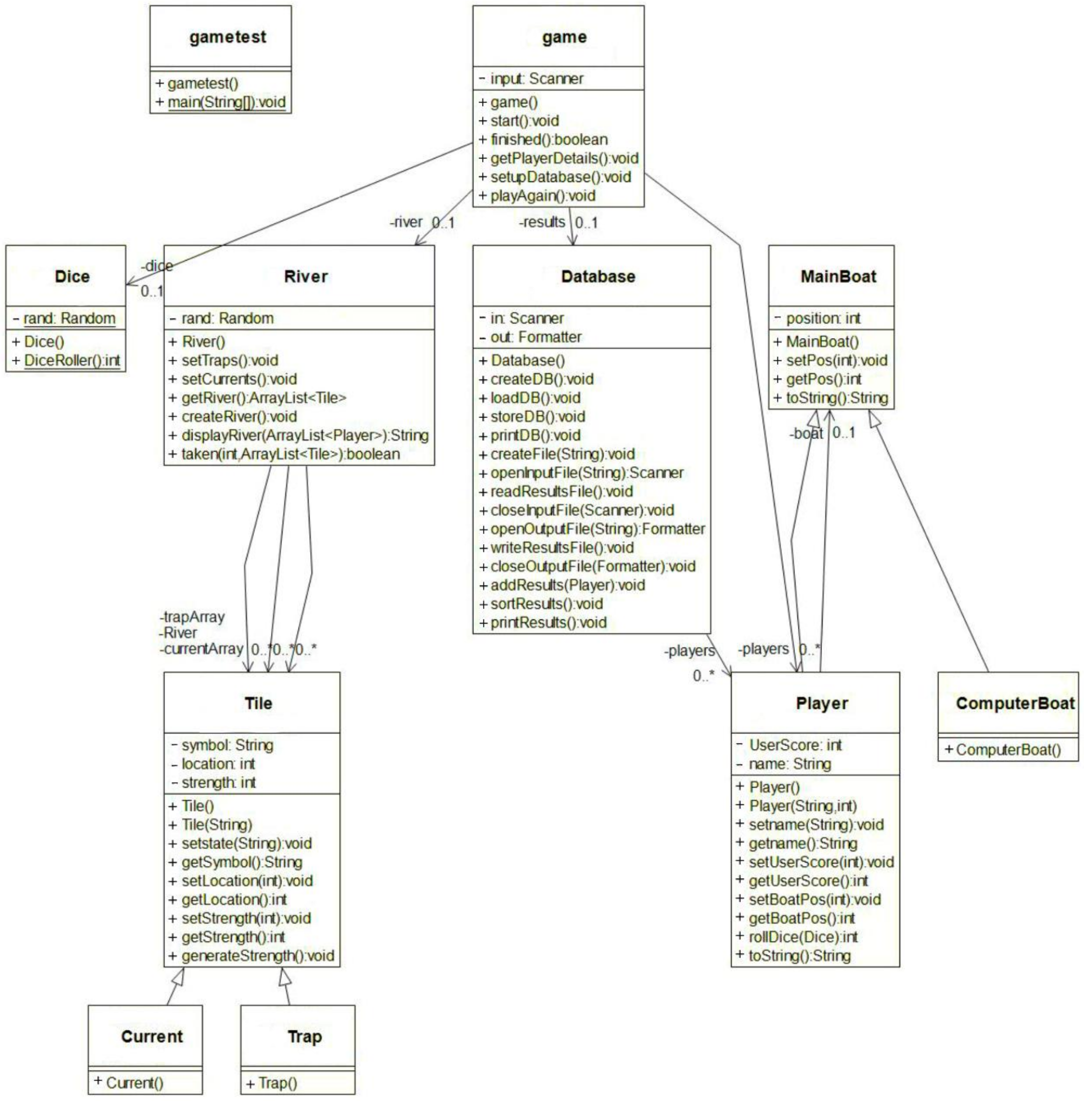
Game test:



Database:

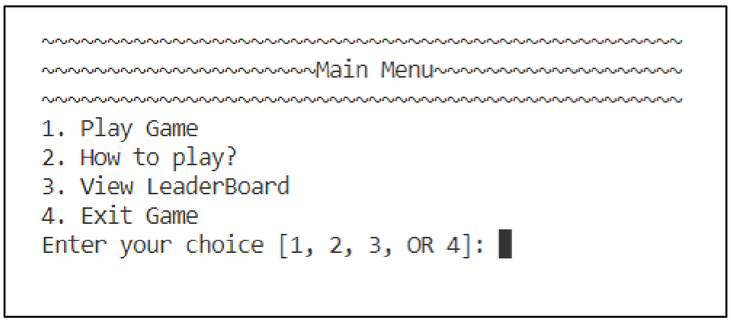

Database
- in: Scanner
- out: Formatter
+ Database()
+ createDB():void
+ loadDB():void
+ storeDB():void
+ printDB():void
+ createFile(String):void
+ openInputFile(String):Scanner
+ readResultsFile():void
+ closeInputFile(Scanner):void
+ openOutputFile(String):Formatter
+ writeResultsFile():void
+ closeOutputFile(Formatter):void
+ addResults(Player):void
+ sortResults():void
+ printResults():void

Class Relationship Diagram



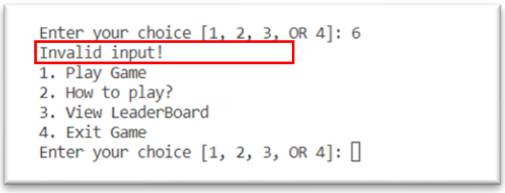
Demonstration

1.1 Main Menu



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~Main Menu~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1. Play Game
2. How to play?
3. View LeaderBoard
4. Exit Game
Enter your choice [1, 2, 3, OR 4]: 
```

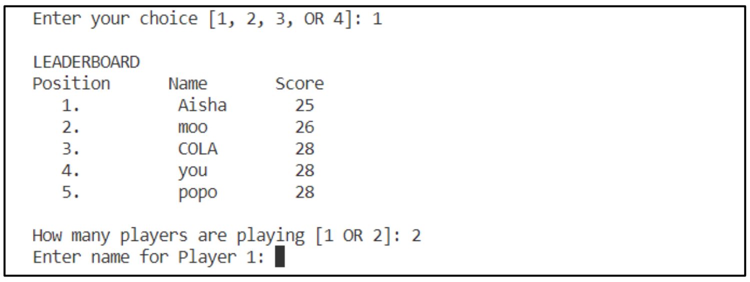
This is the landing page that consists of the main menu of the game. When users run the program, it will land them on this main menu where users are allowed to choose which function they would like to execute. In this game, the selection of choices is from 1 to 4. If users choose a number other than that, it will display an error message and the menu for users to renter the valid input.



```
Enter your choice [1, 2, 3, OR 4]: 6
Invalid input!
1. Play Game
2. How to play?
3. View LeaderBoard
4. Exit Game
Enter your choice [1, 2, 3, OR 4]: 
```

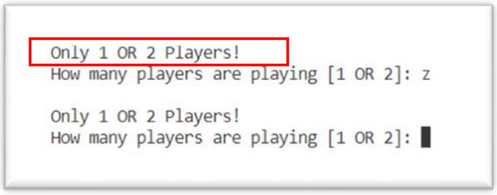
An error message will be printed to notify users that their choice is invalid.

1.2 Play Game



```
Enter your choice [1, 2, 3, OR 4]: 1

LEADERBOARD
Position      Name      Score
   1.          Aisha      25
   2.          moo        26
   3.          COLA       28
   4.          you        28
   5.          popo       28

How many players are playing [1 OR 2]: 2
Enter name for Player 1: 
```

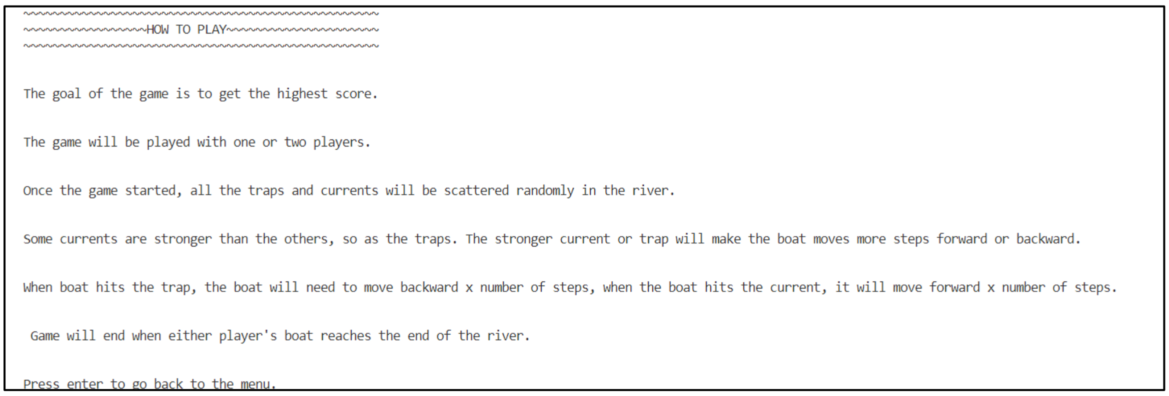
When the user input is ‘1’, it will start the game by displaying a leader board that tracks the scores of previous players. Users will be prompted to input the number of players followed by their names.



```
Only 1 OR 2 Players!
How many players are playing [1 OR 2]: z

Only 1 OR 2 Players!
How many players are playing [1 OR 2]: 
```

An instruction message will be printed to limit the answers to this question.

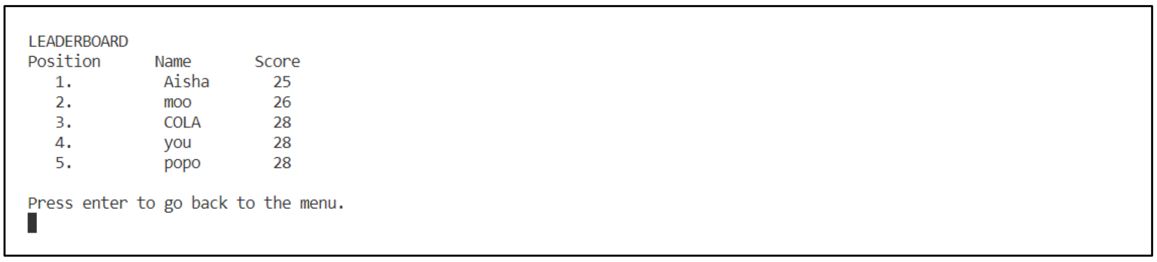
1.3 How to play?



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~HOW TO PLAY~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


The goal of the game is to get the highest score.


The game will be played with one or two players.


Once the game started, all the traps and currents will be scattered randomly in the river.


Some currents are stronger than the others, so as the traps. The stronger current or trap will make the boat moves more steps forward or backward.


When 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.


 Game will end when either player's boat reaches the end of the river.


Press enter to go back to the menu.
```

This section is created for the new users to learn the instructions before starting the game. The information and details about the boat racing game are written here as well. After reading the tutorials, users can press enter to return to the main menu and make another selection.

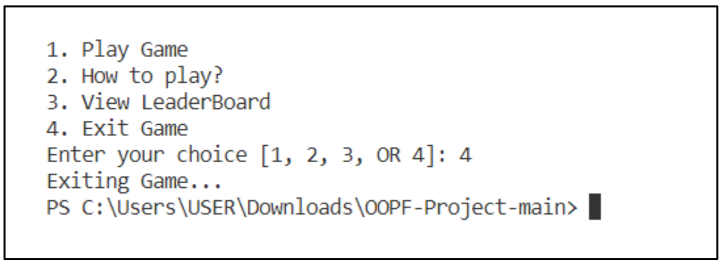
1.4 View LeaderBoard



```
LEADERBOARD
Position      Name       Score
   1.          Aisha       25
   2.          moo         26
   3.          COLA        28
   4.          you         28
   5.          popo        28

Press enter to go back to the menu.
```

In order to provide a user-friendly browsing experience, users are allowed to view the list of ranking, namely leaderbroad from the main menu. This leaderboard displays that the scores of the player that are within the top 5 scores. It stores players' details such as position, names, and scores in ascending order. Similar to the tutorial section, the way to go back to the main menu is by pressing enter.

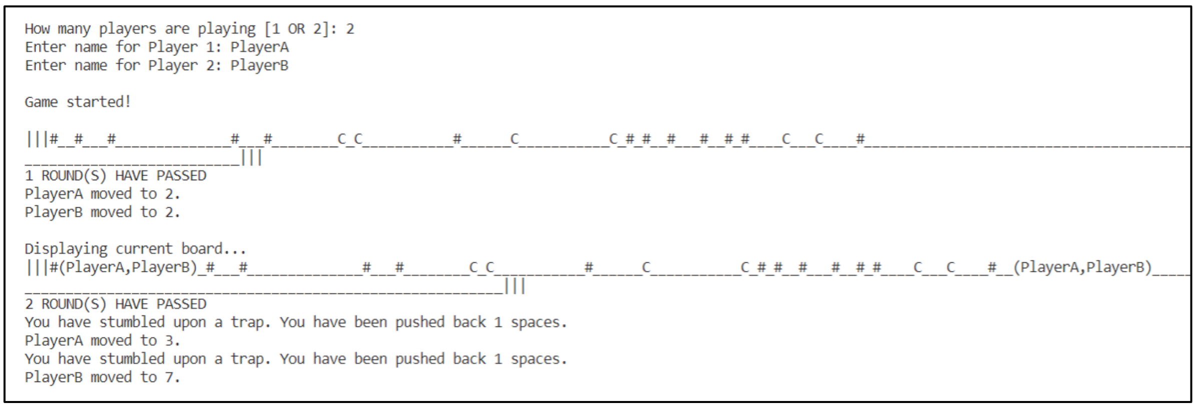
1.5 Exit Game



```
1. Play Game
2. How to play?
3. View LeaderBoard
4. Exit Game
Enter your choice [1, 2, 3, OR 4]: 4
Exiting Game...
PS C:\Users\USER\Downloads\OOPF-Project-main>
```

To exit the game and application, users can enter '4' in the main menu.

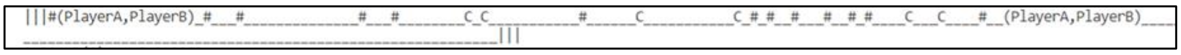
1.6 Boat Racing



```
How many players are playing [1 OR 2]: 2
Enter name for Player 1: PlayerA
Enter name for Player 2: PlayerB

Game started!

|||#_#__#_____________#__#_______C_C__________#_____C__________C_#_#__#__#__#_#___C__C___#________________________________________
_________________________|||
1 ROUND(S) HAVE PASSED
PlayerA moved to 2.
PlayerB moved to 2.

Displaying current board...
|||#(PlayerA,PlayerB)_#__#_____________#__#_______C_C__________#_____C__________C_#_#__#__#__#_#___C__C___#__(PlayerA,PlayerB)____
_________________________________________________________|||
2 ROUND(S) HAVE PASSED
You have stumbled upon a trap. You have been pushed back 1 spaces.
PlayerA moved to 3.
You have stumbled upon a trap. You have been pushed back 1 spaces.
PlayerB moved to 7.
```

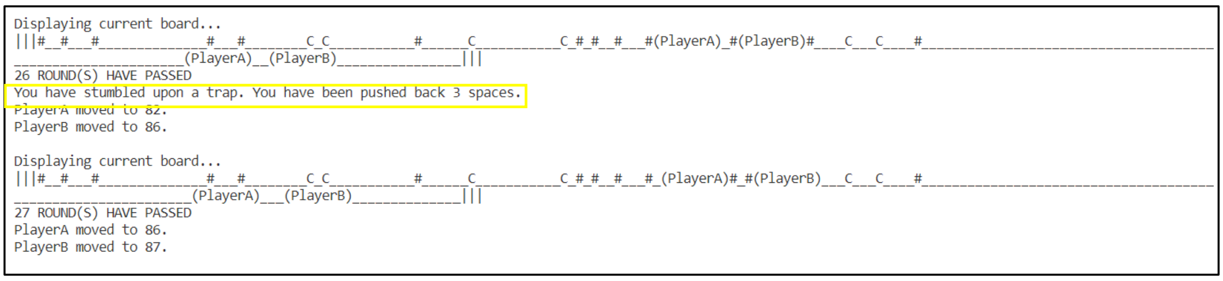
This is the interface when the boat racing game starts. It shows a river that is scattered with traps and currents. These traps and currents are represented by the symbol '#' and letter 'C' respectively. And the river boundary is visualised by using 3 vertical lines (|) at the starting point and endpoint.

After every move of each player, the locations of boats which are named by their name will be displayed for a better visualisation experience.



```
|||#(PlayerA,PlayerB)_#__#_____________#__#_______C_C__________#_____C__________C_#_#__#__#__#_#___C__C___#__(PlayerA,PlayerB)____
_________________________________________________________|||
```

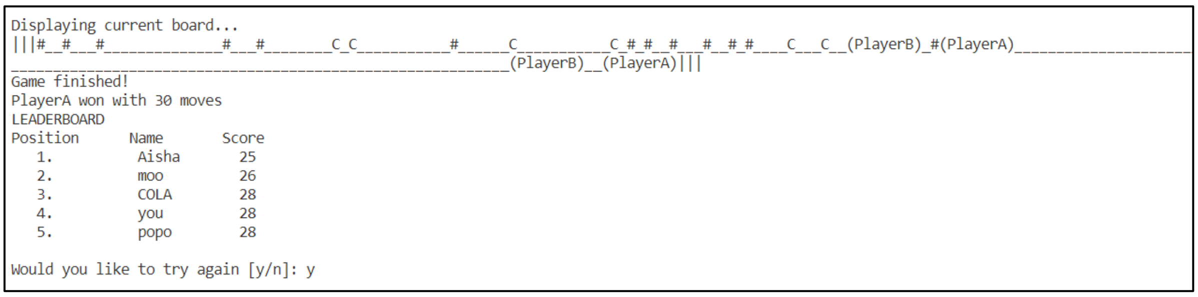
If the round is a two-player game, the players need to take turns throwing the dice so that they can decide how many steps to move the boat forward. During the game, the boat will move forward 2 spaces when it hits the current but be pushed back by a random number when it stumbled upon a trap. An alert message will be printed to inform how many spaces it has been pushed back.



```
Displaying current board...
|||#_#__#_____________#__#_______C_C__________#_____C__________C_#_#__#__#(PlayerA)_#(PlayerB)#___C__C___#__________________________________
____________________(PlayerA)__(PlayerB)_______________|||
26 ROUND(S) HAVE PASSED
You have stumbled upon a trap. You have been pushed back 3 spaces.
PlayerA moved to 82.
PlayerB moved to 86.

Displaying current board...
|||#_#__#_____________#__#_______C_C__________#_____C__________C_#_#__#__#_(PlayerA)#_#(PlayerB)___C__C___#________________________________
_____________________(PlayerA)___(PlayerB)_____________|||
27 ROUND(S) HAVE PASSED
PlayerA moved to 86.
PlayerB moved to 87.
```

In the 26th round, PlayerA had hit a trap and it has been pushed back by 3 spaces.

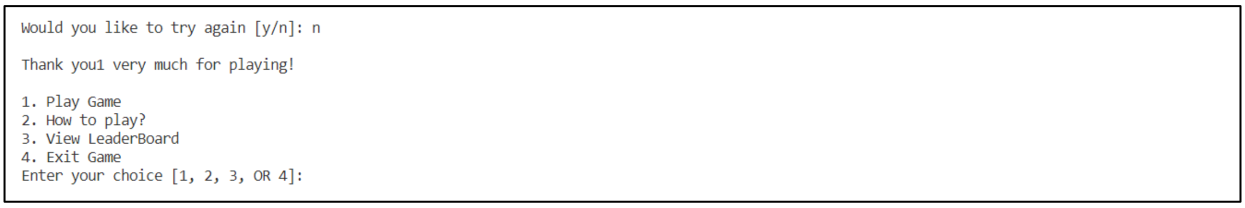
1.7 End Game



```
Displaying current board...
|||#__#___#______________#___#________C_C___________#______C___________C_#_#__#___#__#_#____C___C__(PlayerB)_#(PlayerA)____________________
___________________________________________________________(PlayerB)__(PlayerA)|||
Game finished!
PlayerA won with 30 moves
LEADERBOARD
Position      Name      Score
   1.          Aisha      25
   2.          moo        26
   3.          COLA       28
   4.          you        28
   5.          popo       28

Would you like to try again [y/n]: y
```

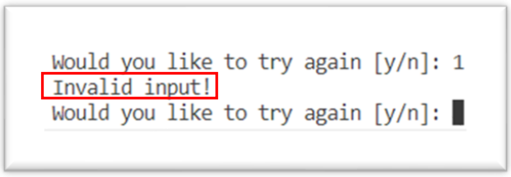
PlayerA reaches the endpoint and thus the game ends.

The design for the interface when the game ends. It displays the leader board with the latest records of players that are within the top 5.



```
Would you like to try again [y/n]: n

Thank you1 very much for playing!

1. Play Game
2. How to play?
3. View LeaderBoard
4. Exit Game
Enter your choice [1, 2, 3, OR 4]:
```

To stop playing the game, users can enter ‘n’ for the last question. It will then print a thank-you message and direct the user back to the main menu.

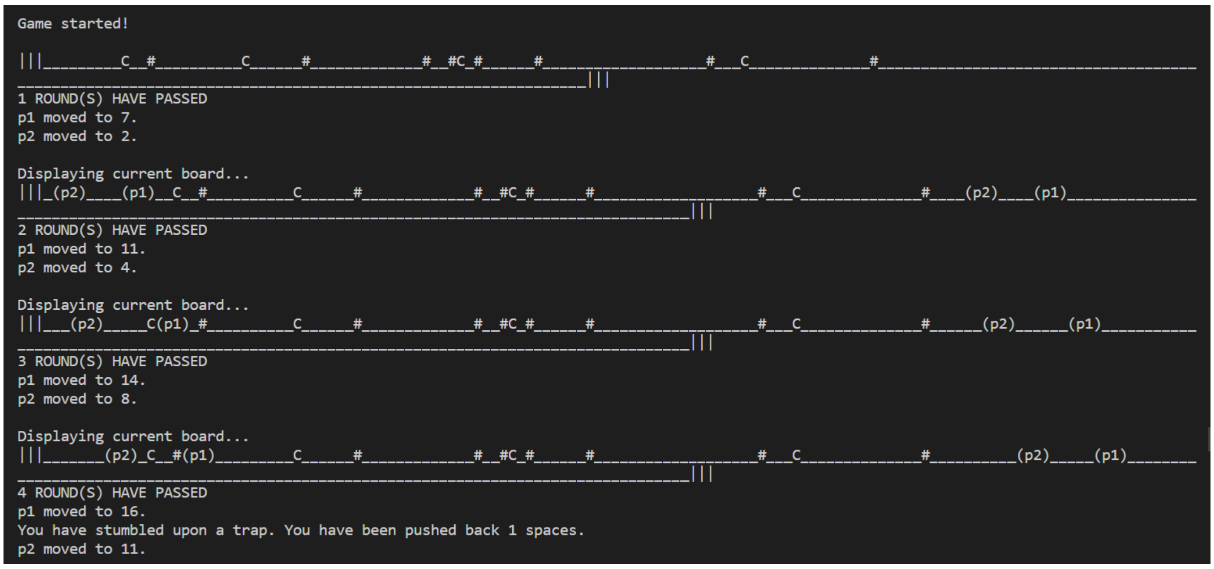


```
Would you like to try again [y/n]: 1
Invalid input!
Would you like to try again [y/n]:
```

A validation check is applied here to ensure the user enters only yes or no.

Additional Functionality

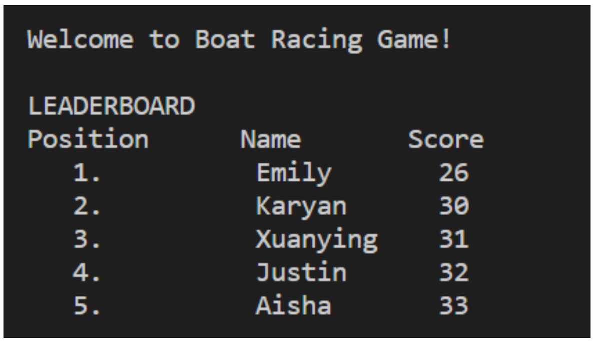
The additional functionality added to our project is the graphical user interface (GUI) application. It is made up of tables and labels to enhance the visual experience of the users and allow easy interaction while they are playing the game. The GUI can be implemented on the terminal and will be displayed as the program started running. For example, instead of merely returning an integer to indicate the position of players and obstacles, our program included GUI in the gameplay, in which the obstacles and player position can easily be recognized along the sequences of underscore symbols. This feature improves the readability of the game, where the visuals are beneficial for users of any level to comprehend and understand the game's progress.



```
Game started!

|||________C__#_________C_____#___________#__#C_#_____#________________#__C___________#_______________________________
____________________________________________________________|||
1 ROUND(S) HAVE PASSED
p1 moved to 7.
p2 moved to 2.

Displaying current board...
|||_(p2)____(p1)__C__#_________C_____#___________#__#C_#_____#________________#__C___________#____(p2)____(p1)_____________
_______________________________________________________________________|||
2 ROUND(S) HAVE PASSED
p1 moved to 11.
p2 moved to 4.

Displaying current board...
|||___(p2)_____C(p1)_#_________C_____#___________#__#C_#_____#________________#__C___________#______(p2)______(p1)__________
_______________________________________________________________________|||
3 ROUND(S) HAVE PASSED
p1 moved to 14.
p2 moved to 8.

Displaying current board...
|||_______(p2)_C__#(p1)________C_____#___________#__#C_#_____#________________#__C___________#_________(p2)_____(p1)_______
_______________________________________________________________________|||
4 ROUND(S) HAVE PASSED
p1 moved to 16.
You have stumbled upon a trap. You have been pushed back 1 spaces.
p2 moved to 11.
```

GUI of the gameplay

The GUI of leader board is implemented in the form of table, consisting of 3 columns and 6 rows. As the program takes records of each player's results after each game, the leader board displays the ranking of the top 5 players with the player names and scores clearly stated in the table.



```
Welcome to Boat Racing Game!

LEADERBOARD
Position       Name       Score
   1.           Emily       26
   2.           Karyan      30
   3.           Xuanying    31
   4.           Justin      32
   5.           Aisha       33
```

GUI of leader board