

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 (FRIDAY)

GROUP: 14

Academic Honesty Acknowledgement

"We Jayshiv, Azri, Elena, & Ashwin,
verify that this paper contains entirely our own work. We 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, we have not copied or inadvertently copied ideas, sentences, or paragraphs from another student. We realize the penalties (*refer student handbook undergraduate programme*) for any kind of copying or collaboration on any assignment."

Jayshiv, Azri, Elena, & Ashwin (26/11/2022)

No	Name	Student ID	Contribution (%)
1	JAYSHIV KAMANI A/L RAJESHKUMAR	22017040	25%
2	MUHAMMAD AZRI BIN MOHD YASSIN	20035853	25%
3	ELENA KHOO SZE KAY	21012570	25%
4	ASHWIN VIGNESWARAN	21009527	25%

TABLE OF CONTENTS

1.0 TASKS & DELIVERABLES	<u>2</u>
2.0 BOAT RACING GAME	<u>2</u>
3.0 CLASSES	<u>3</u>
4.0 UML CLASS DIAGRAM	<u>7</u>
5.0 ADDITIONAL FUNCTIONALITIES	<u>8</u>
6.0 TEST CASES & RESULTS	<u>13</u>

1.0 Tasks & Deliverables

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)

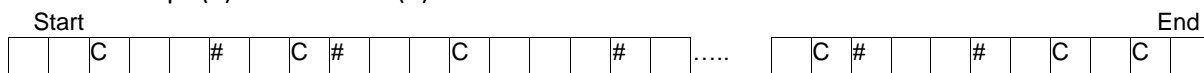
Note: Make sure all the java source files (*.java) are included. Submission with only the *.class file will be given zero mark.

2.0 Boat Racing Game

You are required to build a simple game 'Boat Race' in Java program that fulfil the below requirements. Analyse and develop the Java program as per 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 learnt. 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 visualised as 100-columns track as below, which is filled with 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 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. 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 games. 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.


Additional Functionality:

Design and develop one additional function that may help to improve the game you have developed above.

3.0 Classes


 Dice
-diceNo:Random = new Random()
+generateDiceNo():int


 Scoreboard
-scoreboardOutput:Formatter -scoreboardInput:Scanner -contestants:ArrayList<Contenstant> = new ArrayList<Contestant>
+orderScores():void +printScoreboardData():void +generateFile(n:String):void +openInputScannerFile(n:String):Scanner +readTopScoresFile():void +closeInputScannerFile(i:Scanner):void +openOutputFormatterFile(n:String):Formatter +insertTopScoresFile():void +closeOutputFormatterFile(o:Formatter):void +addScoresToList(c:C.Contestant):void +generateScoreboard():void +captureScoreboard():void +loadScoreboard():void +displayScoreboard():void




 Boat
-boatLocation:int
+<<constructor>>Boat() +setLocationOfBoat(bLocation:int):void +getLocationOfBoat():int +toString():String

Game
-scoreboard:Scoreboard = new Scoreboard() -contestants:ArrayList<Contestant> = new ArrayList<Contestant> -riverStream:River = new River() -dice:Dice = new Dice()
+displayInstructions():void +startGame():void +endGame():boolean +feedback():void +checkContestantDetails():void +openScoreboard():void +drawBanner():void

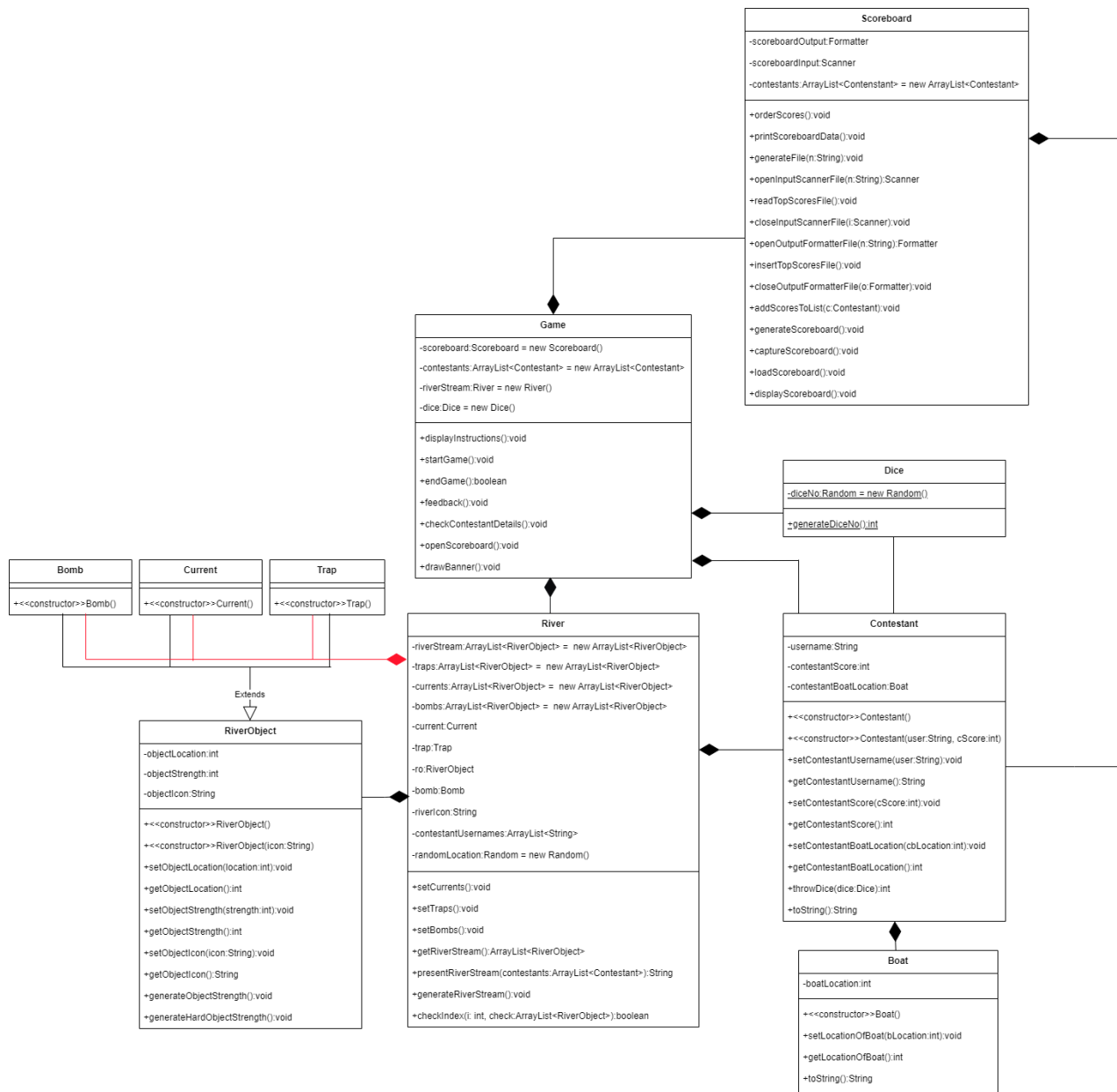
River
-riverStream:ArrayList<RiverObject> = new ArrayList<RiverObject> -traps:ArrayList<RiverObject> = new ArrayList<RiverObject> -currents:ArrayList<RiverObject> = new ArrayList<RiverObject> -bombs:ArrayList<RiverObject> = new ArrayList<RiverObject> -current:Current -trap:Trap -ro:RiverObject -bomb:Bomb -riverIcon:String -contestantUsernames:ArrayList<String> -randomLocation:Random = new Random()
+setCurrents():void +setTraps():void +setBombs():void +getRiverStream():ArrayList<RiverObject> +presentRiverStream(contestants:ArrayList<Contestant>):String +generateRiverStream():void +checkIndex(i: int, check:ArrayList<RiverObject>):boolean

 RiverObject
-objectLocation:int -objectStrength:int -objectIcon:String
+<<constructor>>RiverObject() +<<constructor>>RiverObject(icon:String) +setObjectLocation(location:int):void +getObjectLocation():int +setObjectStrength(strength:int):void +getObjectStrength():int +setObjectIcon(icon:String):void +getObjectIcon():String +generateObjectStrength():void +generateHardObjectStrength():void

 Contestant
-username:String -contestantScore:int -contestantBoatLocation:Boat
+<<constructor>>Contestant() +<<constructor>>Contestant(user:String, cScore:int) +setContestantUsername(user:String):void +getContestantUsername():String +setContestantScore(cScore:int):void +getContestantScore():int +setContestantBoatLocation(cbLocation:int):void +getContestantBoatLocation():int +throwDice(dice:Dice):int +toString():String

 Bomb	 Current	 Trap
+<<constructor>>Bomb()	+<<constructor>>Current()	+<<constructor>>Trap()

4.0 UML Class Diagram



A UML diagram is a diagram based on the UML (Unified Modelling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts, or classes, in order to better understand, alter, maintain, or document information about the system. The UML diagram above consists of and shows all the classes that we created for the “Boat Racing Game”. Each class in the UML diagram is linked and has relationships to better comprehend the layout of how the code will be. It has truly benefitted us by giving a base-level understanding which aids in the coding process.

5.0 Additional Functionalities

There are a variety of additional functionalities implemented into the “Boat Racing Game” we created. Additional functionalities are essentially add-ons that are designed and developed to help improve the game in ways that were not mentioned in the initial requirements. In our game, we created four minor additional functionalities, and one major additional functionality. The functionalities mentioned below were invented to erect a user-friendly and positive experience for contestants that play the game.

Game Banner – Minor Additional Functionality (1):

The game banner constructs a boat made out of symbols as soon as the code starts running. This was created out of display purposes in a motive to give users a visual experience and feel to the game, as GUI was not allowed.



```
public void drawBanner()
{
    System.out.println("\n          /|~~~");
    System.out.println("        ///|");
    System.out.println("       /////|");
    System.out.println("      //////////////|");
    System.out.println("     \\\=====|===/");
    System.out.println("~~~~~");
}
```


Instructions – Minor Additional Functionality (2):

Guidelines or instructions are always necessary for a game as there can exist people who are not familiar with the game and how it works. Instructions give players an idea and view of how the game works before playing so that confusion or dismay does not occur. Therefore, we decided that it would be a great reason to add instructions to the game.

```
INSTRUCTIONS:
1) Firstly, a display of the river will be shown to players each round
2) The river will have about 100 cells, in which currents and traps will be placed randomly.
3) The contestant(s) will have to throw their dice in order to move along the river.
4) Contestant(s) have a choice of playing solo, or duo.
5) In order to win, you must reach the end of the river
6) Stepping on a current will push you forwards in the river.
7) However, stepping on a trap will push you backwards in the river.
8) In hard mode, watch out for bombs! They push you back really far.
9) If there's an error with the scoreboard at first run, please play the game completely for the scoreboard to appear the following rounds.
10) Good luck and have fun! Make sure to avoid traps! :)
```

```
public void displayInstructions()
{
    System.out.println("\nINSTRUCTIONS:");
    System.out.println("1) Firstly, a display of the river will be shown to players each round");
    System.out.println("2) The river will have about 100 cells, in which currents and traps will be placed randomly.");
    System.out.println("3) The contestant(s) will have to throw their dice in order to move along the river.");
    System.out.println("4) Contestant(s) have a choice of playing solo, or duo.");
    System.out.println("5) In order to win, you must reach the end of the river");
    System.out.println("6) Stepping on a current will push you forwards in the river.");
    System.out.println("7) However, stepping on a trap will push you backwards in the river.");
    System.out.println("8) In hard mode, watch out for bombs! They push you back really far.");
    System.out.println("9) If there's an error with the scoreboard at first run, please play the game completely for the scoreboard to appear the following rounds.");
    System.out.println("10) Good luck and have fun! Make sure to avoid traps! :)\n");
}
```

Feedback – Minor Additional Functionality (3):

Two-way communication is an important factor to consider when developing any social platform - including games. Receiving feedback and response can be a great way to monitor and keep track of user experience. We designed a feedback option to appear right after the game has ended so that users can have the option to write back on how certain aspects of the game could be improved.

```
Please enter 'yes' if you would like to give us feedback on the game, and 'no' if no: yes
Enter feedback here (500 characters): The game is perfect!
Great feedback! Thank you for playing Group 14's Boat Race Game! Please come again soon!
```

```
public void feedback()
{
    String response = " ";
    String feedback = " ";

    System.out.print("Please enter 'yes' if you would like to give us feedback on the game, and 'no' if no: ");
    response = input.next();

    while ((response.equals("yes") == false) && (response.equals("no") == false))
    {
        System.err.println("-- Please enter 'yes' or 'no' only! --");
        System.out.print("Please enter 'yes' if you would like to give us feedback on the game, and 'no' if no: ");
        response = input.next();
    }

    if (response.equals("yes"))
    {
        System.out.println();
        System.out.print("Enter feedback here (500 characters): ");
        feedback = input.next();

        if (feedback.length() > 500)
        {
            System.err.println("-- Please keep feedback under 500 characters! --");
            System.out.print("Enter feedback here (500 characters): ");
            feedback = input.next();
        }
        System.out.println();
        System.out.println("Great feedback! Thank you for playing Group 14's Boat Race Game! Please come again soon!");
    }
}
```

One-Player / Solo Option – Minor Additional Functionality (4):

Not everyone has a friend to play with. Players should be given the option to play solo if they like or if they have no one else to play with. Constricting a game like this to only have two players is the grounds of not promoting inclusivity and thoughtfulness. Hence, we have given contestants the option to play solo if they want to by allowing them to enter 1 or 2, to indicate whether they are playing solo or duo. Regardless of whether they enter 1 or 2, it will still be passed in the list as a contestant of the game.

```
Please enter the number of contestants (solo / duo) that are going to participate in this match (Min = 1, Max = 2): 1
Enter the username for contestant number 1: jayshiv
```

```
public void checkContestantDetails()
{
    int contestantNumber = 0;

    while (true)
    {
        try
        {
            System.out.print("Please enter the number of contestants (solo / duo) that are going to participate in this match (Min = 1, Max = 2): ");
            contestantNumber = input.nextInt();

            if (contestantNumber > 0 && contestantNumber < 3)
            {
                break;
            }
            else
            {
                throw new Exception();
            }
        }
        catch (Exception e)
        {
            System.err.println("-- You are only allowed to enter a number between 1 to 2! --");
            input.nextLine();
        }
    }
}
```

```
    for (int index = 0; index < contestantNumber; index++)
    {
        Contestant contestantInGame = new Contestant();
        System.out.println();
        System.out.printf("Enter the username for contestant number %d: ", index + 1);
        contestantInGame.setContestantUsername(input.next());

        contestants.add(contestantInGame);
    }
}
```

Bomb – Major Additional Functionality:

To spice up the game, we have constructed two different difficulties that contestants can choose from: easy or hard. For those that want an easy-going and relaxed game, easy is the way to go. On the other hand, the hard mode is for those that want a real competition. The key difference between easy and hard is that hard has an additional obstacle in the river known as a “Bomb”. The bomb obstacle was created similarly to how trap and current were created. In distinction from current and trap, the icon assigned to the bomb is “(B)”, and the bomb is able to generate a strength of either 10, 15, or 20. The bomb’s purpose in the game is to push back boats that land on it by 10, 15, or 20 steps. Contestants are allowed to choose between easy or hard, if they select hard, the river will be filled with currents, traps, and bombs. For a better understanding, please refer below:

```
Please enter your level of difficulty (easy / hard): hard
-----
Loading.....
Digging up a pathway....
Pouring water to create river....
Placing currents.....
Placing traps.....
Placing bombs.....
All done!

The game has started!
```

River Created:
 ~Go!~(T) (B) (C) (T) (T) (C) (T) (C) (T) (C) (B) (T) (C) (T) (C) (T) (C) (B) (C) (T) (C) (B) (T) (C) (T) (C) (T) (C) (B) (T) (C) ~End!~

```

/*
 * @author Group 14 Assignment (OOPF Boat Racing Game)
 */
public class Bomb extends RiverObject{ // Is-A-Relationship between Bomb and RiverObject (Inheritance).

    // Bomb will call the 2nd constructor in RiverObject and pass in the symbol specified.
    // Bomb is a subclass of RiverObject.
    // Additional Attribute

    // Constructor(s)
    public Bomb()
    {
        super(" (B) ");
    }
}

public void generateHardObjectStrength()
{
    int[] blist = {10, 15, 20};
    Random chooserrandoms = new Random();
    setObjectStrength(blist[chooserrandoms.nextInt(blist.length)]); // Chooses random strength (power) for the bomb object within the blist.
}

```

```

System.out.println();
System.out.print("Please enter your level of difficulty (easy / hard): ");
difficulty = input.next();

while ((difficulty.equals("easy") == false) && (difficulty.equals("hard") == false))
{
    System.err.println("-- Please enter 'easy' or 'hard' only! --");
    System.out.print("Please enter your level of difficulty (easy / hard): ");
    difficulty = input.next();
}

```

```

if (difficulty.equals("hard"))
{
    System.out.println("-----");
    System.out.println();
    System.out.println("Loading.....");
    System.out.println("Digging up a pathway....");
    System.out.println("Pouring water to create river....");
    System.out.println("Placing currents.....");
    System.out.println("Placing traps.....");
    System.out.println("Placing bombs.....");
    System.out.println("All done!");
    System.out.println();
    System.out.println("The game has started!");
    System.out.println();
    System.out.println("River Created:");
    System.out.println();
    riverStream.setCurrents();
    riverStream.setTraps();
    riverStream.setBombs(); // Additional Attribute
    System.out.println(riverStream.presentRiverStream(new ArrayList<Contestant>()));
    System.out.println();
}

```

```

if (difficulty.equals("easy"))
{
    System.out.println("-----");
    System.out.println();
    System.out.println("Loading.....");
    System.out.println("Digging up a pathway....");
    System.out.println("Pouring water to create river....");
    System.out.println("Placing currents.....");
    System.out.println("Placing traps.....");
    System.out.println("All done!");
    System.out.println();
    System.out.println("The game has started!");
    System.out.println();
    System.out.println("River Created:");
    System.out.println();
    riverStream.setCurrents();
    riverStream.setTraps();
    System.out.println(riverStream.presentRiverStream(new ArrayList<Contestant>()));
    System.out.println();
}

```


The Contestant Rolls The Dice:

```
ash, enter any letter or number (except enter) to throw the dice: t

ash threw the dice and got a 2.
ash moved to 2.
```

The Contesntant Lands on a Trap:

```
ash threw the dice and got a 2.
ash moved to 15.
Oh no! ash, you landed on a trap and moved back by 6 :(!
ash moved to 9.
```

The Contestant Lands on a Current:

```
ash threw the dice and got a 4.
ash moved to 21.
Yay! ash, you landed on a current and moved front by 7 :)!
ash moved to 28.
```

Boat Moving Across The River:

```
~(Go!)-(T)(C)__(T)(C)__(T)(C)__(T)(C)__(C)__[ _ ash's Boat]____(C)__(T)__(T)__(T)(C)__(C)__(T)__(T)(C)__(T)(C)__(C)__(T)(C)__(T)(C)__(T)__(End!)~
```

Contestant Finishes The Game:

```
~(Go!)-(T)(C)__(T)(C)__(T)(C)__(T)(C)__(C)____(C)__(T)__(T)__(T)(C)__(C)__(T)__(T)(C)__(T)(C)__(C)__(T)(C)__(T)(C)__[ _ ash's Boat]~(End!)~

You reached the finish line!
The game has ended! Yay!
ash has won the game with exactly 31 turns! Congratulations, you played very well!
```

Feedback If Yes:

```
-----
|TOP 5 SCOREBOARD|
(Placing)----- (Username)----- (Score)
  1.           ashh           31
  2.           ash           31
-----

Please enter 'yes' if you would like to give us feedback on the game, and 'no' if no: yes
Enter feedback here (500 characters): This game is very fun! I enjoy playing it.
Great feedback! Thank you for playing Group 14's Boat Race Game! Please come again soon!
```

Feedback If No:

Please enter 'yes' if you would like to give us feedback on the game, and 'no' if no: no

Thank you for playing Group 14's Boat Race Game! Please come again soon!

SCENARIO 2: If there are 2 contestants and the difficulty is set to hard:

```
Please enter the number of contestants (solo / duo) that are going to participate in this match (Min = 1, Max = 2): 2
Enter the username for contestant number 1: elena
Enter the username for contestant number 2: azri
Please enter your level of difficulty (easy / hard): hard
```

```
Loading.....
Digging up a pathway...
Pouring water to create river....
Placing currents.....
Placing traps.....
Placing bombs.....
All done!

The game has started!

River Created:

~(Go!)~_(C)__(T)(C)(B)__(T)__(T)(C)__(C)__(T)__(T)(C)__(C)__(T)__(C)(B)__(T)(C)__(T)__(T)(C)(B)__(T)(C)__(T)(C)(B)__(T)(C)__(T)__(C)__(T)(C)__(T)(C)(B)__(End!)~_
```

Contestants Rolling The Dice:

```
elena, enter any letter or number (except enter) to throw the dice: t
elena threw the dice and got a 4.
elena moved to 4.

azri, enter any letter or number (except enter) to throw the dice: t
azri threw the dice and got a 3.
azri moved to 3.
```

The Contestant Lands on a Bomb:

```
elena threw the dice and got a 4.
elena moved to 27.
Oh no! elena, you landed on a bomb and moved back by 10 :(
elena moved to 17.
```

- [Go!] ~ _____ (T) (C) ____ (T) (C) (B) ____ (T) [_ elena's Boat]_ (T) (C) (B) ____ (T) (C) (B) [_ azri's Boat]_ (T) (C) ____ (T) ____ (T) (C) ____ (T) (C) ____ (C) ____ (T) (C) ____ (T) (C) ____ (T) (C) (B) ____ (T) (C) ____ (T) (C) ____ (T) (C) ____ (C)

```
You reached the finsh line!  
  
The game has ended! Yay!  
  
azri has won the game with exactly 43 turns! Congratulations, you played very well!
```

```
Please enter the number of contestants (solo / duo) that are going to participate in this match (Min = 1, Max = 2): 0  
-- You are only allowed to enter a number between 1 to 2! --  
Please enter the number of contestants (solo / duo) that are going to participate in this match (Min = 1, Max = 2): 1
```

```
Please enter your level of difficulty (easy / hard): meow  
~~ Please enter 'easy' or 'hard' only! ~~  
Please enter your level of difficulty (easy / hard): easy
```

```
Please enter 'yes' if you would like to give us feedback on the game, and 'no' if no: hello  
~~ Please enter 'yes' or 'no' only! ~~  
Please enter 'yes' if you would like to give us feedback on the game, and 'no' if no: yes
```

```
Enter feedback here (500 characters): tttttttttttttttttttttttttttttttttttttttaoakotawatkoakewtttttttttttttttttttttttttttttttttttttttwaointowinfoienaiowenf  
~~ Please keep feedback under 500 characters! ~~  
Enter feedback here (500 characters):
```


Scenario Summary

As soon as the game starts, the menu will be displayed which includes the banner, welcome message, instructions, and scoreboard. In both scenarios, the contestant(s) will enter the number of people playing, and the username(s). They will be given the choice of choosing which mode they want to play in, which is either “easy”, or “hard”. If easy, the river will be filled with only currents and traps. If hard, the river will be filled with currents, traps, and bombs. The contestant(s) roll their dice each round and will be able to see their boat move on the river display. If they land on a trap, bomb, or current, the appropriate message will be displayed. The contestant(s) will keep going until they reach the end and finish the game. Once they finish, the game will congratulate the winner and the scoreboard will be displayed again. The contestant(s) will be able to see if they made it into the scoreboard. After that, the contestant(s) will have an option to type in and send feedback about the game. If they enter “yes”, they will be able to send feedback and after that the game will thank them for their feedback and for playing the game. If they enter “no”, the game will thank the contestant(s) for playing the game.