

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**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 2023; SEMESTER 2,3,4**

**PRG1203: OBJECT ORIENTED PROGRAMMING FUNDAMENTALS**

**DEADLINE: 31 JULY 2023 11:59PM (Monday)**

**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 Lee Xing Le, Liew Jieh Cheng, Tee Yu Chen, Yen Ming Wey (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.”



..................., ..................., ...................., …..................(Student’s signature / Date)



**Group Number: 26**

**Team Members:**

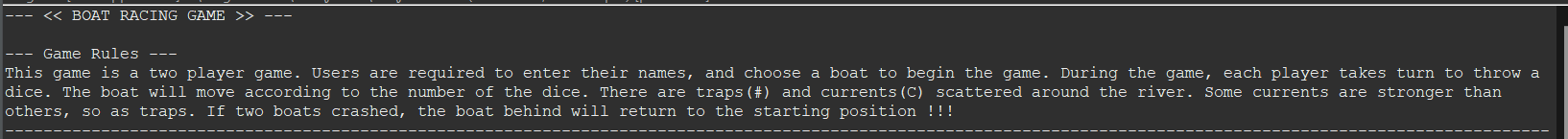
|  |  |  |  |
| --- | --- | --- | --- |
| No | Name | Student ID | Contribution % |
| 1 | Lee Xing Le | 21030234 | 23.3% |
| 2 | Liew Jieh Cheng | 21045471 | 23.3% |
| 3 | Tee Yu Chen | 21030317 | 30% |
| 4 | Yen Ming Wey | 21030457 | 23.3% |

**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** | |  |  |

**Test Cases**

**1.0 Start Screen**



A black screen with white text

Description automatically generated

A screenshot of a computer program

Description automatically generated

When the game is started, players will be greeted with the game title and a clear set of rules, ensuring they understand how to engage with the game effectively. Beneath the rules, a scoreboard showcasing the top 5 historical scores, along with corresponding player names, will be displayed. To generate the scoreboard, data from the 'TopScore.txt' file is read and stored in an ArrayList. Then, a for loop is used to print out the players and scores in a proper format. Subsequently, the players will be prompted to enter their respective names and make boat selections from a range of 5 available options. These boat choices are readily accessible through an array created in the Boat class. The implementation of these features aims to provide a user-friendly and intuitive experience for players, setting the stage for an enjoyable gaming session.

**1.1 Validation of Player Information**

A black background with white text

Description automatically generated

A screen shot of a computer

Description automatically generated

To maintain data integrity and prevent invalid entries, players are required to enter their names as a single word without any spaces. This is achieved through a continuous prompting process using a while loop**, while (name.contains(“ ”))**. The loop checks if the name contains any spaces and keeps iterating until a valid one-word name is provided. In the test case, “Tom Cruise” is entered. Thus, a message is shown to get a new value. When “TomCruise” is entered on the second attempt, the game is proceeded. This approach ensures that the players adhere to the required input format before they can proceed with the game.

A black screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

To ensure that players select a boat from the available options in the "boatList" array, the program validates the input to be within the range of 1 to 5. This validation approach get a string input from the user and checks whether the input matches the number 1 to 5 by using **!selectedBoatIndexStr.matches("[1-5]").** When we enter a character or a number that isn’t within the boat selection in this test case, the program responds by prompting them to enter a number between 1 and 5. This validation effectively enforces the constraint that players must choose a boat from the provided list.

**2.0 Game Screen**

**2.1 Boat Moving Screen**

A screen shot of a computer

Description automatically generated

Upon entering the game screen, players will encounter a challenging river filled with treacherous currents (C) and hidden traps (#). The start and end points of the game are clearly marked for reference. Two boats are placed beside the starting points. To set their journey in motion, players are prompted to press any key, which will trigger the rolling of a dice. Once a key is entered, the boats will begin their voyage, propelled by the number randomly generated on the dice. The outcome of the dice roll will determine the distance the boats can traverse on the river.

A group of lines on a black background

Description automatically generated

A black background with many small colored lines

Description automatically generated

In this test case, the dice number is 3. As a result, the boat will move to the third position in the river. In the final step, a clear message will be shown, indicating that the boat is gradually slowing down until it eventually comes to a complete stop.

**2.2 Effect of Trap**

A black background with white lines

Description automatically generated

A screenshot of a computer

Description automatically generated

A black background with many small colored lines

Description automatically generated

In this scenario, we aim to demonstrate that the boat will travel in the reverse direction upon encountering a trap. The dice roll result is 1. Looking at the diagram, we can observe the presence of a trap adjacent to the boat's position. Consequently, the boat starts moving and eventually halts at the location of the trap. An alert message promptly notifies the player about this unfortunate encounter, along with the specific number of steps the boat will move backwards. Subsequently, the boat moves backwards and comes to a standstill at the specific position.

**2.3 Effect of Current**

**A black background with white lines and symbols

Description automatically generated**

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

In this situation, we want to illustrate that the boat will move forward when it encounters a current. The result of rolling the dice is 1. By examining the diagram, we can see that there is a current nearby, right next to the boat's current position. As a result, the boat moves 1 step and eventually stops at a location where the current is positioned. A message is shown to notify the player that the boat will be moving forward for 5 steps. The boat then moved forward for 5 steps and stopped.

**3.0 Ending Screen**

A screenshot of a computer screen

Description automatically generated

A black screen with white text

Description automatically generated

After our previous test case, the gameplay continued, and the boat with the label "|xxx|" has won the game. Subsequently, the program informed the player of their victory and displayed the river scenario. Notably, the player named "asg" took only 22 steps, which stands as the highest score among historical records on the scoreboard. Consequently, the program incorporated the player named "asg" into the scoreboard, updating it to reflect the new addition. The revised scoreboard was then presented to the player.

**Additional Features**

**1. Boat Crashing Effect**

**A black background with blue lines

Description automatically generated**

A screenshot of a computer

Description automatically generated

A screen shot of a computer screen

Description automatically generated

A group of small colored lines

Description automatically generated

In this test scenario, we aim to demonstrate that a boat will return to its starting point upon colliding with another boat. Initially, boat |AAA| is positioned two steps behind boat |DDD|. The player rolls the dice and obtains a result of 2. Consequently, boat |AAA| advances two steps forward, eventually halting at the same position as boat |DDD|. As a result, the program promptly notifies the player that their boat has encountered a collision with another boat, then directs the boat back to its initial starting point. The rationale behind this design is to prevent two boats from staying at the identical position as this may be hard for the player to track the boat. It also significantly improves the player’s experience as the excitement of the game has increased.

**2. Dynamic Boat Moving Animnation**

A screen shot of a computer

Description automatically generated

A group of lines on a black background

Description automatically generated

A black background with many small lines

Description automatically generated

A black background with many small colored lines

Description automatically generated

In this test case, we are utilizing the example previously employed in our initial game screen section to showcase the dynamic boat moving animation feature in the game. Our aim is to improve gameplay smoothness and offer users a more immersive gaming experience. In the original design, only the initial and final positions of the boat were displayed for each turn. However, we have now decided to enhance the game by introducing a step-by-step animation that reveals the boat's movement along the river.

From the diagram, we can see that the number of the dice rolled is 3. Consequently, the river will be displayed three times, highlighting each step of the boat's progression. To create a realistic effect, a brief pause of 0.5 seconds will be inserted between each step of the animation. This pause is realized through the utilization of the **Thread.sleep(500)** function from the **java.lang.Thread** class, allowing the program to temporarily halt for 500 milliseconds before proceeding to the subsequent instruction. By incorporating this dynamic animation, we seek to elevate the overall gameplay experience and engage users more profoundly.

**3. Boat Selections**

**A black screen with white text

Description automatically generated**

The last enhancement we have included is providing five distinct boat options for the players to select from. This valuable addition significantly enhances the interactive element of the game, reducing the risk of monotony that players may experience when playing repetitively. With these diverse choices, players are now presented with an enriched and engaging experience, contributing to a more immersive and enjoyable gameplay.