

SCHOOL OF ENGINEERING AND TECHNOLOGY

FINAL ASSESSMENT FOR THE BSC (HONS) INFORMATION TECHNOLOGY; BSC (HONS) COMPUTER SCIENCE; BACHELOR of SOFTWARE ENGINEERING (HONS) YEAR 2

ACADEMIC SESSION 2023; SEMESTER 3

PRG2104: OBJECT ORIENTED PROGRAMMING

Project

DEADLINE: Week 14

INSTRUCTIONS TO CANDIDATES


- This assignment will contribute 50% to your final grade.
- This is an individual 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.

- Coursework submitted after the deadline will be awarded 0 marks
-

Lecturer's Remark (Use additional sheet if required)

I.....BONG HONG JUN..... (Name)22018519.....std. ID received the assignment and read the comments..........20/08/2023..... (Signature/date)

Academic Honesty Acknowledgement

"IBONG HONG JUN.....(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 to page 16, 5.5, Appendix 2, page 44 of the student handbook diploma and undergraduate programme*) for any kind of copying or collaboration on any assignment."


 20/08/2023..... (Student's signature / Date)

Table Of Content

1.0 INTRODUCTION.....	3
2.0 UML CLASS DIAGRAM.....	4
3.0 Program Features.....	5
3.1 Main Menu.....	5
3.2 About Information.....	5
3.3 Game Initialization.....	6
3.4 Tile Click Interaction.....	6
3.5 Shark Counter.....	7
3.6 Time Counter.....	7
3.7 Winning Condition.....	7
3.8 Losing Condition.....	8
3.9 Revealing All Tiles on Loss.....	8
3.10 Restarting the Game.....	9
3.11 Exiting the Application.....	9
4.0 Object-Oriented Concepts Applied	10
5.0 Personal Reflection.....	11
6.0 Conclusion.....	12
References.....	12

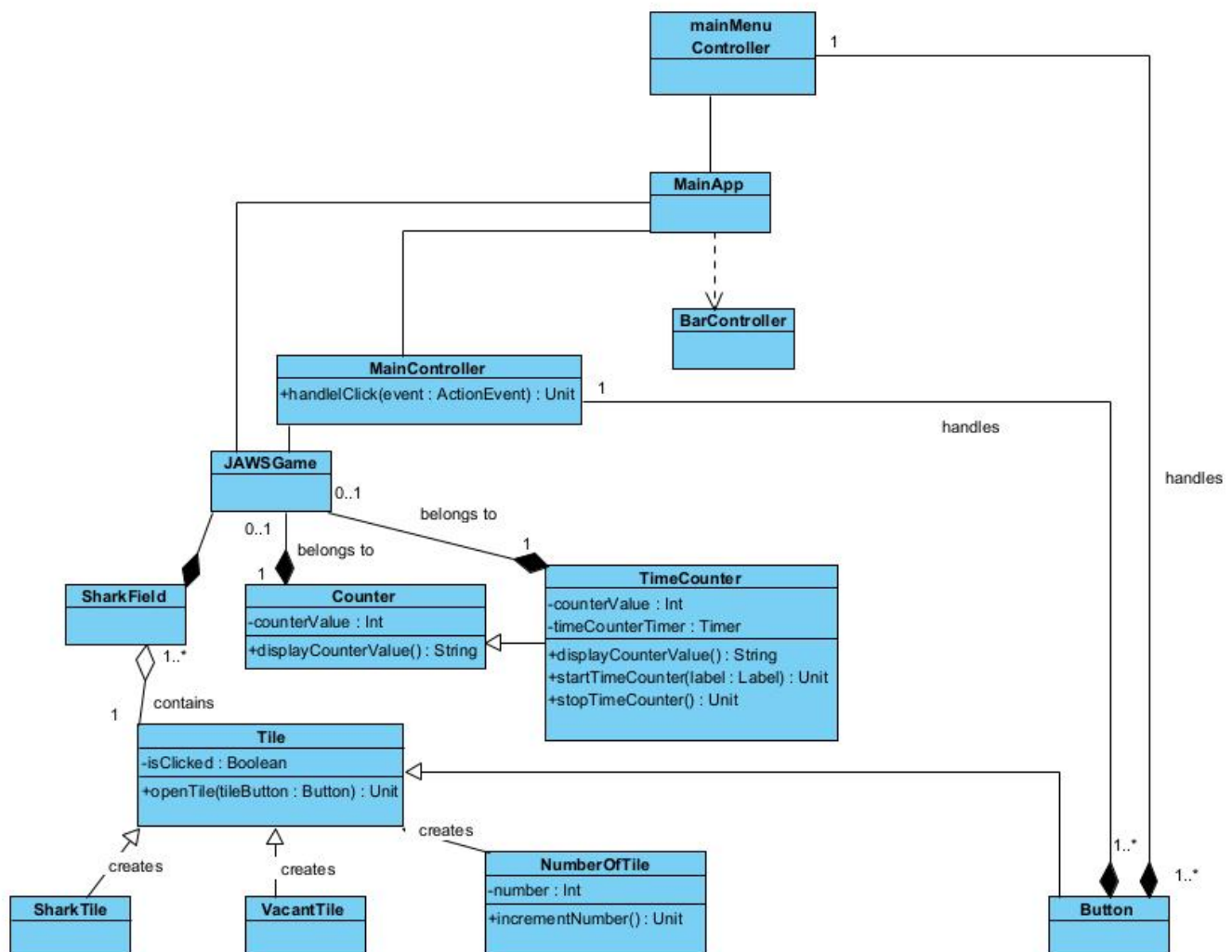
Project Report: JAWS GAME - Minesweeper Game

Gamr Presentation YouTube Video Link: <https://youtu.be/txCmC3tp140>

1.0 Introduction

This project involves the design and implementation of an underwater-themed Minesweeper game called "JAWS GAME." The game utilizes ScalaFX, a GUI library for Scala applications, to create an interactive graphical user interface. The primary goal of this project is to demonstrate proficiency in object-oriented programming concepts, including inheritance, polymorphism, abstract classes, and generic programming, while adhering to good software design principles.

2.0 UML Class Diagram



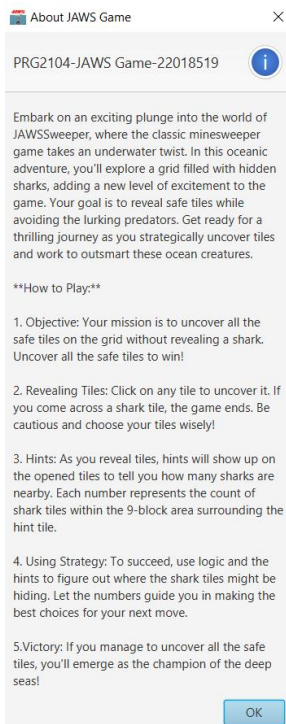
3.0 Program Features



3.1 Main Menu

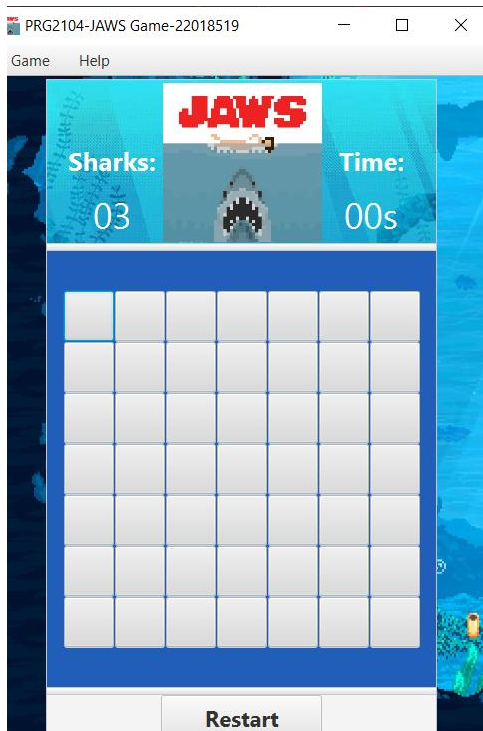
Users are presented with a main menu upon launching the application. The main menu offers options to start a new game, learn about the game, and exit the application.

3.2 About Information



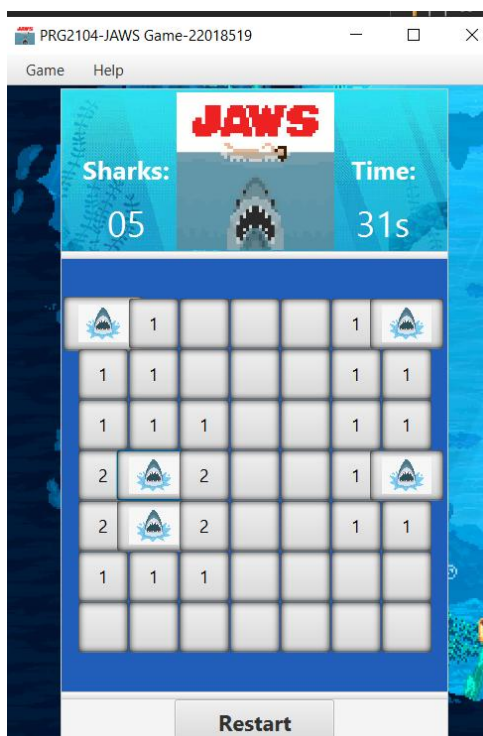
Users can access information about the game's rules and objectives. An information dialog provides details on how to play the game and what the user's goal is.

3.3 Game Initialization



When a new game is started, the game field is initialized with a random configuration of shark and vacant tiles. The mine count and time counter are reset to their initial values.

3.4 Tile Click Interaction



Users can click on tiles to reveal their content. Shark tiles display a shark image, vacant tiles display the number of surrounding shark tiles, and number tiles show their own number.

3.5 Shark Counter



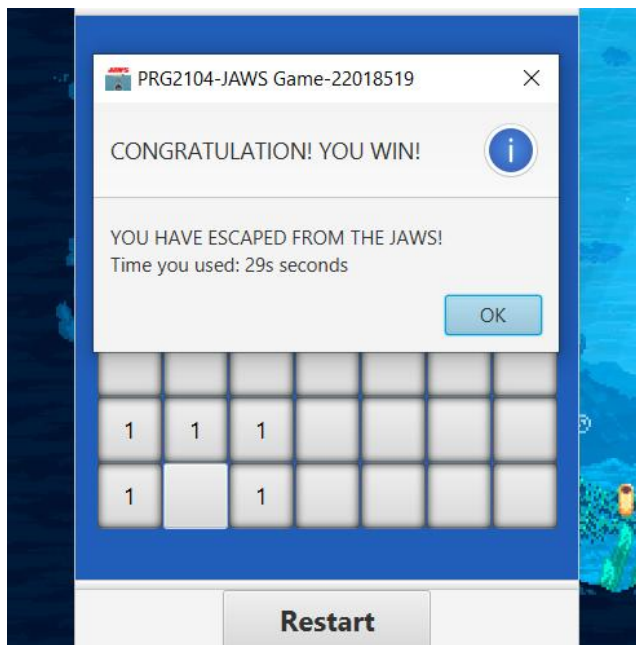
Users are provided with a mine counter that displays the number of remaining shark tiles. The counter decreases as vacant tiles are revealed.

3.6 Time Counter



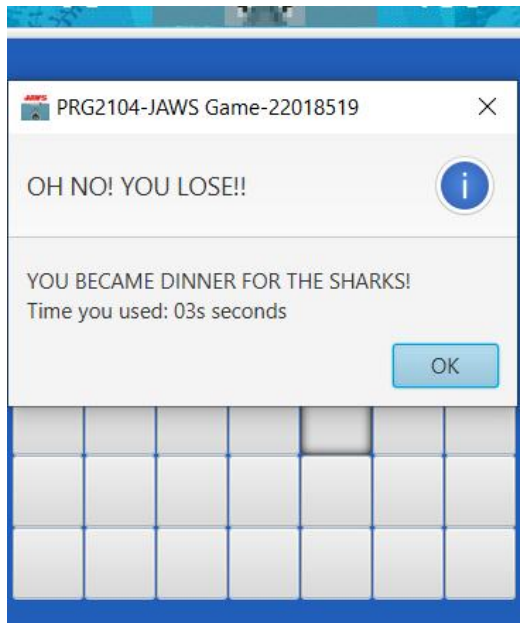
A time counter keeps track of the duration of the ongoing game in seconds. The counter starts when the game begins and stops when the game ends.

3.7 Winning Condition



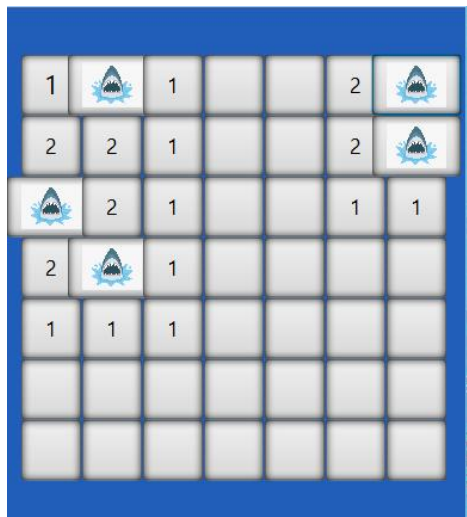
The game ends in a win if all vacant tiles are successfully revealed without revealing any shark tile. An alert message is displayed congratulating the user on their victory.

3.8 Losing Condition



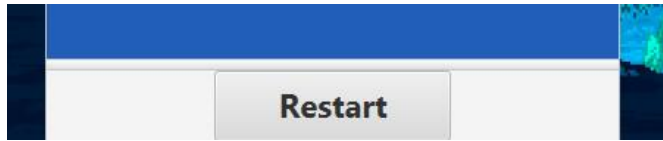
The game ends in a loss if a shark tile is revealed by the user. An alert message notifies the user of their unfortunate fate.

3.9 Revealing All Tiles on Loss



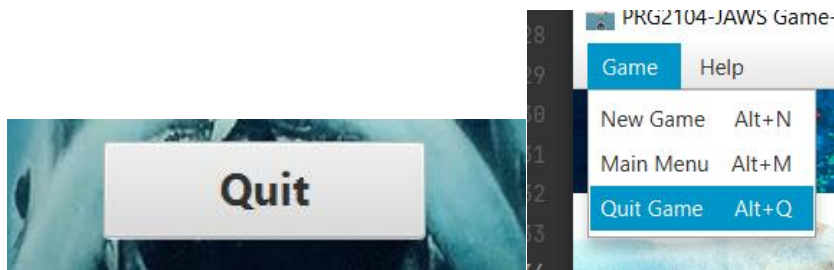
In case of a loss, all tiles are automatically revealed to display their content. Shark tiles are shown with a shark image.

3.10 Restarting the Game



After a game ends (win or loss), users can choose to start a new game by clicking the restart button at the bottom of the game board. The game field is reinitialized, and the counters are reset.

3.11 Exiting the Application



Users can choose to exit the application from the main menu. Graphical User Interface (GUI): The game features a user-friendly GUI built using the ScalaFX library. Buttons, labels, and images are utilized to create an interactive and visually appealing interface.

4.0 Object-Oriented Concepts Applied

4.1 Inheritance and Polymorphism

```

abstract class Tile(val tileType: String) {
    var isClicked = false
    var symbol: String

    def openTile(tileButton: Button): Unit = {
        // Display the button text to show the symbol
        tileButton.setTextFill(Color.Black)
        // Change the look of the button to indicate clicked
        tileButton.setEffect(new InnerShadow())
        isClicked = true
    }
}

```

```

class SharkTile(var symbol: String = "") extends Tile( tileType = "mine") {
}

class VacantTile(var symbol: String = "") extends Tile( tileType = "empty") {
}

class NumberOfTile(var symbol: String = "1") extends Tile( tileType = "number") {
    var numberOfMines = 1
    def incrementNumber(): Unit = {
        numberOfMines += 1
        symbol = numberOfMines.toString
    }
}

```

The project incorporates inheritance and polymorphism through class hierarchies and overriding methods. The Tile class serves as a base class with subclasses SharkTile, VacantTile, and NumberOfTile. Polymorphism is demonstrated by using a common method (openTile) across different tile types.

4.2 Abstract Class

The abstract class Counter is utilized to represent the counters for sharks and time. This abstraction simplifies counter management and allows for easy extension to other counter types if needed.

4.3 Generic Programming

```
// Abstract class to represent different types of counters
abstract class Counter(val counterType: String) {
    var counterValue: Int

    // Increment the counter value
    def incrementCounter(): Unit = {
        counterValue += 1
    }

    // Decrement the counter value
    def decrementCounter(): Unit = {
        counterValue -= 1
    }

    // Get a formatted string to display the counter value
    def displayCounterValue(): String
}
```

Generic programming is employed in the Counter class to create a generic counter structure that can be used for various counter types, such as shark and time counters.

5.0 Personal Reflection

In developing this project, I gained a deeper understanding of how to apply object-oriented programming concepts to create well-structured and modular code. The design process highlighted the importance of class hierarchy and code reuse, leading to more efficient and maintainable code. I faced challenges in managing the graphical interface and event-driven programming, but through diligent research and practice, I was able to achieve elegant and functional GUI components.

5.1 Challenges and Solutions

5.1.1 Graphical Interface

Designing and implementing the GUI was challenging due to the need for proper layout and responsiveness. I overcame this by studying the ScalaFX documentation, experimenting with layout classes, and refining the interface iteratively.

5.1.2 Tile Click Handling

Implementing tile click functionality while maintaining the integrity of the game's logic was complex. To address this, I used a combination of controller methods, model logic, and event handlers to ensure proper tile interaction.

5.2 Strengths and Weaknesses

5.2.1 Strengths

- Effective use of object-oriented principles, resulting in a well-structured and extensible codebase.
 - Mastery in event-driven programming and utilization of ScalaFX's GUI components.
- Demonstrated understanding of inheritance, polymorphism, abstract classes, and generic programming.

5.2.2 Weaknesses

- In some cases, the code could benefit from further optimization and refactoring.
- Documentation could be more comprehensive and should include code comments for better understanding.

6.0 Conclusion

The completion of the JAWS game project has been a valuable learning experience. By integrating object-oriented programming concepts and applying them in a practical context, I have deepened my understanding of software design and coding practices. The challenges encountered during the development process served as opportunities for growth and learning, leading to a more robust and functional application.

References

ScalaFX simpler way to use javafx from scala. ScalaFX Simpler way to use JavaFX from Scala. (n.d.). <https://www.scalafx.org/>

Steven Spielberg, John Williams & John Williams. (1975) *JAWS*. USA.

Becerra, David J. 2015. Algorithmic Approaches to Playing Minesweeper. Bachelor's thesis, Harvard College. Available at: <http://nrs.harvard.edu/urn-3:HUL.InstRepos:14398552>

Google [Google](#)