

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

**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

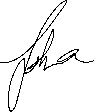
# 

**Lecturer’s Remark** (Use additional sheet if required)

I.............................. (Name) ...................std. ID received the assignment and read the comments....................................... (Signature/date)

**Academic Honesty Acknowledgement**

“I, Joshua Raweng anak Dominic(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 Aug 2023 (Student’s signature / Date)

Table of Contents

[Introduction 3](#_Toc143445549)

[System Functionalities 3](#_Toc143445550)

[Class diagram 4](#_Toc143445551)

[System Feature Description and Show Case 5](#_Toc143445552)

[Personal Reflection 5](#_Toc143445553)

[References 6](#_Toc143445554)

# Introduction

This project's proposed standalone GUI system is a personal game called "Fate/EXTRA Rulers of Ishgria." This report aims to outline the system's development and design, as well as a UML diagram that illustrates the conceptualization behind it. The Playstation Portable (PSP) game "Fate/EXTRA" served as a major inspiration for this game. Only the combat elements from "Fate/EXTRA" are expanded upon in "Fate/EXTRA Rulers of Ishgria," which switches the controls from a PSP to a computer and uses mouse clicks instead of the PSP's buttons to operate the game. The opponents in this game are taken from another game called "Brave Frontier", thus creating a new game that melds both inspiring games into one.

# System Functionalities

|  |  |
| --- | --- |
| Game Tutorial | A tutorial is provided to players to explain how the combat mechanic works. |
| Choosing an attack type | Players can choose between one of three attack types in the game in an attempt to counter the enemies’ attacks |
| Undo attack selection | Should the player make a mistake in selecting an attack, the player can undo the last attack selection made |
| Confirm attack sequence | Players are prompted after choosing the last attack in a sequence to confirm their selection of attacks |
| Attack damage | Each turn, depending on the attacks used by the player and the enemy, health is deducted and this is displayed through the progress bar |
| Reveal enemy attack pattern | Upon correct selection of attack to counter the enemy’s attack, the attack the enemy will use for that turn will be shown to the user for the rest of the fight |
| Zero health outcomes | If the player’s health reaches zero, the player is defeated and the screen shows the defeat screen. If the enemy’s health reaches zero, the player continues to the next stage with their health restored |

# Class diagram

A screenshot of a computer

Description automatically generated

# System Feature Description and Show Case

Video link: <https://youtu.be/oRKXjsvFtHw>

# Personal Reflection

This project has applied the principles of object-oriented programming using a variety of methods. The main purpose of the ScalaFX API was to implement concepts in object-oriented programming. These concepts cover the primary stage, action events, progress bars, double properties, and more, and how to construct JFXApp utilizing polymorphism and inheritance. A navigation bar, an anchor pane, buttons, image views, and other GUI elements were also used by SceneBuilder. Additionally, the Scala collection library was applied in this project, specifically the mutable Stack collection. This collection enabled implementing the means of undoing attack selections.

Many problems were encountered during this assignment. The biggest problem encountered was settling on an idea to do for this project. A significant portion of the timeline was spent figuring out what to do for the project as the project was very open-ended. There were many ideas that came to mind with varying degrees of complexity, and it was difficult to gauge whether an idea was feasible to execute given the timeline. The chosen idea for the project had been revised about two to three times before finally settling on one idea and this decision was only done rather close to the deadline. Upon reflection, the better approach may have been to start simple and expand from a baseline idea, rather than to start on a big idea only to find faults along the way and are far too costly to recover from.

One strength this program has is that it implements a very simple rock-paper-scissors mechanic and has the potential to expand further and add more dimensions and layers to the gameplay. For example, what was implemented in the game from which this GUI system was inspired from, “Fate/EXTRA”, is that the player and enemies also have a set of enhancing or debilitating skills which are layered on top of the main rock-paper-scissors combat style. On the other hand, one weakness that is identified out of this program is that it may not be very well-organised in terms function placements. It is identified that since all four stages of the game generally share the same functions with differences mainly lying in the attack patterns and scenes to proceed to upon player victory that stage, there may be a possibility for organising the code better so that these similar functionalities can be accessible through a central means such as in a class.

Through the use of the Stack collection API, the OOP concept of parametric polymorphism has been integrated. The Stack collection can contain only one type of class but this class can be any class. In this project, Stack is implemented to contain the String variable type for the player to decide their attack sequence.

# References

1. Enemy Images, obtained from: <https://bravefrontierglobal.fandom.com/wiki/Rulers_of_Ishgria#Characters>

2. Combat Backgrounds, obtained from: <https://cardfight.fandom.com/wiki/Planet_Cray#Locations>

3. Progress Bar, learnt through: <https://www.youtube.com/watch?v=nEKQjAP0lrQ>