A close up of a logo

Description generated with very high confidence

**Diamond Hunter Map Viewer**

**G52SWM Coursework 2**

Team:

* Md Alvi
* Bhavish Doobaree
* Omar Mahmoud

Module Conveyor: Hani Behrang

# Table of Contents

[Table of Contents 2](#_Toc501894392)

[Introduction 3](#_Toc501894393)

[Implementation Method 4](#_Toc501894394)

[The Interface 4](#_Toc501894395)

[The Interface Controller 4](#_Toc501894396)

[The Launcher 8](#_Toc501894397)

[Exception Handling and Alerts 8](#_Toc501894398)

[Main Menu 10](#_Toc501894399)

[Design of the main menu 10](#_Toc501894400)

[Main Menu Controller 10](#_Toc501894401)

[Main Menu Display 11](#_Toc501894402)

[Problems Encountered 12](#_Toc501894403)

# Introduction

The aim of this coursework is to create a program that will view and edit the map of the Diamond Hunter game. The program should be able to allow the user to make changes to the map by moving the axe and the boat. I should be interactive and additional features are also welcome.

To be able to complete this project, we had to know our way around the Diamond Hunter code and therefore knowledge from the previous coursework was very helpful. The program will be using features like JavaFX and SceneBuilder to make the interface.

This coursework therefore involved much experimenting with JavaFX to see how it works, how to use it and how it behaves in certain conditions, which proved to be challenging but still fun.

In addition, we will also be using GitHub and Git to synchronise our data across all of our devices that we use for coding.

# Implementation Method

The program consisted of several elements:

* An interface
* A controller for the interface
* A launcher
* Exception handling with alert messages
* Main Menu

## The Interface

To make the interface, we had to use JavaFX together with SceneBuilder, which is an interactive program to make the interface. The file was in the form of an FXML Document. SceneBuilder allows the creation of several elements on the interface. For our purpose, we used buttons and text boxes as well as a canvas (for the map display) for our interface. Here is a screenshot of the planned interface.

A screenshot of a computer

Description generated with very high confidence

## The Interface Controller

The controller is the most essential part of the project. It will tell the interface how to react in certain conditions such as a mouse click. It will also be able to use the Canvas to display the map. It will also be responsible to display error messages and handle exceptions. It is basically the brain of the interface. Here are some screenshots of the interface controller.

A screenshot of a cell phone

Description generated with very high confidence

This part of the code directs the controller where to get initial data from. It also displays the coordinates in the text box. Since the container is 16 times bigger, it compensates with division by 16 for user display.

A screenshot of a cell phone

Description generated with very high confidence

Checks if items have actually been placed to check if position valid later. Compensates for the 16 times again here.

A screen shot of a computer

Description generated with very high confidence

Gets data from the module shown before. Acts on the data by displaying the correct message. Prevents user from causing a bug in the game.

A screenshot of a cell phone

Description generated with very high confidence

This module is responsible for reading the map file, line by line, from testmap.map

A close up of a computer

Description generated with high confidence

This module is responsible for initializing the testtileset.gif file from resources

A screen shot of a smart phone

Description generated with high confidence

This module is responsible for generating the graphics and also making sure that it does not go out of bounds for the map since it is 40x40.

## The Launcher

The launcher is responsible for accessing the FXML document and then executing it as an application. The module has a very simple and straightforward implementation.

A screenshot of a cell phone

Description generated with very high confidence

It sets the title of the window and imports the FXML document as shown.

## Exception Handling and Alerts

These modules were created only for easy debugging. Instead of putting everything in the controller, the separation made it easier to use. The exception handler is invoked whenever try…catch is used so that it is set to displays the proper message. The alert module is only to display the window. They work together to display the messages.

A close up of a screen

Description generated with high confidence

Passes the message using the excptxt variable.

A screenshot of a cell phone

Description generated with very high confidence

Parameters for the display window. Data to be displayed is obtained form the exception handler through the excptxt variable.

## Main Menu

### Design of the main menu

A screenshot of a cell phone

Description generated with very high confidence

This is the initial design of the main menu. It will be located in the Game package.

### Main Menu Controller

A screenshot of a cell phone

Description generated with very high confidence

This will be the main menu controller. It opens and access the FXML document to display and control the main menu.

### Main Menu Display

A screenshot of a cell phone

Description generated with very high confidence

This module will be the main module of the whole game. From this module, the user will be able to access both the game and the tile map editor.

# Problems Encountered

After implementing the project, we have to test if it works and that is where all the problems crop by.

Here are some of them:

* FXML file not recognised due to incorrect hierarchy.
* Cannot access FXML file. Main module cannot determine location.
* Exception Handler unable to communicate with the GUI controller.
* Game states not appropriate causing the program to crash.
* Exception occurred when running the program.