HOCHSCHULE DER MEDIEN

Software Entwicklung 2

Sommersemester 2022

Projektaufgabe: Minigames

Gruppenteilnehmer:

Knülle, Bianca	bk114	Medieninformatik
Fellmeth, Julia	jf126	Medieninformatik
Sfantou, Despoina	ds215	Medieninformatik
Casimiro des Oliveira, Kethella Karoline	kc029	Medieninformatik

Pfad zum Repository: https://gitlab.mi.hdm-stuttgart.de/ds215/MiniGames
--Deckblatt—

Brief description

This project is a summer semester 2022 project which is a collection of mini games. It is part of the Software Development 2 lecture in the second semester at the Hochschule der Medien.

The game consists of various mini-games. One game is *Hangman* in which you try to guess a mystery word. The number of letters in the word is represented by underscores. The player has to type in a letter and the game checkes whether this letter is contained in the word. If so, the letter will appear instead of the underscore, if not, the count will count up to the maximum. *Hangman Multiplayer* is all about guessing the word you are looking for from the other player. Now each player tries to guess his word before the counter reaches the maximum level or the other guesses the word.

The game Beesweeper is about uncovering all the squares except for those under which a bee is hiding. A flag should be placed on squares with a bee. The revealed numbers indicate how many bombs are in the adjacent squares (it can be up, down, right, left or diagonally). If you open all fields that do not contain any bees and also place all flags correctly, you win the game.

Starting class

The Main method is in the *Main* class. This is located in the package minigames.mainpackage.

Special feature

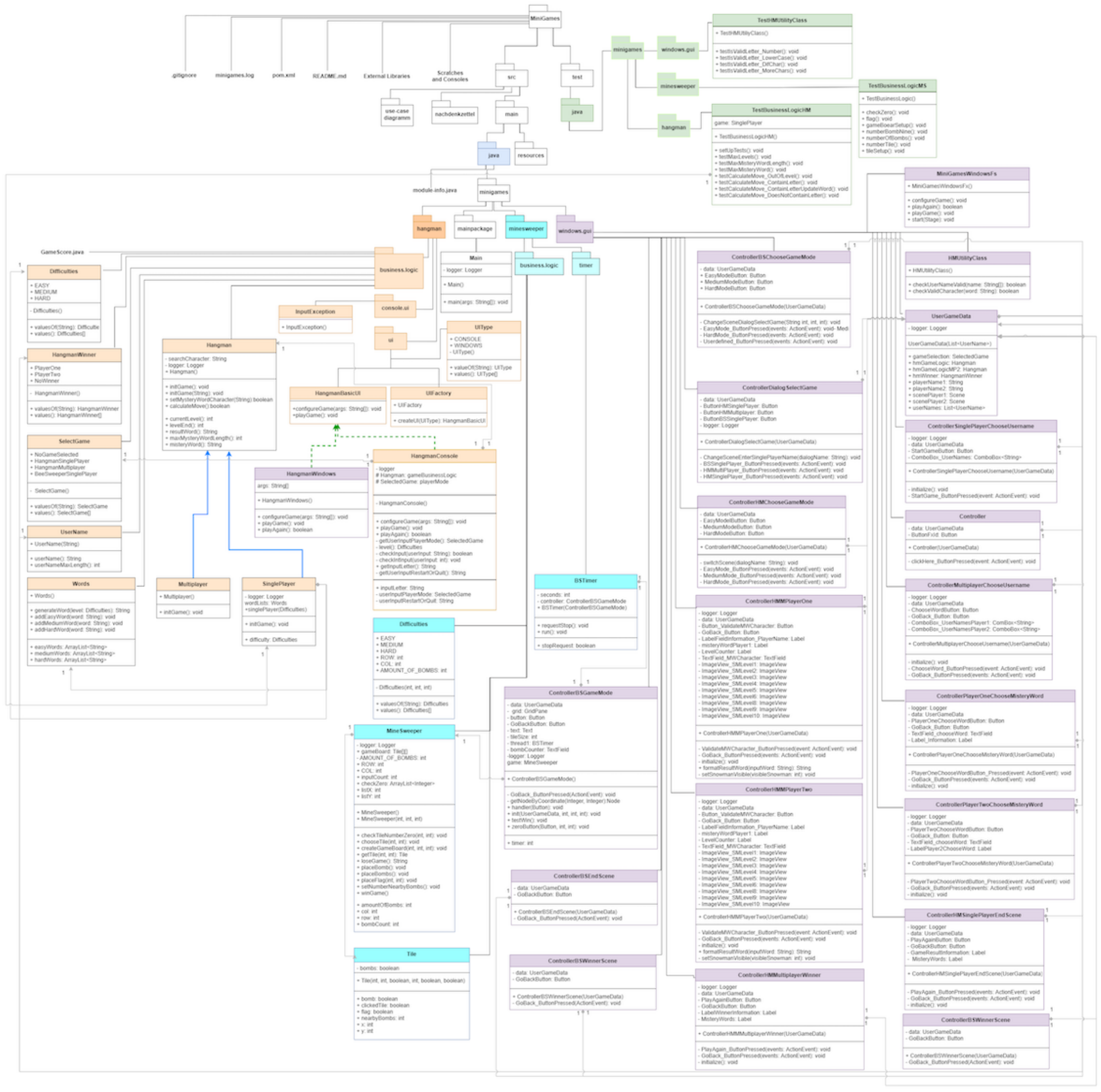
Dependencies for Logging:
The first setup for the logging are the dependecies in the pom.xml file.

```
 <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-api</artifactId>
        <version>2.17.2</version>
</dependency>
<dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-core</artifactId>
        <version>2.17.2</version>
</dependency>
```

UML

UML Class diagram

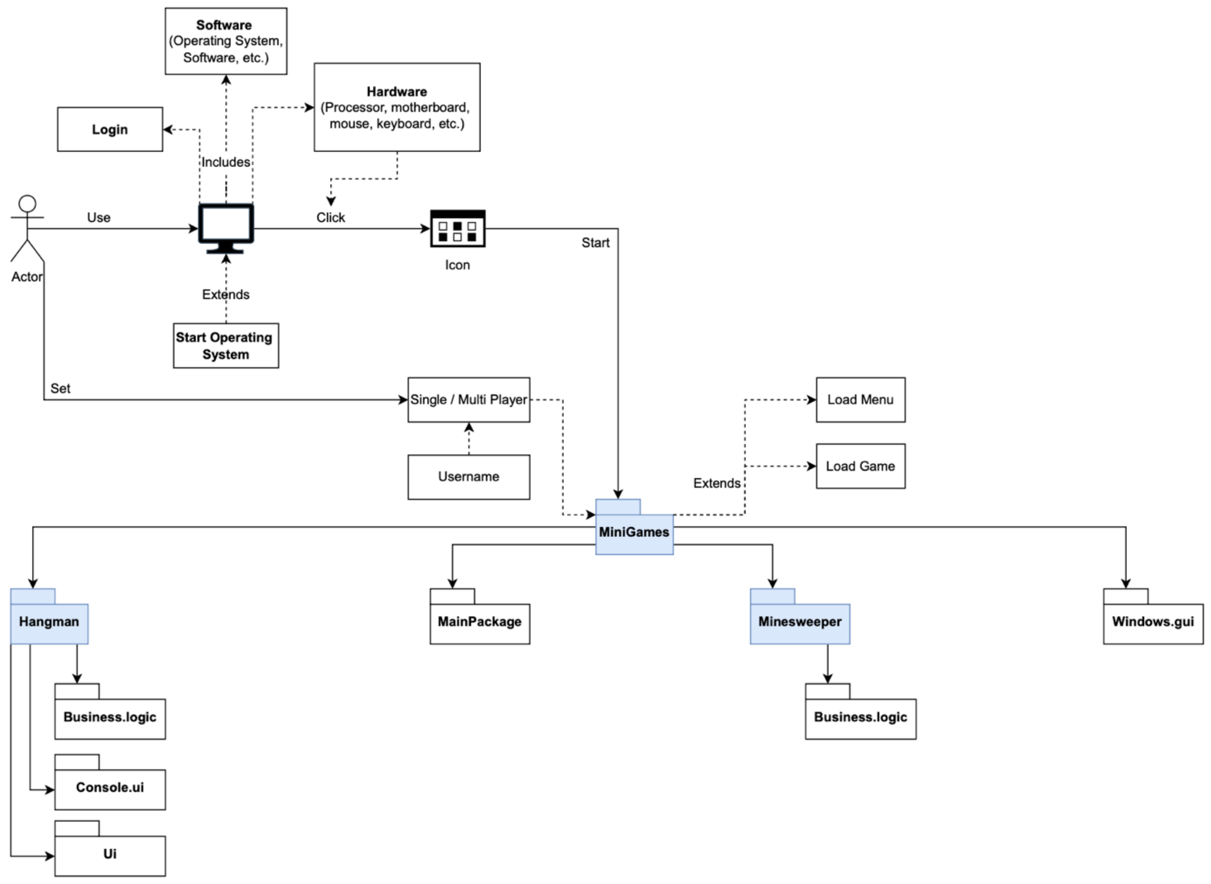
It is available in the git repository as:
uml_final.png (*package UML*)



UML

Use-Case-Diagram

It is available in the git repository as:
Use_Case_Diagramm_V2.pdf (*package UML*)



Statements

Architecture

Division into game logic (package *business.logic*) and independent user interfaces for GUI (package *windows.gui*) and console (package *console.ui*)[1] . The game logic for single player and multiplayer is derived from the abstract Hangman class. Certain functions must be overloaded.
For the user interaction between the business logic and the user interfaces (console, GUI) an interface class *HangmanBasicUI* was defined (interface, package ui).
This interface must be implemented by the *HangmanConsole* and *HangmanWindows* classes. With the *UIFactory* class and the factory method design pattern, the extension with additional user interfaces such as WEB becomes independent.
The GUI class implements the Java class *javafx.application.Application*. The controller classes for the GUI are in the windows.gui package. The associated fxml files are in the *resources* package. The splash screen controller is in the Controller class. The other controllers are generated from here. Data is exchanged between the controllers using the *UserGameData* class.

Clean Code

Members of a class are only accessed via getter and setter methods. The getter methods do not return writable objects.
Constants are declared as enums.
Methods that do not require instantiation of the class are declared static.
Dependencies between classes are minimized and reduced to base types.

Tests

Tests are created with *JUnit*. The tests are in the package *test*.
Both positive tests such as *testCalculateMove_ContainLetter()* and negative tests such as *testCalculateMove_OutOfLevel()* are implemented.
Tests are mainly implemented for the game logic classes such as *TestBusinessLogicHM* and *TestBusinessLogicMS*. For the UI classes, the help functions for checking user input have been implemented, such as *TestHMUtilityClass*.

GUI(JavaFX)

The GUI was created with the JavaFX Scene Builder. The controllers are in the *windows.gui* package. The *resources* package contains the fxml files for describing the scenes. Complex GUI elements such as the *ComboBox* with a variable list and additional input options are used (see *ControllerMultiplayerChooseUsername* and *ControllerSingleplayerChooseUsername*). In addition, the game history is supported with JPG images for the current score, such as *ControllerHMPlayerOne*. The GUI is nested, the controllers call each other.
The user names are only used in the GUI and are therefore only available there.
User input is checked for correctness. If an incorrect entry is made, suggestions for correction are made to the user.

[1] The UI for the console was only programmed as an example for the single player mode because a console UI was not part of the task. But it should show the architectural approach.

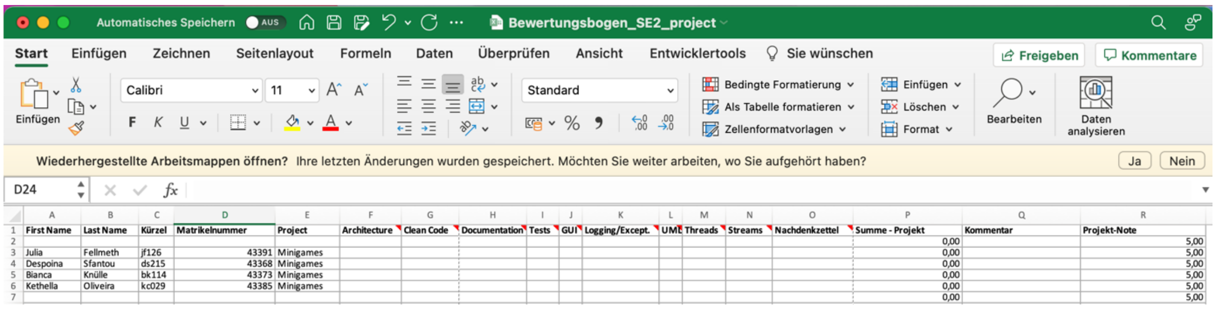
Logging/Exceptions
After the dependency the logging was created using an XML file (log4j2.xml). The level of detail started at level INFO and is indicated by the levels INFO and ERROR. They are being shown on the console and saved in a file named "minigames.log".
In the file and the console will appear a date, time, log level, class name, and the log message. The message can be simple, or it can also receive parameters to make it more responsive. Besides, the threads messages are also being printed and saved as logs. For example, it is used for timer in class *ControllerBSGameMode*, indicating that thread has started and stopped.

UML
The UML was assembled according to the classes and its relations. The + (plus) signal indicates that the class/methods/properties are public, the - (minus) signal indicates that they are private and # indicates they are protected. The first section of information in the packages tags the fields, after comes the methods and constructors, and ultimately comes the properties. When the class indicates an implement from another, it is signed with a green dotted line, and if the relation is an extension, it is indicated by a blue line. Furthermore, the class colors also indicate which package this class belongs to. In the “Use_Case_Diagramm_V2” file it is also possible to see the user’s perspective when accessing the application, a model developed at the beginning of the assignment to “draw” the first ideas of the project.

Threads
Threads are used in Beesweeper by using a Timer to count up the seconds. A thread starts when one selects a game mode (easy, medium or hard) and thus starts a new game. To create the Timer a class named *BSTimer* was created, which has the function of a runnable and counts up the seconds. The thread is initialized in *BSControllerGameMode*, which is also the controller of the runnable, and is started in the *init()* method. If one loses or goes back to the start menu, the timer stops by calling *requestStop*().

Streams und Lambda functions
Streams are used in the *ControllerMultiplayerChooseUsername* and *ControllerSinglePlayerChooseUsername* controllers. The user names are entered in the combo box with the forEach() method (see method *initialize()*). A parallel stream is used to search the list of usernames for existing names (see the *ChooseWord_ButtonPressed()* method).
The stream's *forEach()* method is used to persist the user names on disk (*UserNames.txt* text file).
All streams use lambda methods to check, e.g. for equality of the elements of the stream.

completed evaluation form
It is available in the git repository as:
Bewertungsbogen_SE2.pdf (*package Documentation*)



Bewertungsbogen_SE2_project

Wiederhergestellte Arbeitsmappen öffnen? Ihre letzten Änderungen wurden gespeichert. Möchten Sie weiter arbeiten, wo Sie aufgehört haben?

D24

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	First Name	Last Name	Kürzel	Matrikelnummer	Project	Architecture	Clean Code	Documentation	Tests	GUI	Logging/Except.	UML	Threads	Streams	Nachdenkzettel	Summe - Projekt	Kommentar	Projekt-Note
2																0,00		5,00
3	Julia	Fellmeth	jf126	43391	Minigames											0,00		5,00
4	Despoina	Sfantou	ds215	43368	Minigames											0,00		5,00
5	Bianca	Knülle	bk114	43373	Minigames											0,00		5,00
6	Kethella	Oliveira	kc029	43385	Minigames											0,00		5,00
7																0,00		5,00