CE203\_ss19021\_Assignment2

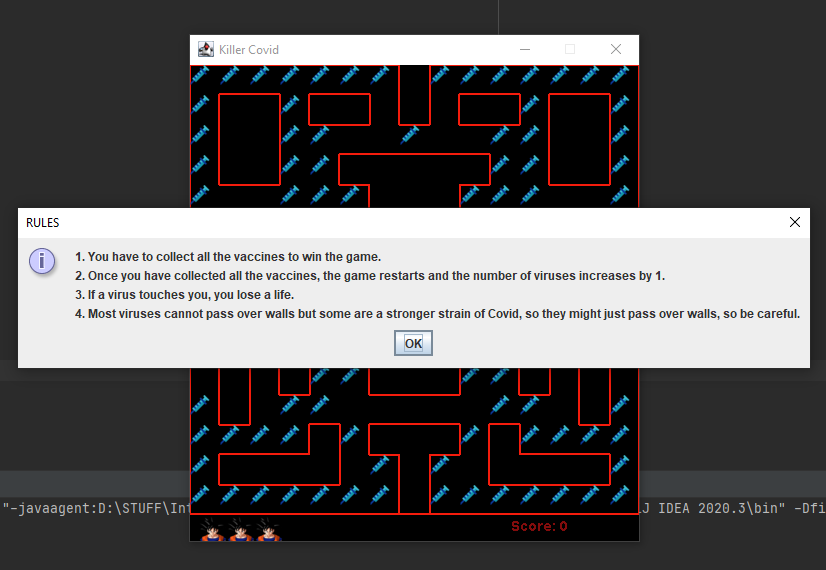
Name: Shane Ethan Steer

Student ID: ss19021

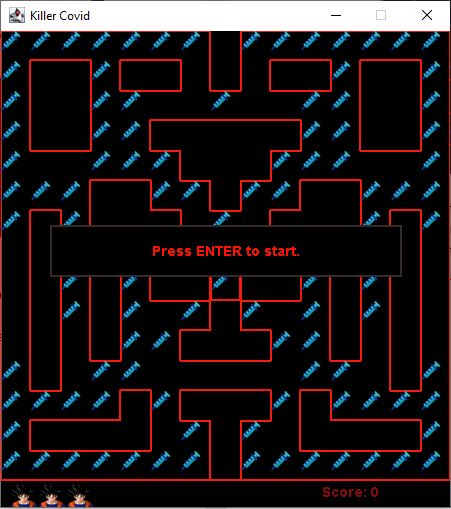
Assignment Report

The Game

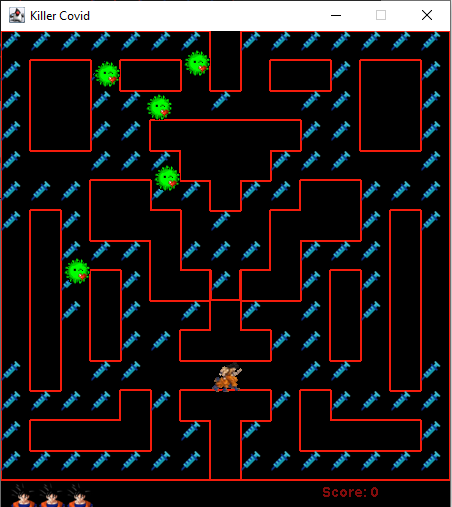
First of all I want to say that the code for this game was inspired by <http://zetcode.com/javagames/pacman/> but was modified the assignment brief and additional features were added.



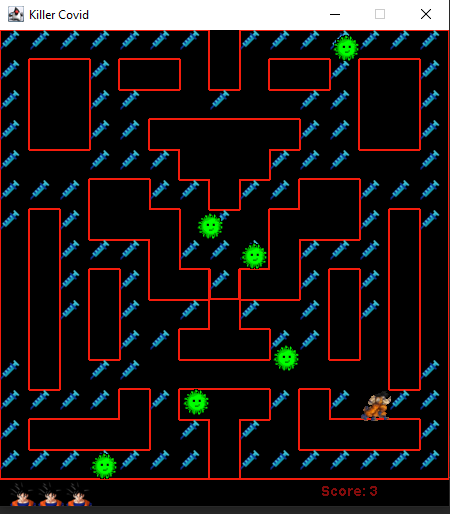
1. When the run button is pressed in IntelliJ the rules for the game pop up in the form of a JOption pane.
2. After the player reads the rules, the ok or close button is pressed and the rules closed.



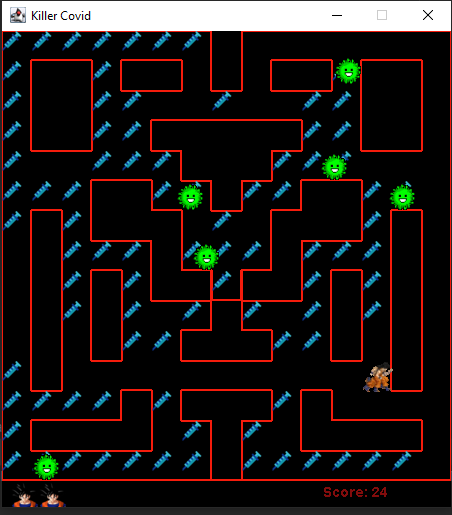
1. The intro screen is shown and asks the user to press the ENTER key to start the game.



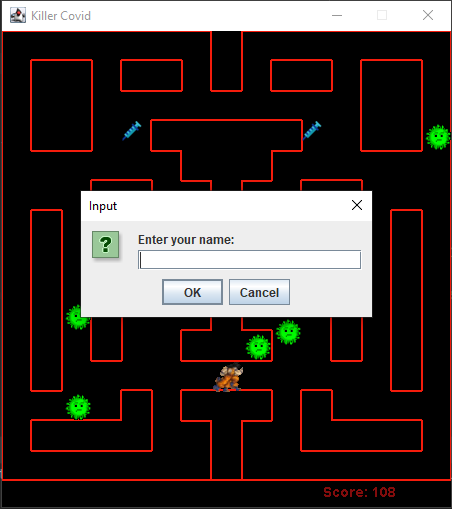
1. When the game starts, the number of lives left is displayed on the bottom right side of the screen and the score is displayed on the bottom right side of the screen.
2. The players character starts at the bottom centre of the screen while the viruses start at the top left of the screen.
3. The viruses moves randomly and at random speeds.



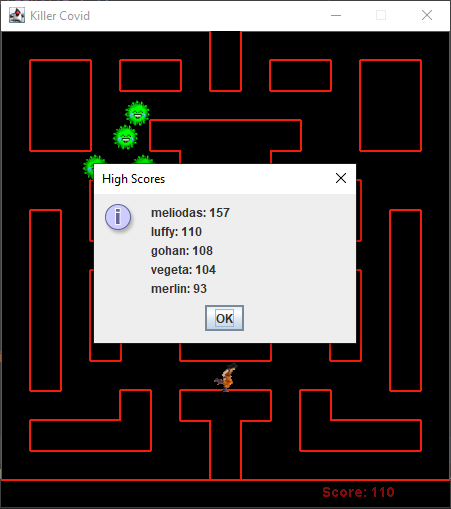
1. The player then uses the cursor keys to move the character.
2. When the character comes into contact with a vaccine, the score is increased by one.
3. The aim of the game is to move on to collect all the vaccines and avoid the virus at all costs to move on to the next level.
4. Most viruses cannot move through barriers but some can as they are a new strain of Covid-19 as shown in the above picture. This was originally a flaw when I resized the window but decided to make it a feature as a new strain of Covid-19 was introduced.



1. When the character comes into contact with a virus, the character loses a life and the number of lives displayed at the bottom of the screen is shown.
2. The character and the viruses start at the starting position again and the game continues.



1. When a player loses all of their lives, they enter their names.

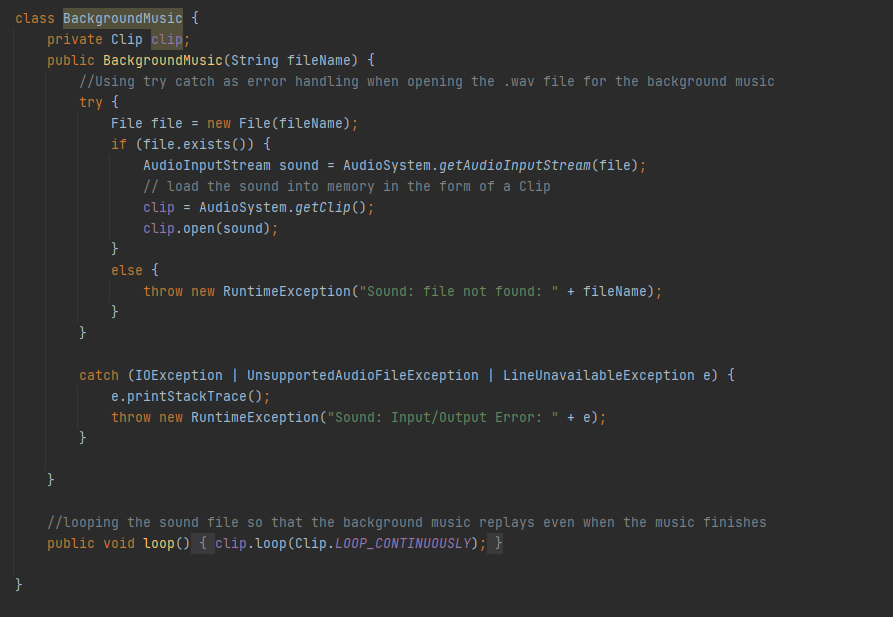


1. A pop up window then appears showing the top 5 scores of the game of all time.
2. If the player manages to collect all the vaccines, the games moves on to the next level and the number of viruses is increased by 1.

Additional Features

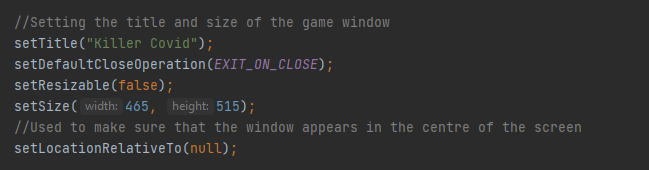
Some additional features which were added to my game were background music and making some of the Covid-19 pass over walls as there was a new strain introduced.

Feature 1



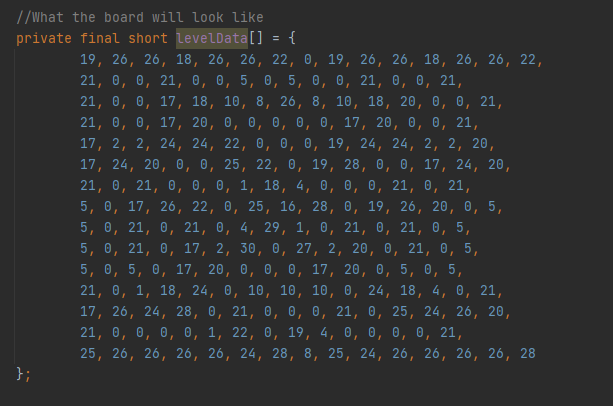
This is the class for the background music. The file for the sound is loaded into memory using a clip. If the file is not found an exception is thrown. Else we call the loop function which just loops the sound file continuously until the program is closed. The background music starts playing as soon as the file is run. The music is a little loud so please turn your volume down if you run the file.

Feature 2

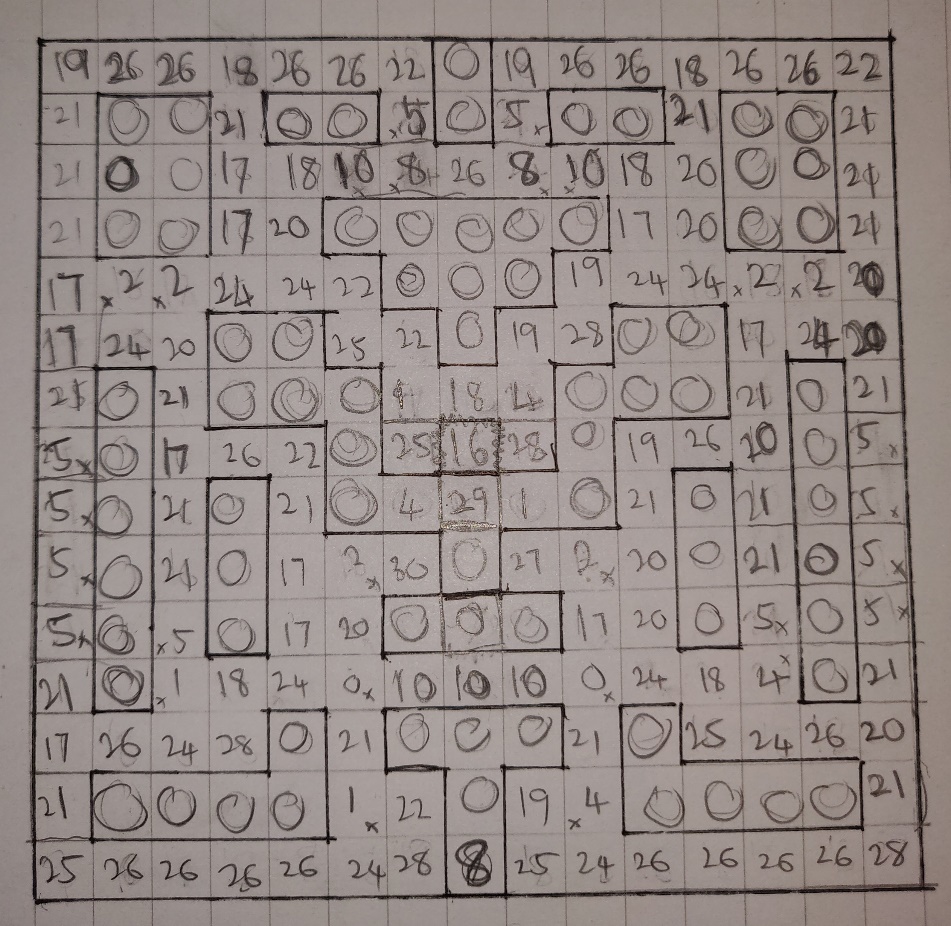


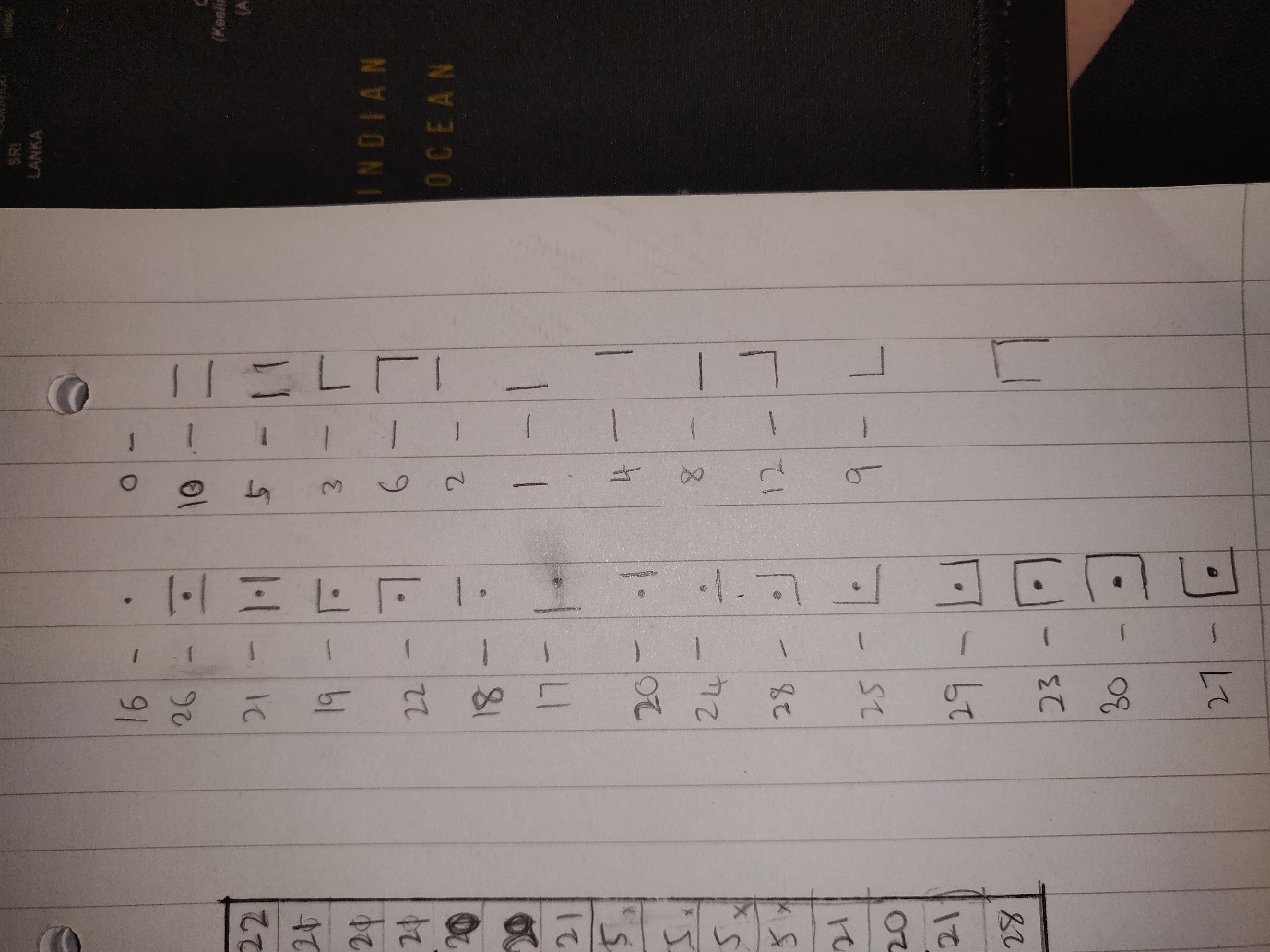
For some reason when I resized my window, some of the viruses were passing over the barriers. I tried to fix it but could not find out what was wrong. Instead, I made it a feature of my game as we recently discovered that there was a new strain of Covid-19. This makes the game a little harder as you do not know which one of the viruses is a new strain and will pass over the barriers.

Feature 3



This is what the board will look like. I had to figure out what the numbers meant. To make it easier, I will include more images below of how I worked it out and explain what the numbers mean.





1. The number 0 means that there is nothing to be drawn in that block.
2. The number 1 means that a left barrier is drawn.
3. The number 2 means that a top barrier is drawn.
4. The number 4 means that a right barrier is drawn.
5. The number 8 means that a bottom barrier is drawn.

If you want to combine barriers, you just need to ass the numbers together.

1. For example, if you want a top and bottom barrier, you just add 2 and 8 together and you get 10.
2. The same goes for a left and right barrier. You just need to add 1 and 4 together and you get 5.
3. A top and left barrier will be 1 + 2 which is 3.
4. A top and right barrier will be 2 + 4 which is 6.
5. A bottom and left barrier will be 4 + 8 which is 12.
6. A bottom and right barrier will be 1 + 8 which is 9.

Next, I will explain barriers and points.

1. The number 16 means that is a point on its own.
2. Hence, 16 + 1 = 17 means that a point with a left barrier will be drawn.
3. 16 + 2 = 18 means that a point with a top barrier will be drawn.
4. 16 + 4 = 20 means that a point with a right barrier will be drawn.
5. 16 + 8 = 24 means that a point with a bottom barrier will be drawn.
6. 16 + 2 + 8 = 26 means that a point with a top and bottom barrier will be drawn.
7. 16 + 1 + 4 = 21 means that a point with a left and right barrier will be drawn.
8. 16 + 1 + 2 = 19 means that a point with a top and left barrier will be drawn.
9. 16 + 2 + 4 = 22 means that a point with a top and right barrier will be drawn.
10. 16 + 4 + 8 = 28 means that a point with a bottom and right barrier will be drawn.
11. 16 + 1 + 8 = 25 means that a point with a bottom and left barrier will be drawn.

Shortcomings

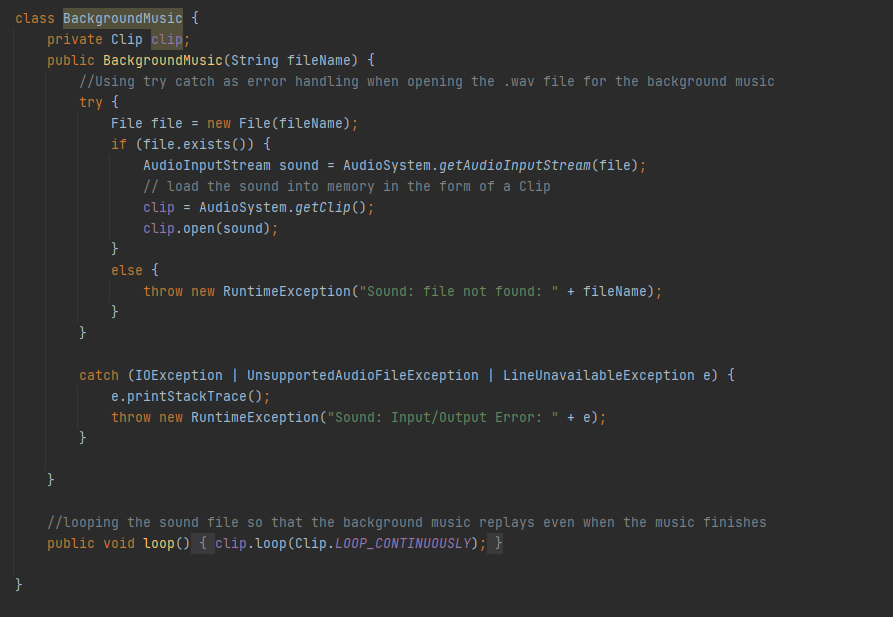
At first, I created a GUI to pop up and get the names of the players before the players started the game. The name would be written to the Scores.txt file first then the scores would be allocated when they had lost all their lives. I then realised that this would cause problem in displaying the scores file as a score would not get recorded if the player decided to quit the game before they ran out of lives. I then decided that it was easier to just use a JOption pane That pops up after they finish all their lives to get their names when the game ended. This was then easier to put the name and scores in a single variable and write both the name and scores to the Score.txt file at the same time. This then solved the problem of names being recorded but not scores if someone quit mid game.

The second shortcoming was that the player would just keep running even when the cursor key was released. I realised that I had to set the speed of the character to zero each time the key was released and that I had to initialise the speed again when the cursor key was pressed again.

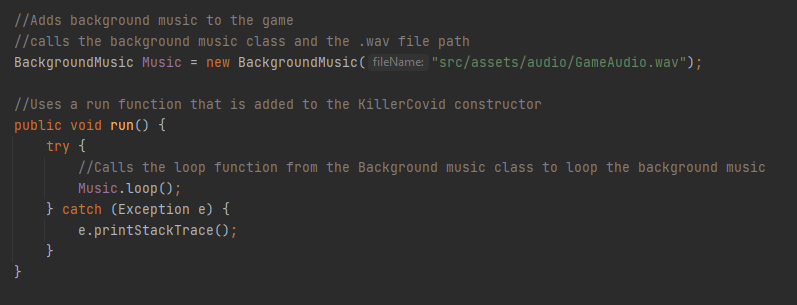
The third shortcoming was to add in a power up or after you collect a certain amount of vaccines you could kill the viruses. I tried many ways but could not figure out how to do it and I was afraid I was not going to finish the assignment in time. If I had more time, I would have liked to try and implement this.

Design Pattern

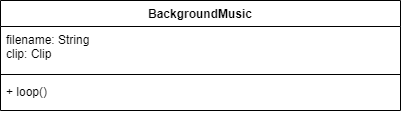
A design pattern that I implemented in my code was cohesion. Cohesion makes code well managed, easy to read and well defined. Having code in separate classes that can be grouped together later on is an example of High Cohesion. For example, in my background music and PLayerScores class , I am able to separate it from the drawing class hence making the code more concise and readable. The related functionalities will be in their own respective classes.



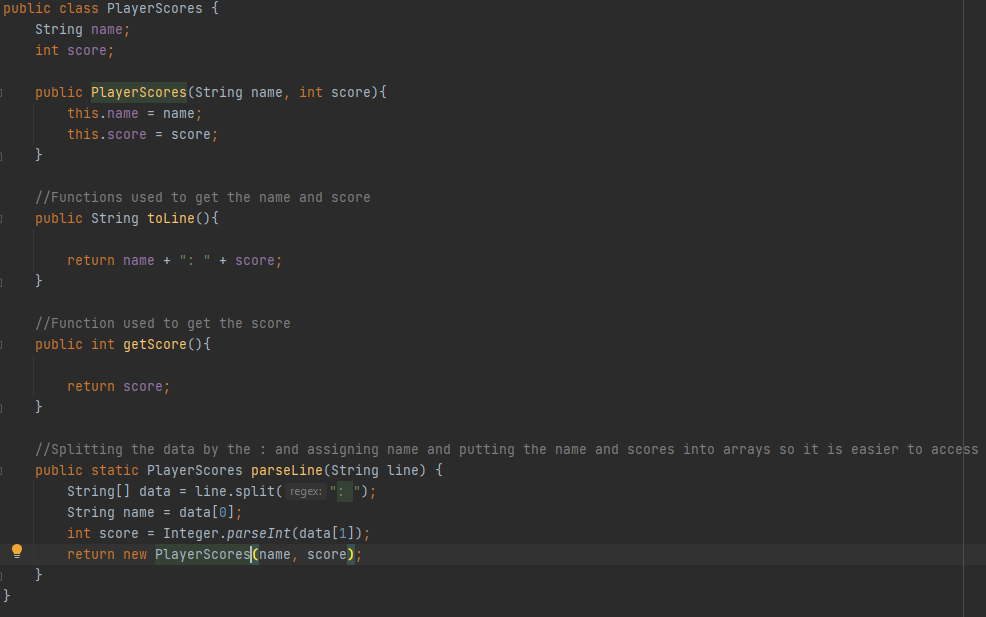
This is the background music class. In here I get the filename and store it in the form of a clip. If there is no sound file, an exception is thrown.



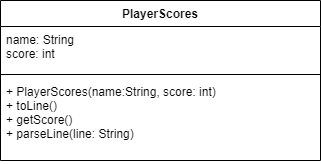
This is where is I call the background music class in the game class



This is the UML diagram for my BackgroundMusic class.



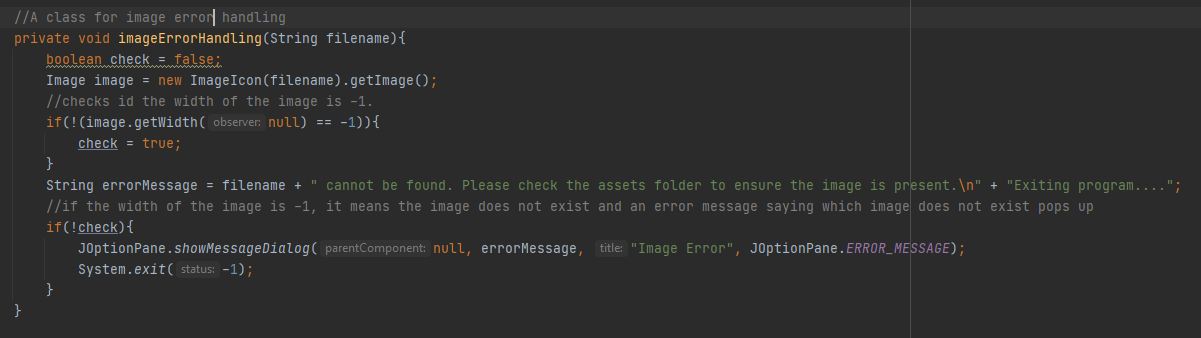
This is the PlayerScores class. In here I get the name and score and split the data by the “: ” and put them in an array.



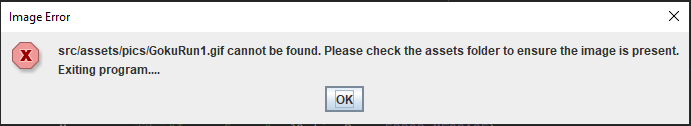
This is the UML diagram for my PlayerScores class.

Error Handling

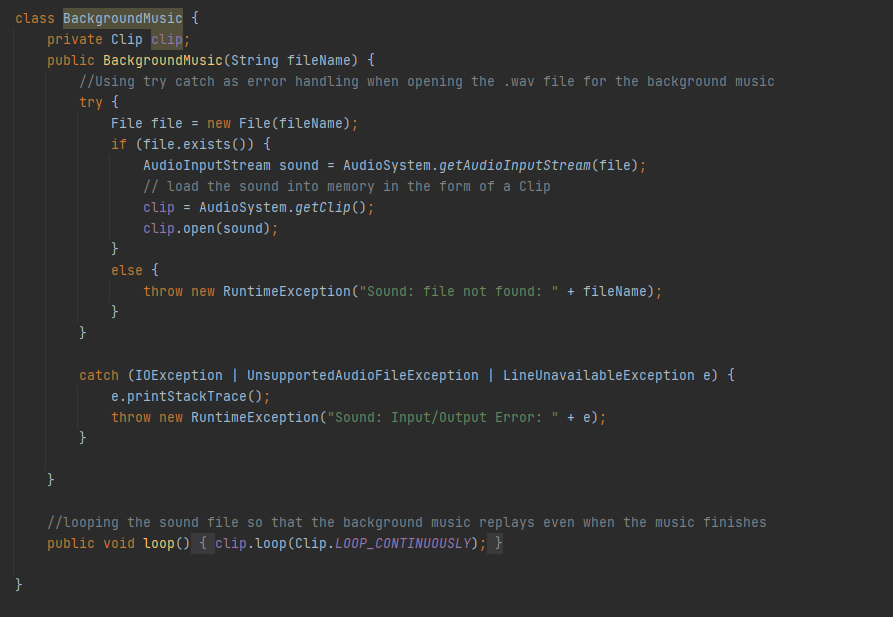
I have done error handing for the image and sound files.



This is the error handling for the image files.



If a file does not exist, a JOption pane pops up with an error message saying which file does not exist and exits the program when the ok button is pressed.



A run time error is shown when the audio file cannot be found.

Database

A more appropriate way to store the players high scores would be to implement a SQL Database in Java which is JDBC (Java Database Connection). This would be more efficient as the user file Score.txt could get corrupted as I had experienced earlier and will also not allow the user or manager to edit the Scores.txt file to their liking and change scores or add in new ones manually. We would need to load a driver class into the database system and connect it to the database then create a statement to store the names and scores of players to the database. We can then sort the names and scores stored in the database and display it using a JOption pane or on the display screen.

Pseudo Code

import java.sql.\*;

try{

Load external driver class for the database system;

Connect the driver to the Database;

}

catch(Exception){

Print error;

}

Create statement

try{

Create a statement object;

}

catch(Exception){

Print error;

}

//writing the name and scores to the database

Name = Players name

Score = Players score after losing all lives

Statement to store the name and score in the database

JOptionPane(“Name: ” + statement selecting the players name + “: ” + statement selecting the corresponding score);