SPŠE Ječná

Informační technologie

Roulette of Despair

Marek Sedláček, C2a

1. # Point of project

   The point of this project is to make a game on the style of Russian Roulette for one player, where they are up against a different NPC. The player should be able to choose if they'll shoot themself or the opponent, and the program will say, if the game continues or one died.

2. # Description

   ## 2.1 Story

   The protagonist is kidnapped and stuck in an unknown room, with a different person on the other side of table from them. On the table they can see a gun, with which they're instructed to play russian roulette against eachother. Whoever wins gets a huge cash price, as for the other, they die. The protagonist takes up on this opportunity as they are in severe debt.

   ## 2.2 Characters

   There are a few different characters against which the protagonist can go against, 3 to be exact.
   For one, Emily. She's a young woman with shoulder-length hair, who doesn't really like the concept of this game and seems really worried. She can be easily distracted.
   Number two, Oscar. He's a young man with short hair, who doesn't really trust his opponent. He's not as easily distracted as Emily, but sometimes can be caught off guard.
   Number three is Paige. A middle aged woman with long hair tied into a ponytail. She's extremely vary of her surroundings, and is barely caught off guard, so she's barely distracted.

   ## 2.3 Game Mechanics

   The player has an option to check their inventory, where they have one-time use items. These can help them with either

distracting the opponent or even seeing if there's a bullet in the current barrel. Afterwards, the player can choose to either shoot the opponent or themself, however if they choose to shoot the opponent and there isn't a bullet, the opponent will be more likely to try and shoot you aswell.

3. System Requirements

The project was made with the programming language Java, with JDK 24. Additional libraries or external resources are not needed to be downloaded. Everything in the project is solved with Java libraries. The project can be opened in any program supporting Java, for example Intellij IDEA.

4. Program Structure

The program is made with JavaFX, where classes are in packages depending on their categories, however they all can communicate with eachother. Main is the class with which the whole project is started. Next up is GameMenu, it's constructor is activated through Main. GameMenu Class makes a screen pop up, where the player can choose to start or exit the game. Afterwards there's the class Game itself, which initializes all the information about the game and the screen, like the width and height of it. It also uses the factory method of the abstract Level class, which chooses what level starts first depending on the level Integer. Classes Level1, Level2 and Level3 all extend the abstract Class Level, each being more difficult and including different NPCs. In the abstract Class Level, all the information, buttons, scenes, screens and even cutscenes are all initialized in it and used. With the Level Class also communicates the Inventory Class, which initializes and shows all the items the player can use in the game. The items each have a button made and the player can hover over it to see what it does, and then

click. There's also a button with which the player can return back to the level. The inventory is initialized in Game's constructor, and then passed and used in Level's constructor. Another Class used is the LoadingScreen class, which is used in GameMenu's start( ) method, describing the situation the player is in before starting the game and communicating with Game Class. In a lot of the classes, mostly Level, Thread.sleep is also used to give the player more time to process what's currently going on in the game.

5. Testing data

It's possible to test the program manually by choosing different scenarios in the game. For example, trying out different options between shooting the opponent or self, but also trying out the various items to see if they work as they should. Afterwards, it is important to check if the program correctly establishes the ending of the game and the continuation. It isn't really possible to test out different inputs from the ones required, as user input is not used in the game except for pressing buttons.

6. User manual

The program is controlled by clicking various buttons which each do a different thing. The player can click the gun when in game, where three texts will pop up, to either shoot the NPC, shoot themself or go back to choosing. As the gun's barrel has 6 spots for bullets, and only one is filled, then the maximum turns each level can take is 6 turns. The game goes on like this until the player either dies, or the third level is beaten.

7. Conclusion

I've had multiple problems with writing the program, most of them being me overlooking small mistakes or additional stuff I

accidentally left in. For example, I forgot that I set a few of the buttons and texts to not be visible, so when running the program they simply wouldn't appear. I spent quite a long time figuring it out, before finally finding out about this. Another problem I have had was with the cutscenes, as they wouldn't show properly, but it'd still end the program after the time passed as intended. Turns out I needed to make each picture a different Scene and make the stage show each scene after a bit of time thanks to Thread.sleep( ). However, this project helped me a lot with how JavaFX works, as this was my first time using graphics in a Java project. Overall, I would rate this project as beneficial.