**ICT221 MiniDungeon Game Report**

**2025 Semester 1**

University of the Sunshine Coast

Queensland

|  |  |
| --- | --- |
| Author Full Name: | Tadas Grigutis |
| Author Email: | T\_g084@student.usc.edu.au |
| Author Student ID: | 1185919 |
| GitHub Repository URL: | <https://github.com/UniSunshineCoast/ict221-mini-dungeon-Tawood002> |

\* The report has 20% weight of the assignment. The number in each section title shows the weight of that section. Please refer to the assignment instruction file and the marking rubric for detailed requirements for the report.

# Introduction (2%)

This project implements a simplified version of a 2D dungeon crawler game, developed using Java. The objective of the game is to navigate through two 10x10 maze-like dungeon floors, collecting gold, defeating enemies, and reaching the ladder to exit the dungeon, all within 100 steps per floor. Players begin with 10 health points and lose if their health drops to 0 or they reach the maximum number of steps. The game features melee and ranged enemies, health potions, traps, and a high score tracking system that records date of the top 5 runs based on gold collected.

# UML diagrams (10%)

(Diagrams with some text explanations. <https://www.draw.io> is recommended.)

A diagram of a game

AI-generated content may be incorrect.

The overall code works and can be followed however, is not perfect. Many areas in the code have been bloated to fix issues and keep the game running smoothly, instead of a clean-cut environment housing all functions in easy-to-read areas.

For instance, the “mobs” and “objects” functions come to mind, as all first-floor objects can be found in the “GamePanel” class and mobs in “MobManager” class. Whereas, for the second floor the objects and mob functions are both placed into the “Ladder” class. This can be attributed to time restraints, lack of planning for the UML design and overall Java coding capabilities, as what work, usually stayed.

The core GamePanel loop and implementation of the hero’s sprite (.png images) were based RyiSnow’s video guides on YouTube (RyiSnow, Sprites and Animation - How to Make a 2D Game in Java #3, 2022) (RyiSnow, Game Loop and Key Input - How to Make a 2D Game in Java #2, 2022). As the game was built periodically, functionality was prioritized meaning the game was always in a state which worked. Thus, there no testing Junit testing done for any of the classes or methods.

There wasn’t much planning into the design or layout of the floors. Both floors had the same gameplay loop, with their own unique approaches allowing them to be completed with relatively easy. However, due to placement and game mechanic of the ranged Mutantes having 50/50 chance to miss the play, this introduced new aspects, which in turn added replay-ability. This due, to the player having the option to take risks to fight monsters guarding extra gold or health potions, with every play through being unique as it could either punishes or reward the player for their actions. This made so the play had to weigh the risk and rewards, thus put more thought into their movements.

The difficulty meter wasn’t added due it potentially taking away from the game and the player’s experience. The is already quite easy to traverse, the only thing to make it easier would be removing the ranged Mutants or all Mutants entirely, this would effectively make the game less exciting and hollow like. Instead, I’d propose that difficulty meter would instead increase the amount of total mobs, make them stronger (doing 3 hp instead of 2hp) or creating a new entirely for veteran/experienced players.

Each map was created via using .txt file was a square block of numbers. This block was then placed into a method which read and assigned each number a tile and a function. This allows for easy scalability and future advancements.

# GUI sketches (5%)

(Sketches with some text explanations. Sketches can be made with paper and pencil.)

 A screenshot of a video game

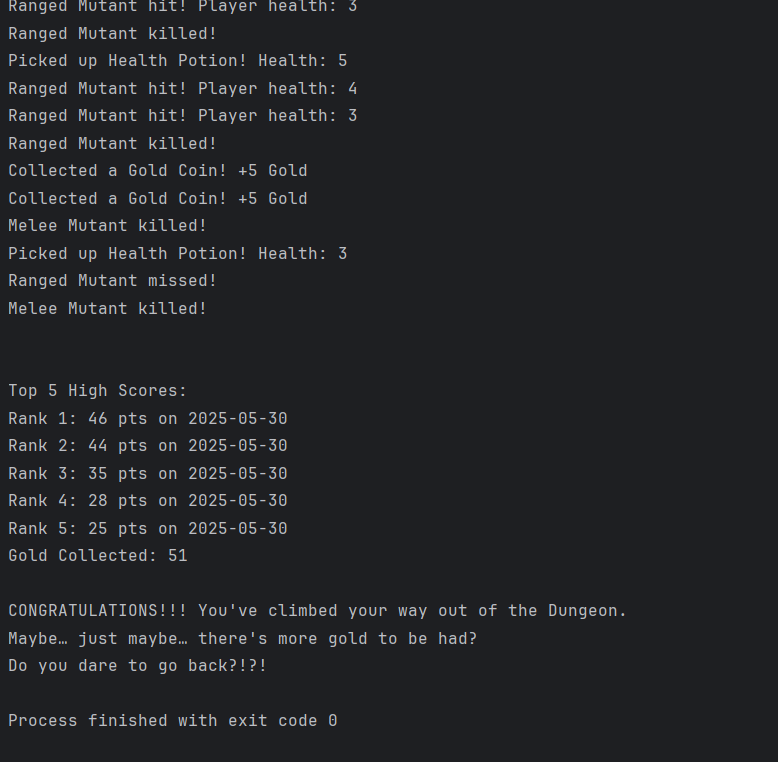
AI-generated content may be incorrect.

The GUI was obtained from a free online resources pack, it was chosen as it fit the theme of gameplay. (0x72, 2018) The screen shots are from my game.

The main game panel was rendered as a 12x12 block, with a 10x10 area for the player to manoeuvre, surrounded by boarder blocks. Each block was 64 pixels big, as the original size of 16 pixel was too small (thus multiplied by 4). Due to average monitor screen being able to display a resolution of 1920x1080, the chosen resolution of the game was 768x768. As this comfortability fits on all modern screens and provides the player with visual real estate to play the game.

Some symbols were changed due to stylistics preferencing (these being: the player (P), walls (#), Melee Mutant (M) and Ranged Mutant (R) also tile being placed to display his attack range.) These were replaced with models from the mentioned resource pack. This was done achieve a more intriguing environment and feedback loop for the player, such as the hero model changing direction according to keystrokes.

The Console Panel was utilised as a text log showing messages like “You picked up x gold.” or “Ranged mutant missed!” to help the player understand their environment. Example of the console is below.



# Reflection (3%)

This project provided valuable experience in designing and implements Java applications. One of the most challenging aspects was ensuring the gameplay loop and mob functionality worked correctly without affecting player performance or collision detection.

If I were to repeat the project, I would prioritize implementing the JavaFX GUI earlier to catch integration issues sooner and incorporate menu-like features, for player satisfaction. These features would allow players to reload saves, select a difficulty, view and potentially change gameplay controls and overall create a more game like experience.

Additionally, there would be more time put into the layout of the UML game engine to test all available classes and methods to ensure the run and reads smoothly. Although these not all criteria weren’t reached, the core gameplay and functionality are ironed out and solid. As of submitting the game there is no known bugs.

# Works Cited

0x72. (2018, May 09). *16x16 Dungeon Tileset*. Retrieved from https://itch.io/: https://0x72.itch.io/16x16-dungeon-tileset

RyiSnow. (2022, Dec 22). *Game Loop and Key Input - How to Make a 2D Game in Java #2*. Retrieved from Youtube.com: https://www.youtube.com/watch?v=VpH33Uw-\_0E

RyiSnow. (2022). *Sprites and Animation - How to Make a 2D Game in Java #3*. Retrieved from youtube.com: https://www.youtube.com/watch?v=wT9uNGzMEM4