Name

Fahima Afrin Anuva (2321049042)

MD. Minhaz Ahamed Rohan (2232065642)

MD. Tushar Khan (2232344642)

Text-Based Dungeon Quest Game

Abstract

Dungeon Quest Game is a text-based adventure game developed using **Java** and **Java Swing** to provide an interactive dungeon exploration experience. The game is designed around core **Object-Oriented Programming (OOP)** principles, ensuring modularity and scalability. Players navigate a randomly generated dungeon, where they encounter treasures, enemies, and traps, all while managing their health. The game uses event-driven gameplay to create a dynamic and engaging environment. Through the use of **exception handling**, the game ensures robustness by preventing crashes due to invalid inputs. This project showcases fundamental programming concepts such as **encapsulation**, **inheritance**, and **polymorphism** while providing a user-friendly graphical interface. Future enhancements could include multiplayer features, advanced graphics, and expanded game mechanics. The game's codebase is structured for easy modification and expansion, encouraging community contributions

1. Introduction

This report provides an in-depth overview of the **Text-Based Dungeon Quest Game**, a simple yet engaging adventure game developed using Java. The game simulates a dungeon exploration experience, where the player makes decisions that affect the outcome of the story. The primary purpose of the game is to demonstrate fundamental object-oriented programming (OOP) principles, exception handling, and graphical user interface (GUI) design.

Project Objective

The objective of the project is to create a text-based adventure game that includes interactive choices, a combat system, and dynamic story progression. The game is designed to demonstrate core programming concepts, such as OOP, exception handling, and GUI development, while also providing an engaging experience for players.

2. Project Structure

The **Text-Based Dungeon Quest Game** is structured into several key components, each responsible for a distinct aspect of the game:

- **Game Logic**: Handles the game's mechanics, such as player health, combat, and decision-making.
- **GUI (Graphical User Interface)**: Manages the visual elements of the game, displaying information to the player and capturing user input.
- **Exception Handling**: Ensures that potential errors or unexpected events are properly managed to prevent the game from crashing.

Core Components

The game consists of the following main components:

- **JFrame**: The main game window that holds all other components.
- **JPanel**: Panels used to group components, such as the title, buttons, and player stats.
- **JButton**: Buttons representing the player's choices, which trigger different actions in the game.
- **JTextArea**: Displays the game's text, such as narrative and dialogue.

3. Implementation Details

3.1 Object-Oriented Programming (OOP)

The game is designed using object-oriented programming principles, which ensure a modular, maintainable, and scalable codebase. The key OOP concepts implemented in the game include:

- **Classes and Objects**: The game is organized into classes such as `Game`, `ChoiceHandler`, and `TitleScreenHandler`. Each class encapsulates specific functionality.
- **Encapsulation**: Data such as player health, weapon, and position are encapsulated within the `Game` class, providing control over how these values are accessed and modified.
- **Inheritance**: While not explicitly demonstrated, the game could be extended by creating subclasses for different character types or game scenarios.
- **Polymorphism**: The `ChoiceHandler` class demonstrates polymorphism by handling different player choices through method overrides, depending on the current game state.

3.2 Exception Handling

Exception handling is implemented to ensure smooth gameplay and prevent crashes in the event of unexpected inputs or errors. The `try-catch` blocks are used to catch potential exceptions that could arise from user interactions or invalid game states.

```
try {
    createGameScreen();
} catch (Exception e) {
    System.err.println("Error during title screen handling: " +
e.getMessage());
    e.printStackTrace();
}
```

This exception handling ensures that any errors during the execution of game logic (such as displaying the title screen) are caught and logged without crashing the game.

3.3 Graphical User Interface (GUI)

The GUI of the game is built using Java Swing components, which include:

- **JFrame**: The main window for the game.
- **JPanel**: Containers for organizing components such as the title, buttons, and player stats.
- **JTextArea**: Displays the game's narrative and dialogue.
- **JButton**: Allows the player to make choices, such as attacking or talking to NPCs.

The GUI is designed to be intuitive and user-friendly. It provides a dynamic interface that updates the display as the game progresses, reflecting the changes in the player's health, inventory, and game scenario.

Key GUI Features:

- **Title Screen**: Displays the game's title and a "Start" button to begin the game.
- **Choice Buttons**: Presents multiple choices to the player, which determine the story's direction.
- **Player Stats**: Displays information such as the player's health (HP) and current weapon at the top of the screen.
- **Dynamic Text**: The main area of the window updates with text describing the game's events and player choices.

Example of GUI setup:

```
window = new JFrame();
window.setSize(800, 600);
window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
window.setLayout(null);  // Using null layout for custom positioning
window.setIconImage(logo.getImage());
con = window.getContentPane();
con.setBackground(Color.red);
```

4. Game Features

The **Text-Based Dungeon Quest Game** includes several key features that drive the gameplay:

- **Multiple Choices**: The player is presented with a series of choices at various points in the game, which influence the narrative and lead to different outcomes.
- **Combat System**: The game includes a simple combat system where the player can fight monsters. Combat damage is determined by the player's weapon and a random number generator.
- **Player Stats**: The player's health (HP) and weapon are displayed throughout the game, allowing them to track their progress.
- **Item Collection**: The player can obtain items, such as a Silver Ring or a Long Sword, which affect their stats and gameplay choices.

4.1 Game Flow

The game progresses in stages, starting with an introduction at the town gate. The player can choose to talk to or attack the guard, explore different locations, fight monsters, and collect items. The game ends when the player wins by defeating the monster or loses if their health reaches zero.

5. Conclusion

The **Text-Based Dungeon Quest Game** successfully implements core programming concepts such as OOP, exception handling, and GUI design. It offers an engaging and interactive experience where the player's choices determine the course of the game. The use of Java Swing for the graphical interface ensures that the game is visually appealing and easy to interact with.

This project demonstrates the ability to design and implement a fully functional game using Java, and it can be extended in various ways to include more complex features, such as additional locations, characters, and a more elaborate combat system.

6. Future Work

While the game is functional, there are several areas for future improvement:

- **Expand the Story**: Adding more locations, characters, and complex decision trees would enhance the gameplay experience.
- **Combat System**: Improving the combat system with more diverse attacks, enemy types, and health management could make the game more challenging.
- **Graphics**: Incorporating images or animations into the GUI would provide a more immersive experience.
- **Save/Load Functionality**: Allowing the player to save their progress and resume later would make the game more user-friendly.

7. Reference

RyiSnow