Sudent Information:

Full Name: Yigit Baslayici
Student ID: GH1042297

Github Repository: https://github.com/TheLuqio/AfterTheCollapseGame

Video Demonstration: 2025-12-18 18-03-55.mp4

Introduction:

The "After The Collapse" Project is a console based text adventure game made in programming language JAVA, which allows players to take on the role of a survivor in a post-apocalyptic environment and make various decisions, manage resources and proceed through various narrative scenes. This project meets four main goals:

Demonstrating the usage of OOP principles in JAVA.

Creating an organized, multi class application design.

Supporting console-based input and output, along with the effective use of time and game's pacing.

Creating an inventory system that allow players to utilize items that they acquire in the game.

Enhancing replayability through the inclusion of random elements.

The project considers it’s problem domain as a form of "Interactive Storytelling" and "Survival Mechanics" where player choices and experience will shape how they progress through the game. This project is important because it provides an example of core JAVA functionality being used in a complete interactive application, rather than simply being an accumulation of individual usage exercises of JAVA.

System Architecture:

The game is entirely based on a console architecture and it does not utilize a graphical user interface or database. All the user interaction is conducted through text in terminal.

Critical Components:

Console based Application: The game was coded to run through the command line, where the game output is made by System.out.println and the game input is provided by Scanner.

Object Oriented Design (OOP): Each class in the project has its own unique task, organizing all the game parts to their own responsibilities.

The primary classes used in the application includes:

Main.java: This is where the game runs.

StoryManager.java: This class coordinates the overall flow of the game scene changes, story progression and player's choices.

Player.java: The Player class has all of the personal information about a player. It contains the player's name, health, stamina and some inventory features.

Inventory.java: The Inventory class has a collection of items stored as an Array List and allows the player to add, remove and search for items within their Inventory.

Item.java; Every Item that a player can use, has a similar structure which includes: Item Name, Item Type, Health, Stamina, Weapon and Effect Value.

Organizing the classes in this way creates an increase in both readability and maintainability of the application, and increases the possible number of increases in the number of ways the application may be expanded.

Implementation

The Development process used to develop the application was an incremental approach where the features were implemented step by step and tested.

Development Life Cycle

Project Planning :

The following items were considered when developing the game:

1) The game concept

2) The game's Story Structure

The Base Class Structure :

Implemented Player Class, Item Class, Inventory Class and Main Game using the StoryManager.

Building the Game Logic :

Implemented scenes as individual Methods. Implemented conditional statements to provide branching Choices in the Game Logic. Java's Random Class was used to provide an element of Surprise to the player when Random Events occurred.

Inventories and Items

The Item Class implemented lookup and usage methods to allow the player to use items. The Item Class Restricted item usage based on item type.

Debug and Refine

Fixed issues related to User Input handling. Confirmed that the player's Health and Stamina values remain in valid ranges.

To enhance the player's ability to read and play the game, some pause functionality added throughout the game to slow down the pace of play.

Tested throughout development to confirm that all features were functioning as intended.

Results

The result of the work is an application that runs from beginning to end and allows players to:

Select a character and starting items from some options

Traverse through many story scenes

Collect items by searching.

Access and utilize health and stamina items stored in the user's inventory.

Random events that differ in each playthrough.

Challenges and Their Solutions

Challenge #1 - Item Not Found Bug

Issue:

Some items were displayed in the user's inventory were unable to be used.

Solution:

This problem was investigated and resolved by correcting the comparison logic within the item lookup function and including input trimming to ensure correct response to user input. Finally, the application was rebuilt, so the correct version was compiled and executed.

Challenge #2 - Scene Flow Skipping

Issue:

Scenes appeared to skip too rapidly and therefore were difficult for players to follow the story.

Solution:

The incorporation of a pause method (utilising Thread.sleep()) and "Press Enter to continue" will enhance both the readability and pacing of the scenes.

6. Conclusion and Future Work

This project showcases how to create a fully functioning console based application using Java through OOP principles. There were many skills gained throughout this project, but the most noticeable were:

Separation of class responsibilities & designing classes correctly

Validation & Ensuring input was entered correctly

Solving problems & Debugging

When building a larger program, understanding How to Manage the Order of Your Program Logic

Future Improvements:

Adding Combat mechanics

Saving & Loading Your Game Progress

Improving Input Validation

Expanding upon the Main Story Line and Scene Components

Graphical User Interface (GUI).

In conclusion: This project has provided an excellent opportunity to gain some hands on experience in developing a properly built Java Program, while also providing a way to apply things learned from in class to a practical situation.