Even Semester (2024)



**BINUS UNIVERSITY**



**BINUS INTERNATIONAL**



**Assignment Cover Letter**

**(Individual Work** **)**

**Student Information :**

**Surname**  **Given Names Student ID Number**

1. Mikael  Jayson 2440032442

**Course Name :** Program Design Methods

**Name of Lecturer(s) :** Jude Joseph Lamug Martinez

**Course Code**  **:** COMP6699

**Class**  **:** L2AC

**Major**   **:** Computer Science

**Title of Assignment** **:** Text-based Java RPG

**Type of Assignment** **:** Final Project

**Submission Pattern :**

**Due Date**  **:** 22-6-2021   **Submission Date**  **:**  22-6-2021

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer’s instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

**Plagiarism/Cheating**

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

**Declaration of Originality**

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

Jayson Mikael

**Text-Based Java RPG**

**Name : Jayson Mikael**

**ID : 2440032442**

**Table of Content**

**I.** Project Specification………………………………………….....................................Page 3

**II.** Solution Design………………………………………………………………………Page 3

**III.** Libraries Used ………………………………….……………....................................Page 4

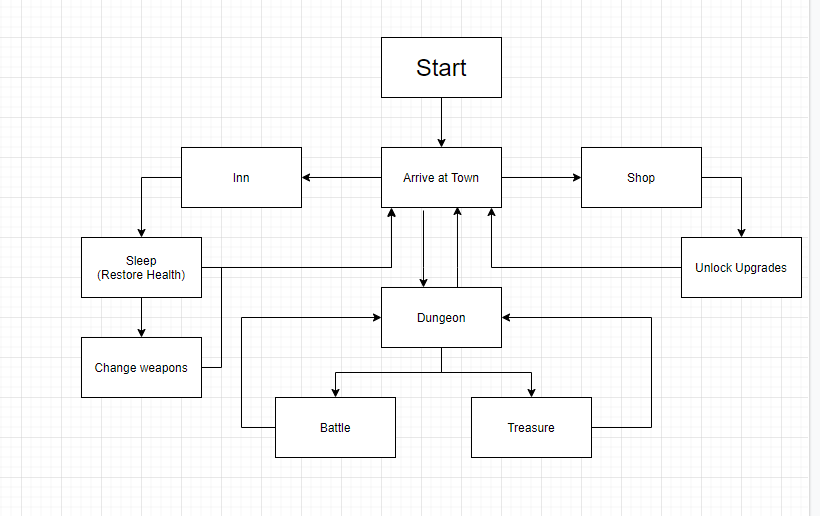
**IV.** Code Implementation………………….……………………………………………..Page 5

**V.** Screenshots/Evidence of Working program………………………………………….,Page 17

**VI.** References…………………………………………………………………………...Page 22

1. **Project Specifications**

The program that I wanted to create is a Text-based RPG using the Java programming language. It’s a simple game that requires specific inputs to make sure the programs run smooth. The program is created using Java and the libraries provided by other people. The game is also created to generally fill people’s time when doing other tasks.

1. **Solution Design**

The flowchart above shows the direction of the user when playing the game. It would mainly start at the town, to which the user is able to sleep at the inn, unlock upgrades at the shop, or go to the dungeon for battles or treasure.

1. **Libraries used**

There are some libraries used to make the program, such as:

1. **Java.util.Hashtable**

This is used to store a key-and-value pair data, mainly the user’s attributes.

1. **Java.util.ArrayList**

This is used to store the user’s unlocks / purchases from the store.

1. **Java.util.concurrent.TimeUnit**

This is used to pause the code and allow some time for the user to slow down and to read the story as some parts doesn’t need specific input but rather time for the user to read.

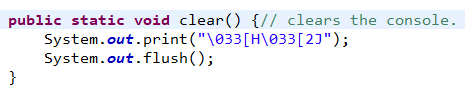
1. **Java.util.scanner**

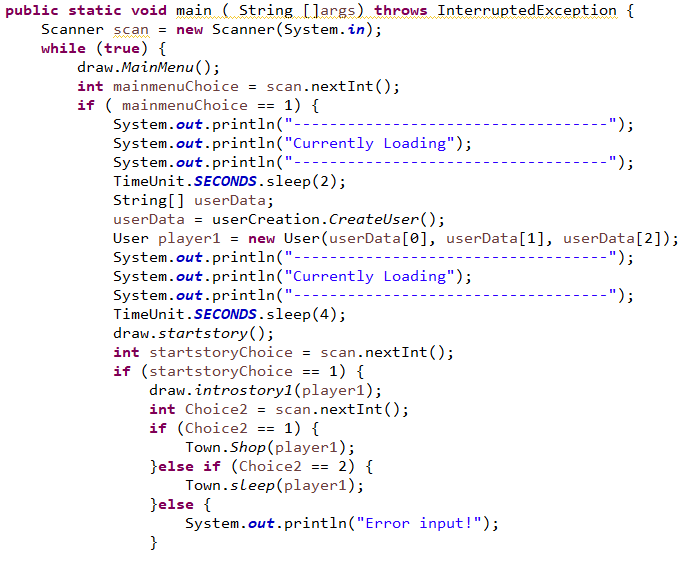
This is used so that the user would be able to input to the program.

1. **Code implementations**

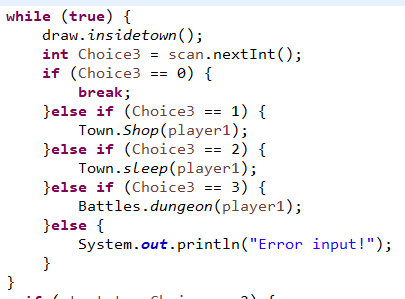
The program is created on Eclipse IDE mainly implementing OOP methods. By separating the code into each of its own elements, this allows easier debugging and easier time for reader to understand specific parts of the code.

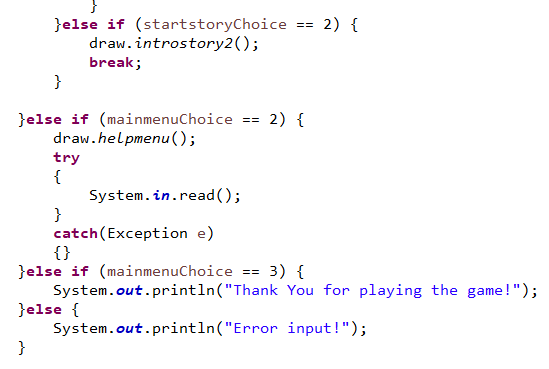
1. **main.py**

main.py is the program to run as it combines all of the code into one and allows the user to play the game.

Since the game is meant to be played on the console / windows CMD, it needs to be cleared first and make the console cleared. The code above makes that happen. With this each of the scene would be isolated from the previous.

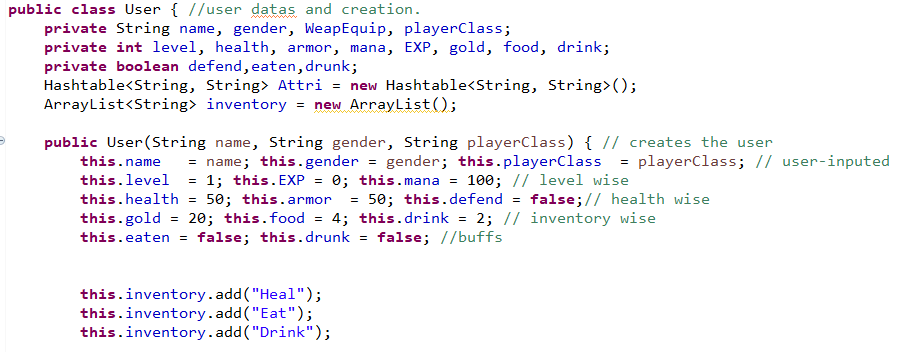
The next bit is the start of the code. To start the code would need to run in a constant loop, thus the use of “while true”. And then, the code calls from draw.java to print out the main menu. After that, it would request an input from the user so that it would be able to direct the user to either play the game, the help menu or to exit the program. if the user decides to play the game, the player would need to make a user object so that the data can be called and used. The player would mainly need to input a name, gender and their desired class. After making the player, the user is then directed to the starting story. After the user/player would read the starting story, they would be met with a problem that would either continue the story and allows the player to play the majority of the game or kills the player and exits the game to the main menu, forcing the user/player to restart.

If the user chooses 1, the user would be greeted with the 2nd half of the starting story and thus be able to continue the story. Inside the town, the user would be able to sleep to restore lost health and mana at a cost or to unlock upgrades with the little money, the user owns.

After that, another loop would start. This loop will continue until the user decides to exit to the main menu as can be seen if the user inputs “0”. This part of the code will direct the flow to either the upgrades shop, the inn, or to the dungeon.

These are the rest of the code that directs it to the other choices from the previously mentioned ones. \

1. **User.java**

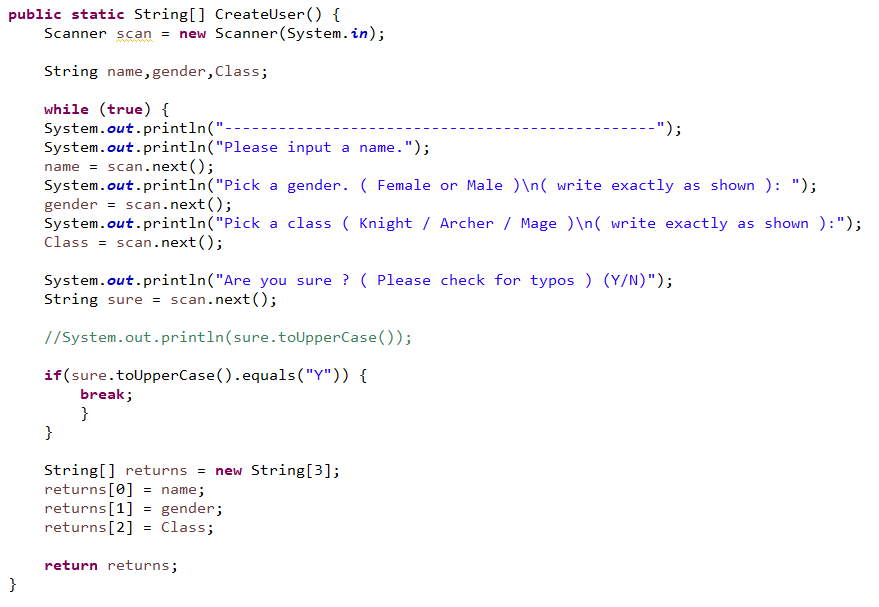
User.java is the class that handles all of the user object. It also stores the user’s data. This class also handles data retrieval and data editing.

The code above shows all of the data on the player, such as name, gender, and others. The main ones are the name, gender, and class as those are user preference. the majority of the player’s data are already configured as they are stock configurations.

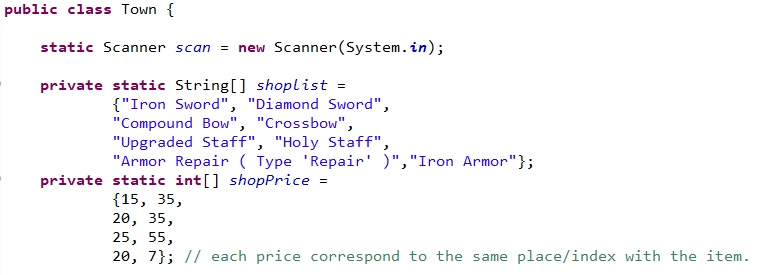
The code above shows the user’s class and their respective attributes and equipped weapon. These attributes will be used to change the chances of running from battle, or other aspects.

 The code above is responsible for checking the user’s level and levelling the user up when the user has reached a certain amount of Exp. By levelling up, the user’s attribute would increase and have better chances at running from battle due to higher agi and lck attribute.

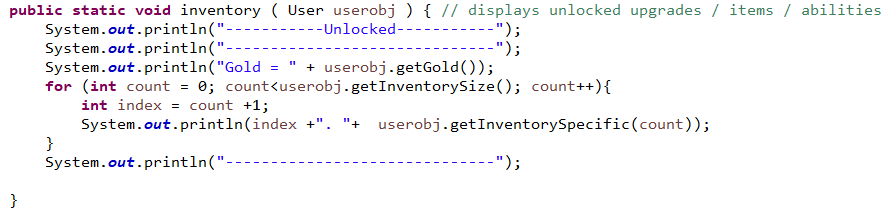
1. **userCreation.java**

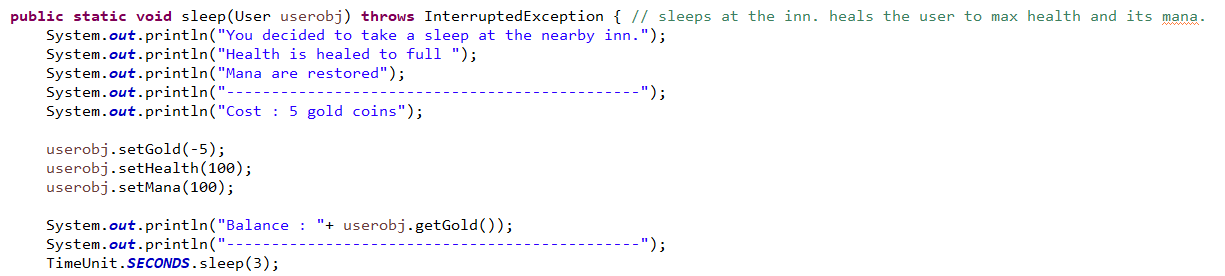
A simple class that handles the creation of the user. This is mainly used to get the user input for their preferred name, gender and class.

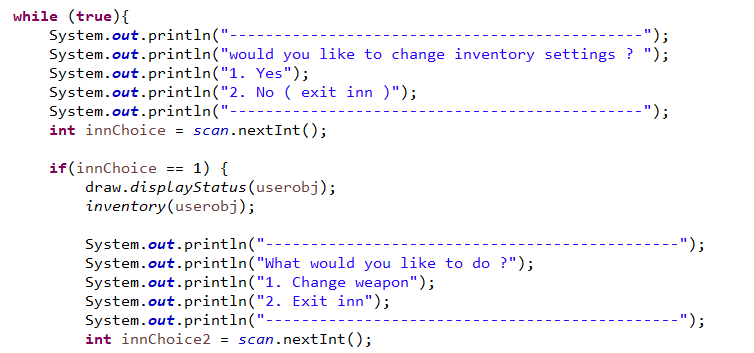
1. **Town,java**

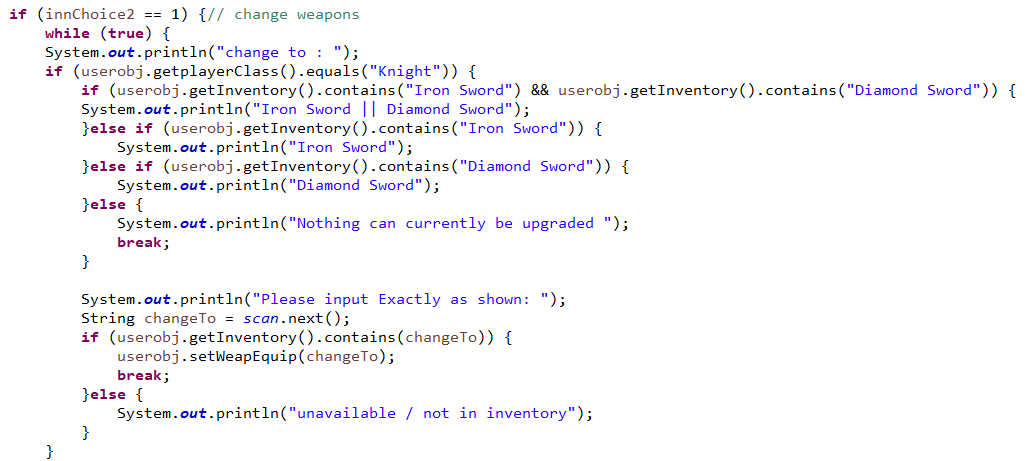
Town.java handles the upgrades unlocking and the inn. This class is one of the more important parts as it help the player get better gear and restore health.

The code above is for the shop’s items and its respective pricing. The scanner is also initiated so that the program can receive the user’s input.

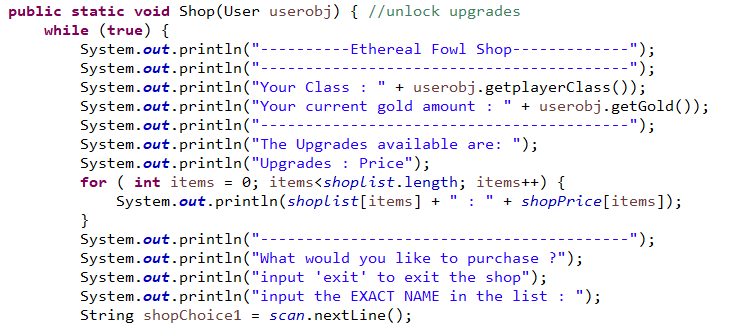
 The code above is responsible to print out what the user have unlocked from the shop and the amount of gold the user has.

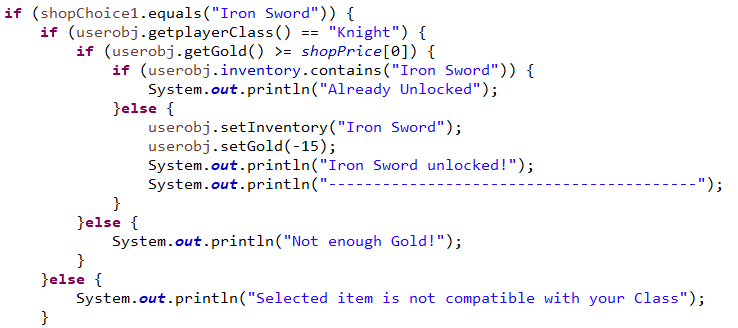
 The code above is the inn. This part of the function is to heal the user and to deduct gold from the user. It also restores mana that will be used in the future. It also prints out the remaining gold after the payment.



The code above is the main part of the inn. The inn would have another main part that is to change weapons of the player. If the player has unlocked upgrades in that department, then the player would be able to increase their damage when battling.

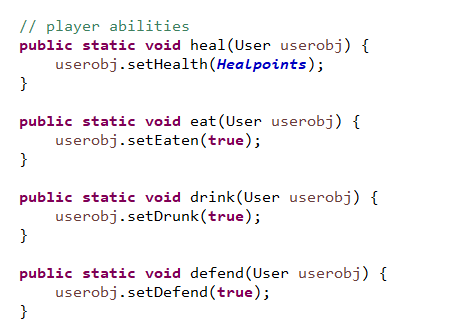
This part of the code is to print and set the equipped weapon that the player would choose. An error handler is present to make sure the player has the item and that the user did not input it incorrectly. This is also changed to fit with the player’s class. A slight change was done after the photo to make sure the input is correctly registered by the program.

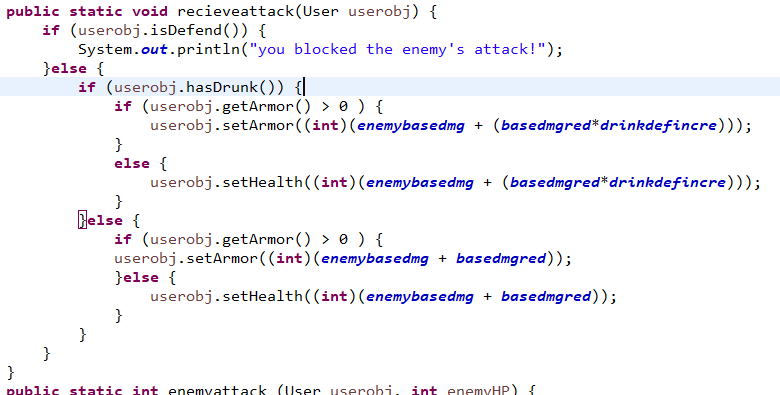


The code above is responsible for the player to unlock upgrades. This part prints out the text and tells what upgrades the user is able to purchase.

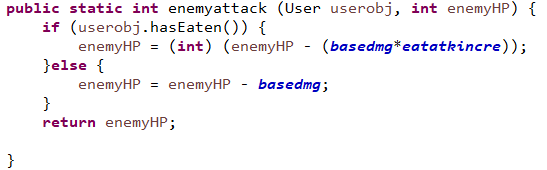
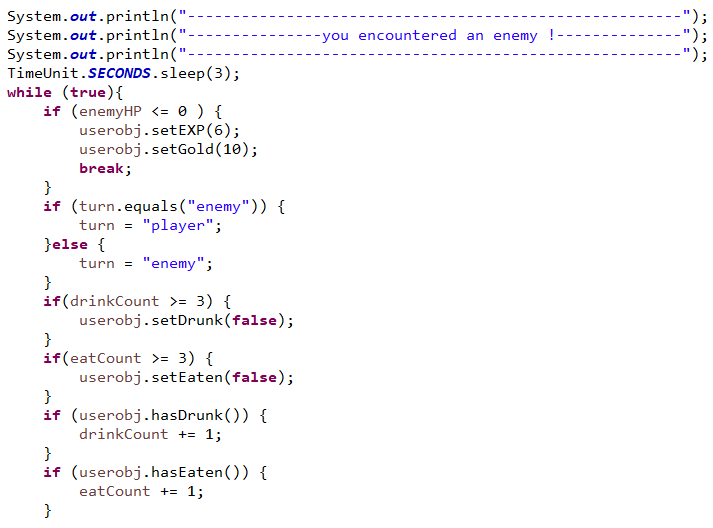
If the user inputs correctly, the program would check the player’s class as each class cannot use another class’ weapon. It would then check whether you have enough gold to purchase it. If you do then, the program checks whether you already unlock it. If the player passes all of these checks, then the user will be able to unlock it and later on equip it in the inn. This is the same for all of the weapons / upgrades sold.

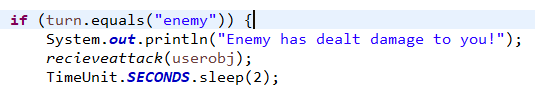
1. **Battles.java & Player.java & Enemy.java**

Battles.java is the class that handles the dungeon and any battles that will take place. This class implements the Player and Enemy Interface as they contain constants that are used for battling.

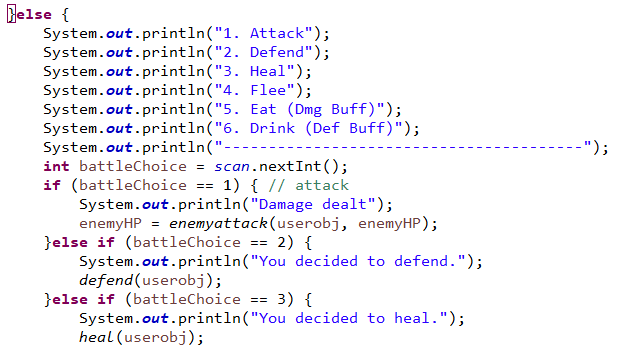
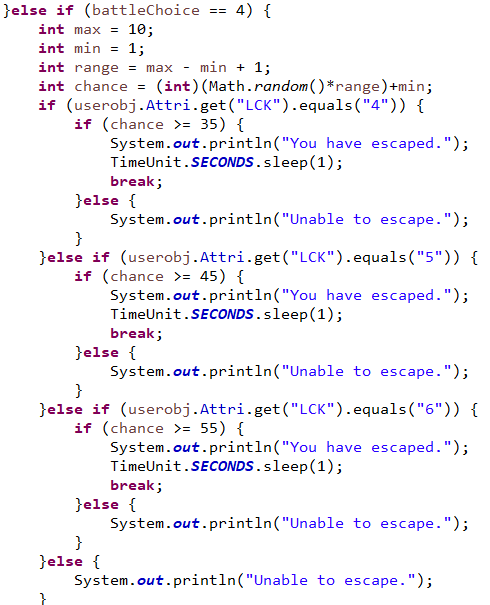
The code above shows the basic abilities that the player can do such as eat, drink, defend, and heal.

The code above is for when the player receive attacks. The damage would be multiplied from the alpha damage from the interface. The final damage is calculated from if the user drinks, which will increase the player’s defence, and the base damage reduction from the Armor that the player is automatically equipped. Due to time constraints, there is a need of future implementations such as, new types of enemies, the implementation of the usage of different weapons and the new enemy type base damage reduction.

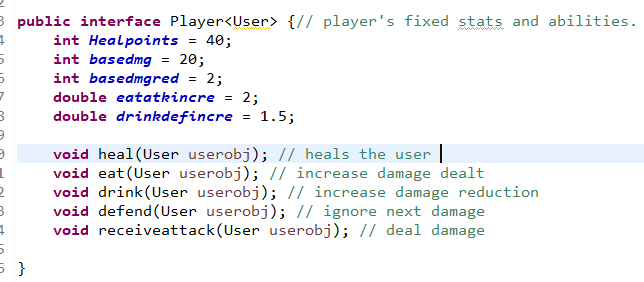
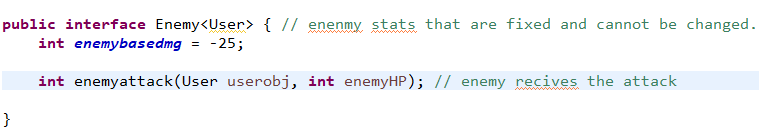
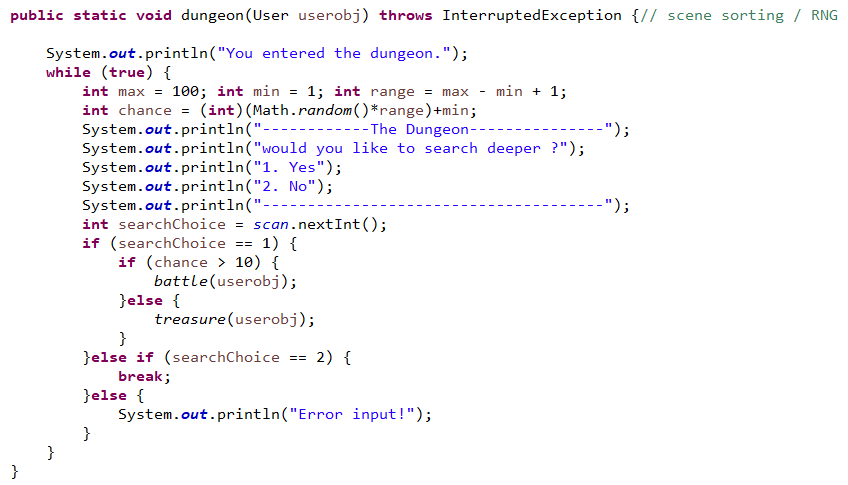
The code above is when the player deals damage. Implementation of the weapon choice is not present as time constraints hindered the implementation. But the code is simple as the alpha damage is set from the interface multiplied with whether the player eats or not.

To make it all work, the program would check a couple of things beforehand such as, the eat and drink turn count to disable it and the turn. It also checks the enemy HP. If the HP is below or equal to 0, then the enemy has been defeated.

The enemy’s turn is made simple, that is to only deal damage. Future implementations would the randomize what it will do, as the enemy should be able to heal and defend similarly to the player.

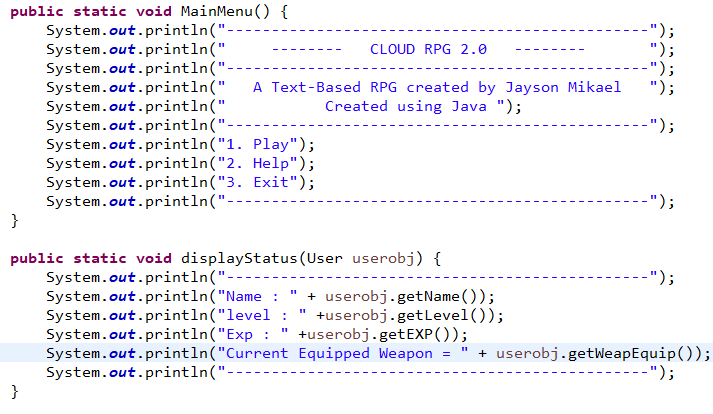
for the player, however, has a lot more choices. The code above shows what the player is able to do and the premade functions that it would call.

In order for the player to flee from battle, however, several factors would contribute. Such as the player’s luck attribute. This would make use of the attributes from the class that the player chose.

**** This part of the code is to sort the scene to where the player would go to. The player is randomized to go to battle 90% of the time and skip the battle to gain rewards 10% of the time. This helps ensure that battles aren’t the only thing that the player would receive.

Both player and enemy interfaces are self-explanatory as they contain what the player is able to do on a basic level.

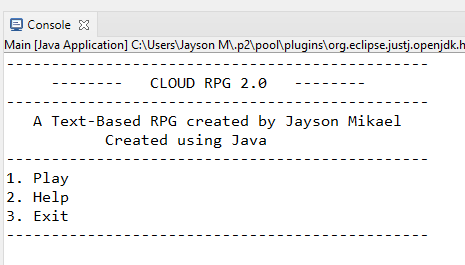
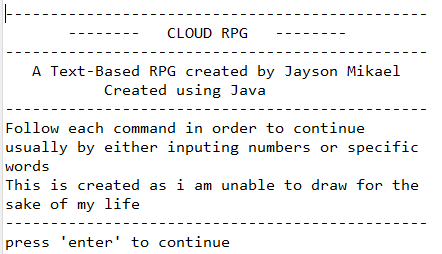
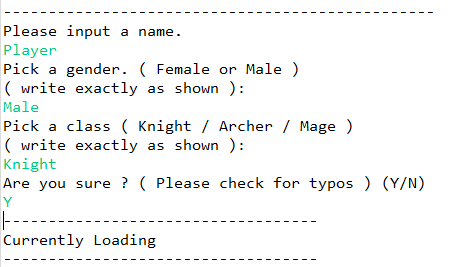
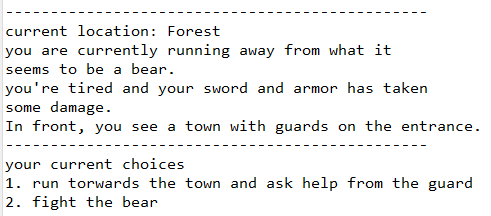
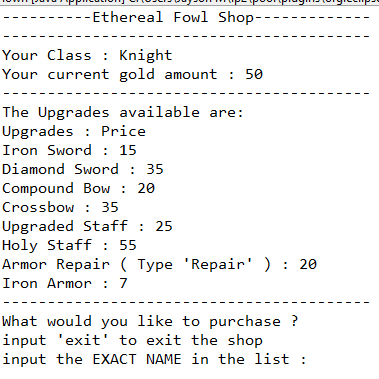
1. Draw.java

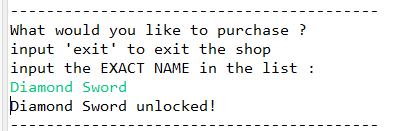
Draw.java is simple as it contains mostly print functions to print out the text and give visuals to the player.

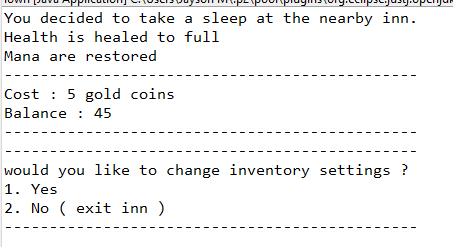
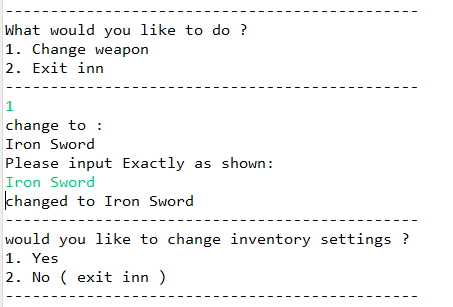
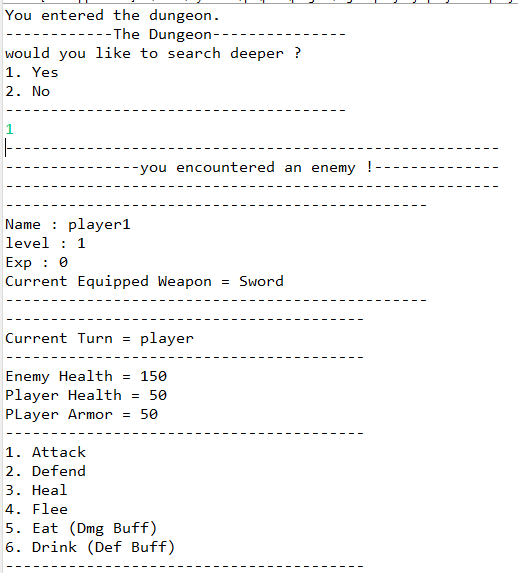
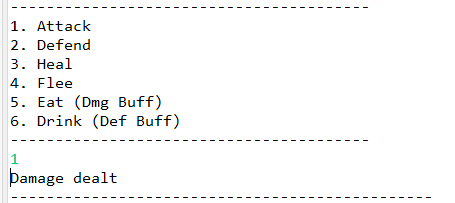
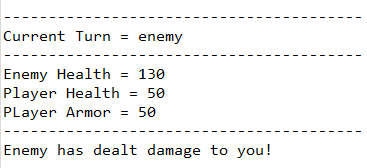
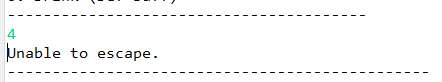
This is two of the many functions inside the class.

1. **Screenshots / Evidence of working program**

These are taken on the eclipse console as I was not able to make it run from the windows cmd, I wasn’t able as I am unable to find the correct path to the file.

1. Main menu
2. Help Menu
3. User Creation
4. Intro Story
5. The Shop
6. Purchasing



1. The Inn
2. Change weapons
3. Battles
4. Attacking
5. Enemy Turn
6. Trying to escape
7. **References**

* Edpresso Team. (n.d.). *How to create a dictionary in Java*. Educative: Interactive Courses for Software Developers. Retrieved June 21, 2021, from <https://www.educative.io/edpresso/how-to-create-a-dictionary-in-java>
* *How to Return an Array in Java - Javatpoint*. (n.d.). Www.Javatpoint.Com. Retrieved June 21, 2021, from <https://www.javatpoint.com/how-to-return-an-array-in-java>
* *Java ArrayList int, Integer Examples - Dot Net Perls*. (n.d.). Dotnetperls. Retrieved June 21, 2021, from <https://www.dotnetperls.com/arraylist-integer-java>
* *Java Interface*. (n.d.). W3schools. Retrieved June 21, 2021, from <https://www.w3schools.com/java/java_interface.asp>
* *Java String toLowerCase() Method*. (n.d.). W3schools. Retrieved June 21, 2021, from https://www.w3schools.com/java/ref\_string\_tolowercase.asp