2AA4 - Assignment 3

Please read this document very carefully. Follow instructions exactly. If you have any questions please post them to MS Teams or ask during office hours.

This assignment is due April 8th, by 11:59pm

I have created an Assignment 3 channel in Teams. If you have questions about the assignment, please post them there. Thank you.

NOTE: You may work and submit this assignment with a partner if you wish. More information on this in the Submitting and Grading section.

Purpose

The primary objective of this assignment is to assess your ability refactor code via design principles and patterns, and justify those decisions. The goal of this assignment is not to assess your ability to code, but rather your ability to design.

Overview

The following files are associated with this assignment.

- `Armour.java`
- `CAI.java`
- `EncryptionProtocols.java`

You are responsible for submitting three files:

1. A3.pdf, .txt, .docx (as long as we can read it)
2. `Q1Code.zip`
3. `Q2Code.zip`

See below for details on what you are responsible for completing.

Your Tasks

Look at the following descriptions of each of the two problems below, as well as the corresponding code. For each question your A3.pdf document must contain the following:

1. A UML class diagram of you refactored code
2. A list of design patterns you applied and why you applied them
3. A list of design principles you feel your design enforces, and a justification as to why it does
4. An explanation of how your design is intended to be used in the overall application.

Question 1

Take a look at the `Armour.java` file. This is intended to be used in an Action RPG (think Diablo, Path of Exile, Torchlight etc.). Traditionally, in these games, the player will kill enemies and enemies have a chance to drop *loot*. For our purposes, we will assume enemies only drop armour.

As the code stands right now the Armour class is functional, but not necessarily maintainable. You need to consider some potential changes that could occur. For example,

- New modifications which could be added. Right now there are bonuses to Strength, Intelligence, and Dexterity, but many more could potentially be added later. For example, fire resistance, experience gain, health, mana, etc.
- New sorts of armour could be added (shields, pauldrons, rings, amulets, belts, pants).
- New rarities could be added (mythic, legendary, unique, etc).
- You may want to control how Armour is generated in some way. That is, you would expect that armour dropped from a boss would in general be of a higher quality than armour dropped by a normal enemy.

You have quite a bit of control over how you refactor the code, however, the `getDescription()` method should still function for any variable of type Armour found potentially elseware in the application.

Question 2

You have been hired to clean up the security at a secret government base that the British spy Jim Stock is rumoured to be infiltrating soon; their design is to have their Central Artificial Intelligence (CAI) send out new encryption protocols to security sites each day to confuse would-be infiltrators. You do NOT have to implement any encryption protocols. Instead try to determine a common interface across them.

The security sites currently make use of Ouroboros and Cerebus key encryptions, using either one based on the security level of the message. Moreover, the CAI set each of its site protocols individually. This is not wise as it would prefer that all its sites use the same protocol. Note, in the CAI class right now there is a multitude of encrypt/decrypt methods for each site – this is not maintainable.

Refactor the CAI and EncryptionProtocols classes to account for the following functionalities and changes which may occur.

- New sites could be added to the network. Moreover, a site could also be removed from the network.
- The CAI may wish to change the encryption protocol and notify all remote sites of the change so they can update their security. Moreover, they should have a roster of protocols they randomly select from.
- New encryption protocols may be designed and potentially added to the roster.

Submitting and Grading

This assignment will be submitted electronically via Avenue. As stated at the beginning of this document, you may work on and submit the assignment with a partner. If you choose to work with a partner, only one of you will submit the assignment. Furthermore, on the front page of your A3 document you must clearly indicate who your partner is as well as all identifying information (macid and your names as they appear on Avenue). The majority of the grades will be given to your A3 document. You code will not be auto-graded as in A1 and A2, but the TA will look at your code to verify your claims, as well as potentially run it to verify correctness. A rough breakdown is given below.

- Question 1: 50%
  - 10% UML diagram
  - 20% Applying correct/appropriate design patterns
  - 10% Design principle justification
  - 10% Explanation of how your code works, and your overall code in general
- Question 2: 50%
  - 10% UML diagram
  - 20% Applying correct/appropriate design patterns
  - 10% Design principle justification
  - 10% Explanation of how your code works, and your overall code in general

Academic Dishonesty Disclaimer

All of the work you submit must be done by you, and your work must not be submitted by someone else. Plagiarism is academic fraud and is taken very seriously. The department uses software that compares programs for evidence of similar code.

Please don't copy. The TAs and I want you to succeed and are here to help. Here are a couple of general guidelines to help you avoid plagiarism:

Never look at another assignment solution, whether it is on paper or on the computer screen. Never show another student your assignment solution. This applies to all drafts of a solution and to incomplete solutions. If you find code on the web that solves part or all of an assignment, do not use or submit any part of it! A large percentage of the academic offenses in involve students who have never met, and who just happened to find the same solution online. If you find a solution, someone else will too.