Homework 1: (Java)
RPG Sub-System API

Assignment Goal:

An RPG game is derived from a complex set of independent subsystems: Items, Monsters, Player Classes/Skills, Magic Spells, Quests, Combat rules. For this project, identify and select one such RPG sub-system, and fully design it via UML and implement it as its own independent java package. Your API must be completely functional with internal behaviors or interactions consistent with managing and mutating state and not just console logging. Your API must also include JavaDoc files and JUnit testing.

You may use *Lecture Code* as a reference for using Interfaces and Classes effectively and authoring JavaDoc documents and JUnit tests. This RPG Subsystem should contain multiple interfaces, abstract and concrete classes. The system should be designed for reusability, flexibility, and robustness. This RPG Subsystem will be used as the basis for future homework projects but is not intended to be a game in and of itself.

Learning Objective:

You must perform Object Oriented decision-making for designing your project. You must effectively use Interfaces to design a compelling and deep Item system that is reusable, flexible, and robust. For some, this may be the first time “building” something on your own in Java.

- OOP Inheritance & Polymorphism
- Effective use of Interfaces, Abstract classes, & Concrete classes
- Design by Contract, Programming to the Interface
- JavaDocs to generate an API
- JUnit to Unit Test your Item System

Implementation Advice:

Focus on defining interfaces that define behaviors that items may implement to qualify as a member of that defined trait. Here's some tips to make it easier:

- Start by planning out all your interfaces that will represent micro-API.
- Each interface contains only those methods that represent that trait. For example, the interface Flammable might have one method burn().
- Classes should implement multiple interfaces
- Use both abstract classes and concrete classes in your design.
- Be creative & experimental!

Resources:

You may use **Lecture code or Lab code** to start!

Showcases & Demos:

https://codepen.io/MadLittleMods/pen/vYdrWo

Why Write Your Own API?

1. **OOP Design Practice:** Designing complex hierarchical systems forces you to create many different interfaces and classes with a variety of related and divergent elements.

2. **Portfolio project:** Employers and recruiters commonly prefer candidates who have project portfolios to demonstrate their technical capabilities. Passion and creative projects are often good additions.

Item System Inspirations

- **Fantasy Item System**
- **Cooking Item System**
- **Monster System**
- **Hero Class/Skill System**
- **Magic Spell System**
- **Quest System**
- **Combat System**
- ***Hint*: *A crafting or mini-RPG may be a future assignment that extends this one.***

JUnit Testing

Design a Test Plan for your Item System

You should create a Test Fixture for each class and a Test Case for each class. Make sure your Test plan includes checking routine, challenging, and edge-case instances.

JavaDocs

Full API documentation should be included with your Item System

Your Item System is designed to be used by external client code. The API is your contract as an implementer with clients. Ensure you use JavaDoc documentation to produce the necessary API file.

Grading Rubric

```
Part 1: [  OOP  ]  interfaces, abstractions, compositions      [  50%  ]
Part 2: [  API  ]  documentation, comments                     [  20%  ]
Part 3: [ JUnit ]  test plan, test coverage                    [  30%  ]
Part 5: [ Bonus ]  Outstanding Submission;                     [ 0-10% ]
```

Submission:

Tools

Java, IntelliJ, JavaDocs, JUnit 5

Submission:

Submit your source code to gitlab or github. Publish your HTML file into a doc folder of your project. To receive a grade, you must schedule a zoom audit with me where you walk-through your codebase.