CSE 1325: Object-Oriented Programming
University of Texas at Arlington
Summer 2023
Alex Dillhoff

Project Phase I

Description

In this course, you will create a table top RPG combat simulator based on a simple set of rules described below. This application should have support for 2 players. Each player can create their own custom character before playing. Players can fight each other with enough randomness and optional choices to make it interesting each time.
This first phase will require implementing concepts involving classes, objects, logging, and UML diagram creation.

Features

This application will include the following features:

- **Player Creation** - Players can create their own character with a name, some basic stats, and a weapon.
- **Combat System** - A basic combat system based on a simplified version of a popular table top game will be implemented.
- **File I/O** - The program will interface with files both for logging and for reading external data.

UML Diagrams

All classes created for this project must have an accompanying UML diagram. Please submit a published format of the diagrams (image or PDF) that does not require an additional program to view.

Players

Player characters have a **name**, **creation date**, and the following stats:

- Hit Points (HP)
- Armor Class (AC)
- Strength (STR)
- Dexterity (DEX)
- Constitution (CON)

Character Stats

Character stats can be assigned a value in the range $[0, 10]$. Associated with each stat is a modifier value which can augment certain rolls and abilities. A stat value of 5 has NO modifier value (+0). For each point below 5, subtract 1 from the modifier. For example, a value of 3 has a modifier of (-2). For each point above 5, add 1 to the modifier. For example, a value of 9 has a modifier of (+4).

When creating a character, a player has **10 points** to put into their stats however they choose. Their choice of stats will affect how much bonus **HP** and **AC** they have. Each player starts with **20 HP** and **10 AC** by default. After selecting their stats, the player's total HP is calculated as

$$\text{HP} = 20 + \text{CON modifier}$$

and their total AC is calculated as

$$\text{AC} = 10 + \text{DEX modifier}$$

Weapons

A player may select out of the following list of weapons. This list will be provided for you to use with your program in CSV format. You must create a `Weapon class` to represent them in your code.

Each weapon has a name, a damage statistic, and a hit bonus. Typically, weapons that do more damage will be slower and have a lower chance of hitting.

Weapons		
Name	Damage	Bonus To-Hit
Greataxe	1d12	0
Longsword	1d10	1
Warhammer	1d8	2
Shortsword	1d6	3
Dagger	1d4	4

The Map

By the end of this project, the application will utilize a graphical framework to display the characters and the map that they move on. For this phase, the map is a simple **25 by 25** grid. Each character will have an instance field which keeps track of its own position relative to the map they are **in**.

Combat System

When initiating combat, the two players will roll a d20 and add their dexterity modifier. The player with the higher number will go first. If the rolls are equal, they must re-roll until the tie is broken.

Once the initiatives are set, the first round of combat begins. A round of combat consists of all eligible players taking at most 1 action and moving up to 5 grid units.

Actions

In Phase I, the actions will be limited. A player can either attack or attempt to disarm the other player.

Attacking

To attack, the player must first roll to see if they strike. They roll a d20 and add their DEX modifier and weapon bonus. This roll is compared to the target's AC value. If the attack roll is equal to or higher than the target's AC, the attack is successful and the attacker may roll their damage dice determined by their weapon.

Each weapon has some damage value, such as d6, which is rolled whenever an attack is made. The player's STR modifier is added to this roll when calculating the amount of damage done to the target.

Disarming

Instead of attacking, a player can attempt to disarm the other player. If the player chooses this option, both the target and the attacker must roll a d20 and add their STR modifier. If the attacker's roll is STRICTLY higher than the target's roll, then the target is disarmed for 2 rounds of combat. During this time, the target cannot perform any attack or disarm actions. They are still able to move.

Movement

In addition to their action, each player may move up to 5 units during their turn. They may not move into the same space as another player nor may they move anywhere outside of the map area.

The Application

All of these components will be utilized as part of an application, starting with the main menu:

```
1. Start Game
2. Create Characters
3. Exit
```

Creating Characters

When creating a character, start by prompting for the name. A sub-menu should then prompt the user to either enter their stats manually or randomly assign them.

```
Enter Character Name: Player 1
```

```
1. Manual Stats
2. Random Stats
Manual or Random Stats? 1
```

If the user picks manually, the following menu will loop until all stats are assigned.

```
STR: 0
DEX: 0
CON: 0
Remaining: 10

1. Add STR
2. Add DEX
3. Add CON
4. Reset
5. Finish
> 1
```

Weapon Selection

After selecting their stats, the player will select a weapon.

```
1. Greataxe, Damage: 1d12, Bonut To-Hit: 0
...
Select weapon: 1
```

After all selections are made, the user should be shown a Character summary based on their choices. They can choose to either confirm or start over. Once they confirm, the second character should be created.

Once both characters are created, the main menu is shown again.

Grading Requirements

This section describes the requirements, based on the features above, that will be considered when grading projects.

Player Class

- Should have at least 2 constructors.
- Includes instance fields for stats that can only be modified through setters.
- **Creation Date** should be `private`.
- Includes methods which calculate modifier values based on internal instance fields.
- Includes an instance field for the `Weapon class`.
- Must have appropriate accessors and mutators for fields needed by other classes.

Weapon Class

- Should have at least 2 constructors.
- The instance fields should reflect the properties listed in the table above.
- Must have appropriate accessors and mutators for fields needed by other classes.

The Map

- Defined as a **25 by 25** grid.
- Can query to see if a particular space is occupied.

Combat System

- Implement initialization.
- Implement Actions (attack and disarm).
- Should print to the terminal each action and movement that occurred.

Application

- Implement menus and program control flow based on the descriptions above.
- Should be implemented as a separate file.

Code

- Code should be formatted consistently.
- The project should compile without warnings or errors.

UML Diagrams

- Each class in your project should have a corresponding class diagram.
- The main diagram should show the relationship between all classes in your project.

Submitting

Submit all class files, UML diagrams, and other code files on Canvas as a compressed zip file named `LASTNAME_ID_P1.zip`, where `LASTNAME` is your last name and `ID` is your student ID starting with 1xxx.