ThunderShade - COMP1010 Group Project

// Members - Nguyen Nguyen Chuong Le, Caleb Henry, Mrinmoy Saha, Mark Cai

Summary of Our Game:

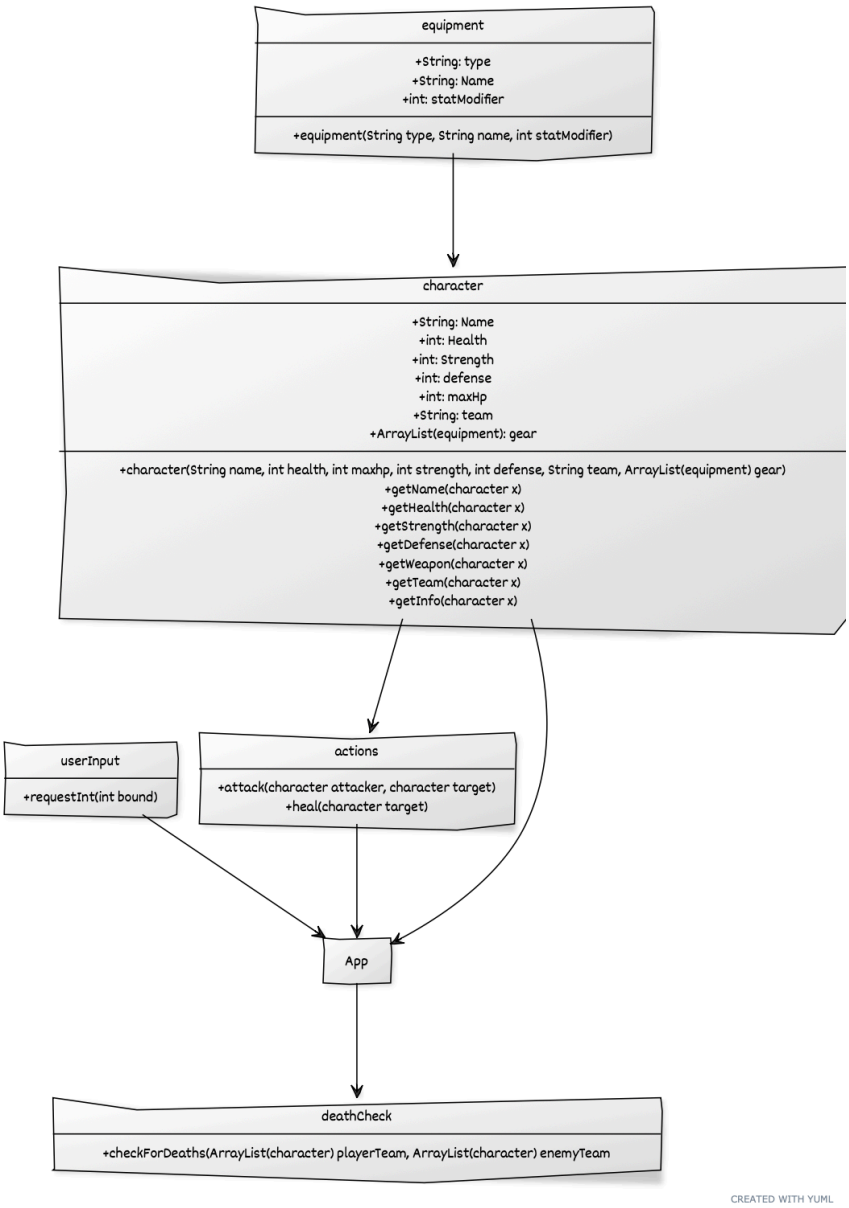
The game initializes characters and weapons.
A player selects a character to control, with the rest assigned to the enemy team.
The player accepts or declines the battle.
If accepted, the game alternates turns between player-controlled characters and enemies:
Players can choose to attack or heal.
Enemies randomly choose to attack or heal based on their health status.
After each turn, the deathCheck method checks if a team is entirely defeated to determine a win or loss.

GITHUB REPO

https://github.com/clbmacq/COMP1010-2024-Assignment

READ (README.MD) BEFORE STARTING GAME

ERD PROGRAM – Jayden



What our program solves

This application is a turn-based RPG (role-playing game) designed to entertain players and provide an engaging solution to boredom by offering a fun and strategic gameplay experience. Players build a team of characters, each with unique attributes, and face off against an opposing enemy team in a series of rounds. Players can choose to attack or heal in each turn, manage health points, and leverage their characters' strengths and equipment. The game incorporates strategic decision-making and chance, as players must anticipate and adapt to enemy actions and the outcomes of random events (such as whether or not an attack will connect with the enemy, how much that attack will do, and how many hit points are healed when a character chooses to "Heal"

DELEGATION OVERVIEW

- **CODE DEVELOPMENT** - All four group members, however class design, unit testing, revision and final touches by Caleb Henry
- **README FILE and "What our program solves"** by Mark Cai
- **ER Diagram** by Jayden
- **Structure of the program** - by Mrinmoy Saha and Caleb Henry

Work Distribution Estimate

- **Caleb - 30%**
- **Mark - 26%**
- **Chuong Le (Jayden) - 24%**
- **Mrinmoy Saha - 20%**

Structure of the Program

***Classes*:**

App - The main class where the game starts, teams are formed and the battle progresses through rounds.

GameCharacter - Represents a player or enemy, containing attributes like health, strength, defense and weapons.

Actions - Contains methods for attacking and healing during the battle.

Equipment- Represents the weapons that each character can use in the game, with attributes like type, name, and statModifier.

DeathCheck- Monitors if any team has been completely defeated.

***UserInput* -** this class's method is called whenever the game requires the user to input an int

***Game flow*:**

1. ***App & GameCharacter***: The **App** class creates instances of the **GameCharacter** class, representing the characters in the game. The **GameCharacter** objects are initialized with their name, health, strength, defense, team, and equipped gear. The **App** class uses the **GameCharacter** objects to manage player and enemy teams and to perform actions like attacking or healing.

2. ***App &* Equipment** : The **Equipment** class represents the weapons used by characters in the game. The **App** class creates instances of the **Equipment** class and assigns these weapons to the character objects.

3. ***App &* Actions** : The **Actions** class contains the methods that define core gameplay mechanics like attack and heal. The **App** class calls the methods in the **Actions** class during gameplay when a player or enemy takes an action, such as attacking or healing.

4. ***App &* DeathCheck :** The **DeathCheck** class is responsible for determining if any team has been fully defeated. After each round of gameplay, the **App** class calls the **checkForDeaths** method to see if the game should end because one team has been eliminated.

5. ***App & userInput*:** userInput's method is called whenever the program wants the user to input an int. This is done:
    a. When the user is selecting a character
    b. When the user is choosing what character to make an action with
    c. When the user is choosing whether to attack or heal with that character