COMP1010 MAJOR ASSIGNMENT
TOPIC 1: MASTERS OF MQ: TURN-BASED RPG COMBAT

Problem Statement

Design and implement a turn-based RPG game that contains two teams where players can choose characters with unique stats, including health, strength, and defence. The two teams of characters battle each other until one team is defeated by reducing all opponent characters to zero health points. Each character has a choice between attacking, defending and dodging, with combat outcomes influenced by a dice-based randomiser.

How to run the program

1. Download the “COMP1010-Major-Assessment” folder.
2. Extract the folder.
3. Open Visual Studio Code.
4. Click on **File** > **Open Folder**.
5. Select the “COMP1010-Major-Assessment” folder and click **Select Folder**.
6. In the top left corner, open **Main.java**.
7. On line 11 of the code file, click **Run** to start the program.
8. In the terminal, press **Enter** to initiate the game.

Gameplay Instructions

- Press **1** to start the game.
- Press **2** to view the tutorial.
- Choose your character’s race by pressing **1, 2, 3,** or **4**.
- In-game actions:
    - Press **1** to "Attack" (randomly targets an opponent).
    - Press **2** to "Defend" (attempts to block incoming damage).
    - Press **3** to "Dodge" (attempts to avoid attacks).

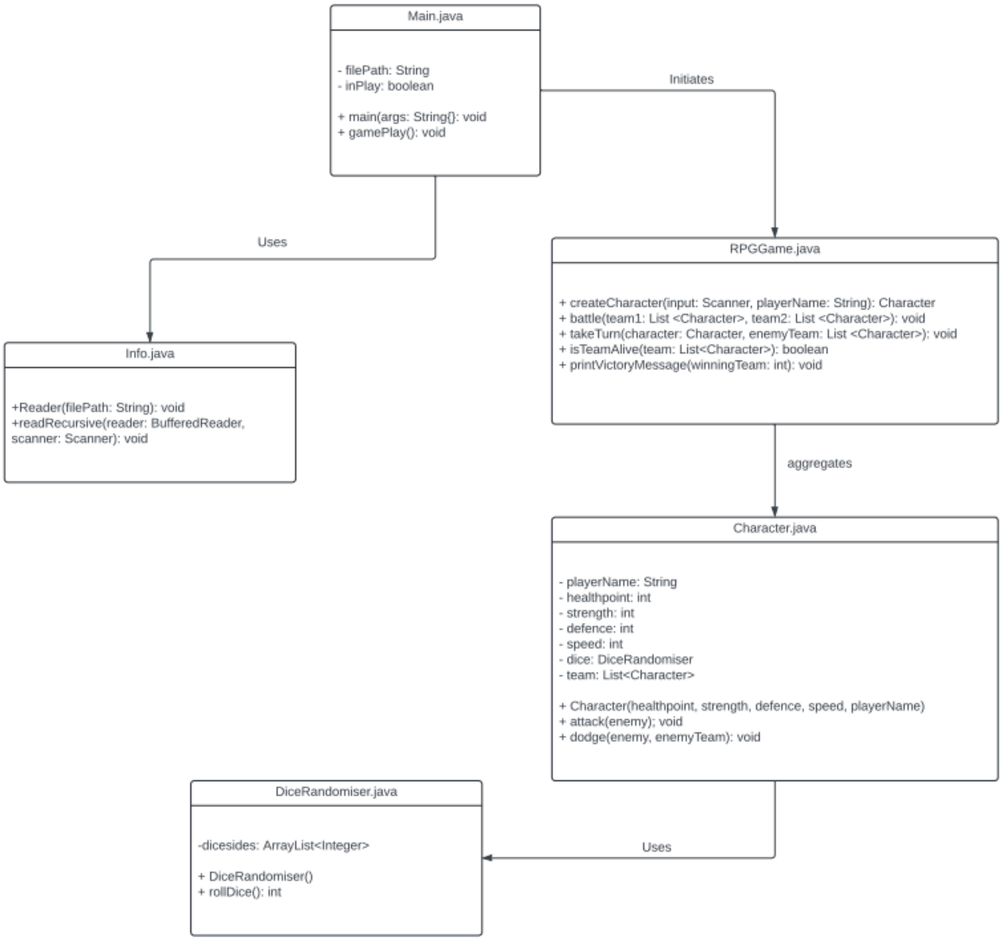
Design Process

DiceRandomiser.java	This is a dice randomiser class made using an ArrayList. Essentially, it is used to roll a 6-sided dice which will be used to randomise the attack damage within the character class.
Character.java	This character class adds randomised damage and combat actions such as attack, defence and dodge. The attack method inflicts a random amount of damage until the enemy's health is zero. The defence method blocks attacks, thus no damage is done to the character. The dodge method has a chance of avoiding all incoming attacks.
RPGGame.java	The RPGGame class controls the game flow. It allows the players to choose characters, and alternate turns between the two teams until one is defeated, it allows the characters to use the combat actions, they can choose between attack, defend or dodge each turn. It also determines and displays the winning team once the battle has ended.
Info.java	The info class reads the content from the info.txt that the main class provided. It allows the player to interact with the information by prompting them to press enter to go view the next content recursively.
Main.java	The main class launches the game. It writes tutorial information to info.txt, it prompts the players to choose between the tutorial or the game, it is responsible for the gameplay as it creates two teams in which players select characters and begin the battle. It ends the game once a team has been defeated.
Main.java	Reformatting output text to be more appealing/easier to read.
RPGGame.java	Implementing a dodging mechanic was more complicated than anticipated. Applying a randomiser to the dodging mechanic proved slightly difficult for certain values to read properly. It was completed with a simple change of initialised values.

Task allocation

Name	Task	Overall %
Jonathan Nehme 47870583	Created DiceRandomiser.java, implemented Character.java, Design process and Problem Statement	25%
Lachlan Zou 48497703	File input/output created Main.java, Info.java (recursively) and Info.txt (the tutorial). Modified dodging mechanic.	25%
Tontapuk Charoenpol 47331909	Created Client.java, Character.java, attack method, and the defence method (methods inside Character.java). Edited and polished System.out.print terminal. "How to run the program" section	25%
Matthew Hadi 46007628	Created the RPGGame.java and edited Character.java, Main.java and DiceRandomiser.java. Created the UML diagram.	25%

UML Diagrams


Main.java
- filePath: String
- inPlay: boolean
+ main(args: String{}: void
+ gamePlay(): void
Initiates
Uses
RPGGame.java
+ createCharacter(input: Scanner, playerName: String): Character
+ battle(team1: List <Character>, team2: List <Character>): void
+ takeTurn(character: Character, enemyTeam: List <Character>): void
+ isTeamAlive(team: List<Character>): boolean
+ printVictoryMessage(winningTeam: int): void
Info.java
+Reader(filePath: String): void
+readRecursive(reader: BufferedReader,
scanner: Scanner): void
aggregates
Character.java
- playerName: String
- healthpoint: int
- strength: int
- defence: int
- speed: int
- dice: DiceRandomiser
- team: List<Character>
+ Character(healthpoint, strength, defence, speed, playerName)
+ attack(enemy); void
+ dodge(enemy, enemyTeam): void
DiceRandomiser.java
-dicesides: ArrayList<Integer>
+ DiceRandomiser()
+ rollDice(): int
Uses