** Noted that Access Modifier Notations can be listed below*

+ (public)

(protected)

- (private)

Underline (static)

Italic (abstract)

Set-Up Instruction

- Don't forget to set your Eclipse workspace and working set.
- **You must submit the JAR file, exported (with source code), from your Eclipse project.**
- **You must check your JAR file to make sure all the source files (.java files) are present. It can be opened with file compression programs such as 7-zip or Winrar.**
- **Failure to export properly will result in your work not getting marked.**

Inheritance

1. Objective

1) Be able to understand and utilize a concept of inheritance and polymorphism in Object-Oriented Programming (OOP).
2) You ONLY have to complete toStudent package and test it with Junit classes in test package, running the game is optional

2. Instruction

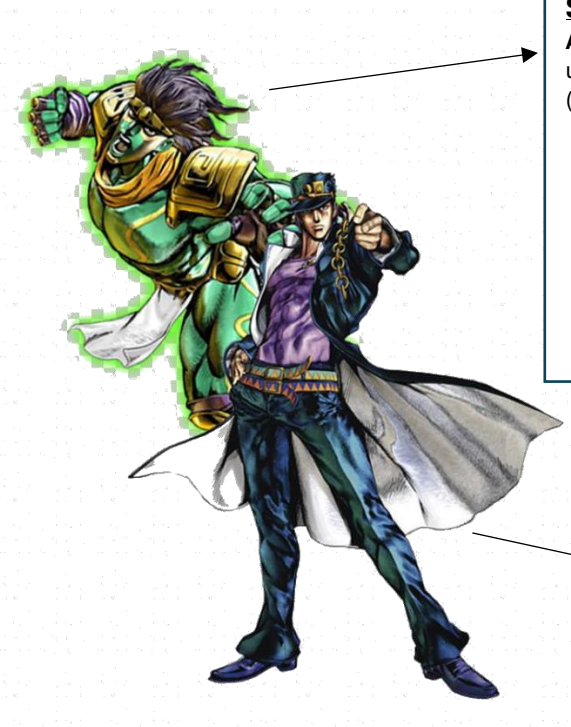
1) Create Gradle Project.
2) Copy all folders to your Gradle source code folder.
3) You are to implement the following classes (detail for each class is given in section 3 and 4). Make UML for them too, or your score will be capped to 80%.

   a) **Character** (package forStudent)
   b) **StandUser** (package forStudent)
   c) **Stand** (package forStudent)

4) **JUnit for testing is in package test.**
5) **To submit:**
   - **Export your project to a JAR file, with source code.**
   - **Submit the JAR file on MyCourseville.**

3. Problem Statement: StandUser Bizarre Fighter

StandUser Bizarre Fighter is a JAVA application RPG game (inspired by Jojo bizarre Adventure series). It includes a single player mode for a player to choose his/her favorite Stand User and Stand to fight in turn-based combat against computer AI. A player can also add more character into the game. Currently, the application is completed except for classes representing Stand User and Stand.



Stand

A Support Character that can be activated in battle to power up Stand User
(Has quote to say when selected & activated)

- **Strength**, how much strength added to the stand user when stand is activated
- **Defense**, how much defense added to the stand user when this stand is activated
- **Charge Turn**, How much charge the stand need for Stand User to activate it

(You don't need to know how Charge Turn works, you only have to use it in Stand constructor)

Stand User

A Primary Character used in battle
(Has quote to say when selected)

- **Hp** (Hit point), how much damage this character can take before losing the battle
- **Strength**, how much damage this character can inflict to an opponent
- **Defense**, how much damage can be reduced when taking damage

4. Implementation Detail

The class package is summarized below.

*** In the following class description, only details of IMPORTANT fields and methods are given. ***

4.1 Package forStudent

4.1.1 Class Character (Already Given)

This is the base class which represents all characters in the game. Both StandUser and Stand extend this class.

Field

Name	Description
- String name	Contains the character name.
- String quote	Contains the quote character will say when they are selected.
- int strength	Contains the character attack power.
- int defense	Contains the character defend power.

Constructor

Name	Description
+ Character (String name, String quote, int strength, int defense)	Assigning name, quote, strength, and defense. name and quote can be empty, but **if strength or defense is negative, it must be set to 0.**

Method

Name	Description
+ String getName()	name getter method.
+ String getQuote()	quote getter method.
+ int getStrength()	strength getter method.
+ int getDefense()	defense getter method.

4.1.2 Class StandUser (Partially Given)

This class represents a main playable character. It has 8 fields total.

(4 of them are inherited from Character)

The Additional Fields

Name	Description
- int maxHp	**/* FILL CODE */** Contains the StandUser maximum hit point.
- int currentHp	**/* FILL CODE */** Contains the StandUser current hit point during gameplay.
- boolean isGuard	Contains the StandUser properties telling if the character is guarding or not.
- Stand stand	Contains the StandUser assigned stand.

Constructor

Name	Description
+ StandUser (String name, int hp, String quote, int strength,int defense)	/* FILL CODE */ Assigning name, hp, quote, strength, and defense by calling parent class's constructor. The StandUser's **hp** value is assigned to both **maxHp** and **currentHp.** if **maxHp** and **currentHp** are lower than 1, they must be set to 1. (Make sure all variables are set in this exact order)

Method

Name	Description
+ int takeDamage (int damage)	**/* PARTIALLY PROVIDED */** This method is used to deal damage to the StandUser. • **The actual damage dealt is the parameter damage subtracted by total defense** (damage - TotalDefense) • **If isGuard is true, subtract by total defense x2 instead**

	(damage – (TotalDefense x 2)) • **After the calculation, if the actual damage is negative, it must be set to 0.** • Then **StandUser's currentHp must be subtracted by calculated damage** • **If the currentHp becomes negative, it must be set to 0.** • **This method returns the calculated actual damage.**
+ int doDamage (StandUser target)	**/* PARTIALLY PROVIDED */** This method calls target StandUser's takeDamage method. It returns that method's result. You must fill the commented code.
+ int getMaxHp()	maxHp getter method.
+ int getCurrentHp()	currentHp getter method.
+ void setCurrentHp (int hp)	currentHp setter method. **If hp is negative, it must be set to 0.**
+ boolean isGuard()	isGuard setter method.
+ void setGuard (boolean isGuard)	isGuard getter method.
+ void setStand (Stand stand)	Stand setter method.
+ void printShowStat()	Print StandUser stat.
+ StandUser select StandUser()	Selecting StandUser function during selecting character process.

4.1.3 Class Stand (Partially Given)

The class represents a support Character that the StandUser has. It has 7 fields total.

(4 of them are inherited from Character)

Additional Fields

Name	Description

- int maxChargeTurn	Contains the Stand charge require to activate.
- int currentChargeTurn	Contains the Stand current charge during gameplay.
- boolean isActive	Contains the Stand properties if it is activated or not.

Constructor

Name	Description
+ Stand (String name, String quote, int strength, int defense, int maxChargeTurn)	/* FILL CODE */ Assigning name, quote, strength, and defense by calling parent class's constructor. If the Stand's **maxChargeTurn** is lower than 1, it must be set to 1. **currentChargeTurn** must be 0 when a Stand is created.

Method

Name	Description
+ int getMaxChargeTurn()	maxChargeTurn getter method.
+ int getCurrentChargeTurn()	currentChargeTurn getter method.
+ void increaseCharge (int i)	This method increases **currentChargeTurn** by the value **i**. if **currentChargeTurn** is greater than **maxChargeTurn**, it must be set to **maxChargeTurn**.
+ void decreaseCharge (int i)	This method decreases **currentChargeTurn** by the value **i**. if **currentChargeTurn** is lower than 0, it must be set to 0.
+ boolean IsActive()	IsActive getter method.
+ void setIsActive (boolean isActive)	IsActive setter method.
+ void printShowStat()	Print Stand stats.
+ Stand selectStand()	Selecting Stand function during selecting character process.

4.2. Package logic (Do not modify any class in this package.)

4.2.1. Class GameHandle

The class contains functions that handlle gameplay in single player mode.

4.2.2. Class main

The main class handles main menu, selecting character, adding character, etc.

5. Score Criteria (Running JUnit is enough. You do not need to run main. The main method is just for your curiosity) Total score is 29 marks (will be scaled to equal to other labs)

Make UML for all classes you modified, or your score will be capped to 80%.

StandTest

testStandConstructor()	**2 marks**

StandUserTest

testConstructor()	**2 marks**
testTakeDamageWithoutStand_NoGuard()	**2 marks**
testTakeDamageWithoutStand_Guard()	**2 marks**
testTakeDamageWithStand_NoGuard()	**4 marks**
testTakeDamageWithStand_Guard()	**4 marks**
testTakeDamageWithStand_NegativeHP()	**4 marks**
testTakeDamageWithStand_NegativeDamage()	**5 marks**
testDoDamage_StandNotActive()	**2 marks**
testDoDamage_StandActive()	**2 marks**