Lab 2: Inheritance

1. Instruction

1.1. Click the provided link on CourseVille to create your own repository.

1.2. Open Eclipse and then "File > new > Java Project" and set project name in this format **2110215_Lab2_2022_2_{ID}_{FIRSTNAME}**

   1.2.1. Example: 2110215**_Lab**2_202**2_2_**6**5**31234521**_Samatcha**.

1.3. Initialize git in your project directory

   1.3.1. Add .gitignore and set up your git.

   1.3.2. Create your remote repository from a given link.

   1.3.3. Commit and push initial codes to your GitHub repository.

1.4. Implement all the classes and methods following the details given in the problem statement file which you can download from CourseVille.

   1.4.1. You should create commits with meaningful messages when you finish each part of your program.

   1.4.2. Don't wait until you finish all features to create a commit.

1.5. Test your codes with the provided JUnit test cases, they are inside package **test.grader**

   1.5.1. If you want to create your own test cases, please put them inside package **test.student**

   1.5.2. Aside from passing all test cases, your program must be able to run properly without any runtime errors.

1.6. After finishing the program, create a UML diagram for classes in package logic.unit and put the result image (UML.png) at the root of your project folder.

1.7. Export your project into a jar file called **Lab2_2022_2_{ID}** and place it at the root directory of your project. Make sure you export all your source (.java) files. You can open the jar file with any zip software to check it. You jar file must include source code.

   1.7.1. Example: **Lab2_202**2**_2_**6**531234521.jar**

1.8. Push all other commits to your GitHub repository.

2. Problem Statement : Colosseum game

We are coding a game. This game is about a colosseum in the Roman Empire. It has competitors competing with other competitors.

- If a competitor has hp equals 0, that competitor will die (removed from the game).
- There are 4 types of competitors:
  - BaseCompetitor
  - Sorcerer
  - Tiger
  - ToughMan.

Every competitor has a name, hp and power. Each can attack another competitor.

A BaseCompetitor is a competitor who doesn't have any special skill.

- When a BaseCompetitor attacks, he will deal damage equal to his power to another competitor.

A sorcerer is a competitor who possesses a skill called "decrease power skill".

- A Sorcerer can use "decrease power skill" to decrease an enemy's power.
- When a Sorcerer attacks,
  - If the target is Tiger, the Sorcerer will deal damage equal to half of the Sorcerer's power.
  - If the target is ToughMan, the Sorcerer will deal damage equal to 1.5 times of the Sorcerer's power.
  - Otherwise, the Sorcerer will deal damage equal to the Sorcerer's power.

A Tiger is a competitor who possesses a skill called "train".

- A Tiger can use "train" skill to increase its own power.
- When a Tiger attacks,
  - If the target is ToughMan, the Tiger will deal damage equal to half of the Tiger's power.
  - If the target is Sorcerer, the Tiger will deal damage equal to 1.5 times of the Tiger's power.
  - Otherwise, it deals damage equal to its power.

A ToughMan is competitor who has a skill called "heal".

- A ToughMan can use "heal" skill to heal his own hp but not more than his max hp.
- When a ToughMan attacks,
  - If the target is Sorcerer, the ToughMan will deal damage equal to half of the ToughMan's power.
  - If the target is Tiger, the ToughMan will deal damage equal to 1.5 times of the ToughMan's power.

- Otherwise, the ToughMan will deal damage equal to the ToughMan's power.

The game will end when there is only 1 competitor left. That competitor wins the game.

3. Implementation Details

3.1 Package logic.unit /* You must implement this package from scratch */

3.1.1 Class BaseCompetitor /* You must implement this class from scratch */

Variable

Name	Description
- String name	This competitor's name.
- int power	This competitor's power. Power cannot be less than 1.
- int hp	This competitor's HP. Hp cannot be less than 0.

Method

Name	Description
+ BaseCompetitor(String name)	Intitiate this competitor with the given name, then set hp to 5, set power to 3.
+ BaseCompetitor(String name, int hp, int power)	set all fields according to the parameters. If power is less than 1, set it to 1. If hp is less than 0, set it to 0.
+ void attack(BaseCompetitor enemy)	This competitor attacks the given enemy, then the enemy would be dealt damage equal to this competitor's power.
+ String getType()	Type is between BaseCompetitor, Sorcerer, Tiger, ToughMan return a String that refers to the type.
+ getter,setter of all fields	Getter and Setter of all fields.

3.1.2 Class Sorcerer /* You must implement this class from scratch */

Method

Name	Description
+ Sorcerer(String name)	set name according to the given parameter, set hp to 4 and set power to 2.
+ Sorcerer(String name, int hp, int power)	Set all fields according to the parameters.
+ void lowerPower(BaseCompetitor enemy, int powerDown)	Decrease the enemy's power by powerDown. But enemy power can't be lower than 1. Beware of unusual case (see JUnit test).
+ void attack(BaseCompetitor enemy)	If the enemy is Tiger, deal damage equals to half of the Sorcerer's power to enemy. If

	the enemy is Sorcerer or BaseCompetitor, deal damage equal to our Sorcerer's power to the enemy. Otherwise, deal damage equal to 1.5 times of our Sorcerer's power to the enemy.

3.1.3 Class Tiger /* You must implement this class from scratch */

Method

Name	Description
+ Tiger(String name)	set name according to the given parameter, set hp to 7 and set power to 5.
+ Tiger(String name, int hp, int power)	Set all field values according to the parameters.
+ void train(int addPower)	Increase power by addPower. Beware of unusual case (see JUnit test).
+ void attack(BaseCompition enemy)	If enemy is ToughMan, deal damage equal to half of its power to the enemy. If enemy is Tiger or BaseCompetitor, deal damage equal to our Tiger's power. Otherwise, deal damage equal to 1.5 times of our Tiger's power to the enemy.

3.1.4 Class ToughMan /* You must implement this class from scratch */

Variable

Name	Description
- int maxHp	This competitor's max hp. It must be more than or equal to its hp.

Method

Name	Description
+ToughMan(String name)	Set name according to the given parameter, set hp as 8, set power as 4 and set maxHp as 8.
+ToughMan(String name, int hp, int power)	Set all fields according to the parameters. Set maxHp to be the same as hp.
+ void heal(int healHp)	Increase this unit hp by healHp. If hp after the increase is more than maxHp, set hp equal to maxHp. Beware of unusual case.
+ void attack(BaseCompition enemy)	If enemy is a Sorcerer, deal damage equal to half of the Toughman's power to the enemy. If the enemy is ToughMan or BaseCompetitor, deal damage equal to our Toughman's power to the enemy. Otherwise, deal damage equal to 1.5 times of power to the enemy.
+ getter,setter of maxHp	Getter and Setter of maxHp.

	For Setter, if maxHp parameter is lower than 0, Set maxHp to 0 and set hp to 0. If maxHp parameter is lower than hp, set hp to be the same as maxHp. Otherwise, set maxHp according to the given parameter.

3.2 Package logic.game /* You must implement something in this package */

3.2.1 Class GameSystem /* You must implement 1 method in this class */

Variable

Name	Description
- ArrayList <BaseCompetitor> allCompetitors	All Competitors currently in the game.
- Boolean gameEnd	True if game end.
- GameSystem instance	A static instance of game system. This will make sure that there is only one game system when our program is running.

Method

Name	Description
- GameSystem()	This is the Constructor. Initialize the competitors ArrayList and the competitors. Add all the competitors to its ArrayList. Set gameEnd to false.
+ void addNewCompetitor(String name,int choice)	/* FILL CODES */ Add new competitor with the same name as the given parameter. Choice 0,1,2,3 is BaseCompetitor, Sorcerer, Tiger and ToughMan respectively. You are only allowed to declare at most 1 new variable. If you do not comply, there will be no score for this method.
+ void printCompetitorsStatus()	Print all competitors' type, name, hp and power.
+ void removeDeadCompetitors()	Remove all competitors with hp equal to 0 or lower.
+ getter,setter of all fields	Getter, Setter of all fields.

Package application

Class Main

This class is the main program. You don't have to implement anything in this class. You can test the program by running this class.

Score Criteria (30 -> will be scaled to 2.5)

BaseCompetitorTest (7)

- testConstructor (1)
- testBadConstructor (1)
- testAttack (1)
- testGetType (1)
- testSetName (1)
- testSetHp (1)
- testSetPower (1)

SorcererTest (4)

- testConstructor (1)
- testBadConstructor (1)
- testLowerAttack (1)
- testAttack (1)

TigerTest (4)

- testConstructor (1)
- testBadConstructor (1)
- testTrain (1)
- testAttack (1)

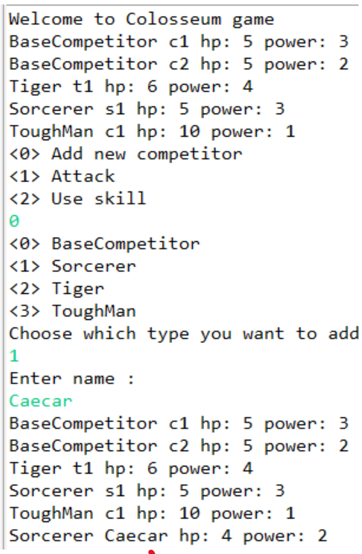
ToughManTest (5)

- testConstructor (1)
- testBadConstructor (1)
- testHeal (1)
- testAttack (1)
- testSetMaxHp (1)

Running the program (2)

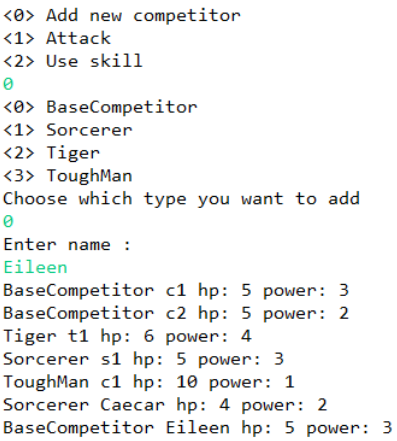
- A program can add a new competitor of all types. (0.5 for each type)

Example run:



```
Welcome to Colosseum game
BaseCompetitor c1 hp: 5 power: 3
BaseCompetitor c2 hp: 5 power: 2
Tiger t1 hp: 6 power: 4
Sorcerer s1 hp: 5 power: 3
ToughMan c1 hp: 10 power: 1
<0> Add new competitor
<1> Attack
<2> Use skill
0
<0> BaseCompetitor
<1> Sorcerer
<2> Tiger
<3> ToughMan
Choose which type you want to add
1
Enter name :
Caecar
BaseCompetitor c1 hp: 5 power: 3
BaseCompetitor c2 hp: 5 power: 2
Tiger t1 hp: 6 power: 4
Sorcerer s1 hp: 5 power: 3
ToughMan c1 hp: 10 power: 1
Sorcerer Caecar hp: 4 power: 2
```

new



```
<0> Add new competitor
<1> Attack
<2> Use skill
0
<0> BaseCompetitor
<1> Sorcerer
<2> Tiger
<3> ToughMan
Choose which type you want to add
0
Enter name :
Eileen
BaseCompetitor c1 hp: 5 power: 3
BaseCompetitor c2 hp: 5 power: 2
Tiger t1 hp: 6 power: 4
Sorcerer s1 hp: 5 power: 3
ToughMan c1 hp: 10 power: 1
Sorcerer Caecar hp: 4 power: 2
BaseCompetitor Eileen hp: 5 power: 3
```

new

UML (8)

- UML of each class in logic.unit (0.5 for each class)
- Showing relationships that allow reuse of code from base class (6)