Lab 1 : Object-Oriented Programming

Instruction

1. Click the provided link on CourseVille to create your own repository.
2. Open IntelliJ and then “Create a project” and set project name in this format **Progmeth_Lab1_2023_2_{ID}_{FIRSTNAME}**
    - Example: **Progmeth_Lab1_2023_2_6631234521_Samatcha**.
3. Initialize git in your project directory
    - Add .gitignore.
    - Commit and push initial codes to your GitHub repository.
4. Implement all the classes and methods following the details given in the problem (some files or part of the files are already given, please see the src folder in the given file) statement file which you can download from CourseVille.
    - You should create commits with meaningful messages when you finish each part of your program.
    - Don’t wait until you finish all features to create a commit.
5. Test your codes with the provided JUnit test cases, they are inside package **test.grader**
    - If you want to create your own test cases, please put them inside package **test.student**
    - Aside from passing all test cases, your program must be able to run properly without any runtime errors.
6. After finishing the program, create a UML diagram and put the result image (UML.png) at the root of your project folder.
7. Export your project into a jar file called **Lab1_2023_2_{ID}** and place it at the root directory of your project. Your jar file must contain source code. Your jar file must be on GitHub too.
    - Example: **Lab1_2023_2_6631234521.jar**
8. Push all other commits to your GitHub repository.

1. Problem Statement: Dungeon Game

As everyone knows, in lots of games there are players and monsters, if a player kills a monster, the monster then drops items (food and potion). In this lab, your job is to implement part of a simple dungeon game. This program can add new players, monsters, foods and potions to your game. You can buy foods, potions and ores from the market and you can sell your foods and potions. Everyday, the player's energy will decrease by 3. If the player's energy is less than 0, it will decrease that player's hp. But eating food can increase energy. Drinking potions will increase player's status. The game ends when a player has four kinds of ores (Platinum, Gold, Silver, Bronze) and that player will be a winner.

The program example is shown below. The program should be run from the Main class in the package main.

```
Enter any string to go to next step
1
=====================================================
Choose an option
-----------------------------------------------------
<1> Add Market
<2> Add Monster
<3> Add Player
<4> Add Food to the market
<5> Add Potion to the market
<6> Show all information of player
<7> Show all information of monster
<8> Buy Food or Potion From Market
<9> Player Attack Monster
<10> Monster Attack Player
<11> Buy Ore
<12> Drink Potion
<13> Eat Food
<14> Sell Potion
<15> Sell Food
<16> End Day
-----------------------------------------------------
<0> Exit
=====================================================
```

At the main menu, There are 16 options

1. Add Market

This option will create a new market. You must type the name of the market, The market is then created. In our program, **a market name can only have 1 word or the program will crash**.

```
=====================================================
1
=====================================================
Enter market name
Harborough
Market created
Enter any string to go to next step
```

2. Add Monster

This option will create a new monster. You must type the monster's name, hp, durability, attack and magic of monster. Then the monster is created. Hp must be positive. Durability, attack, and magic must be non-negative. **A monster name can only have 1 word or the program will crash**.

```
==========================================================
Enter monster name
Drake
Enter hp
15
Enter durability
3
Enter attack
2
Enter magic
4
Monster created
Enter any string to go to next step
```

3. Add Player

This option will create a new player. You must type the player's name, hp, durability, attack and magic of that player. Then that player is created. Hp must be positive but durability, attack, and magic must be non-negative. **A player name can only have 1 word or the program will crash**.

```
==========================================================
Enter player name
Jojo
Enter hp
5
Enter durability
4
Enter attack
3
Enter magic
1
Player created
Enter any string to go to next step
```

4. Add Food to Market

This option will create new food and put it into a chosen market. You must choose the market, then type food name, price, and energy. **Food name can only have 1 word or the program will crash. Price and Energy must be positive (an error message is printed on screen if you give non positive value)**.

```
==========================================================
4
==========================================================
Choose market
<0> Taiyo
0
Enter food name
Nuggets
Enter food price
32
Enter food energy
50
Food is added to market
Enter any string to go to next step
```

5. Add Potion to Market

This option will create a new potion and put it into a chosen market. You must choose the market, then type potion price and the amount of each increasing statuses. **Price must be positive and increasing statuses must be non-negative, or the operation is not carried out**. **Name must have only 1 word, or the program will crash.**

```
==========================================================
Choose market
<0> Taiyo
0
Enter Potion name
Phoenix
Enter Potion price
30
Enter Increasing Hp
100
Enter Increasing durability
44
Enter Increasing attack
2
Enter Increasing magic
3
Potion is added to market
Enter any string to go to next step
```

6. Show all information of player

This option will show all the information of a specified player. You must choose a player to show more detail information. The system then shows the player's name, status, food, potion, and ore.

```
6
=====================================================
Choose player
<0> Player: Akira
Hp: 10 Durability: 2 Attack: 4 Magic: 2 Energy: 10 Money: 100
<1> Player: Ayane
Hp: 5 Durability: 3 Attack: 3 Magic: 6 Energy: 6 Money: 805
<2> Player: Jojo
Hp: 5 Durability: 4 Attack: 3 Magic: 1 Energy: 10 Money: 100
1
Player: Ayane
Hp: 5 Durability: 3 Attack: 3 Magic: 6 Energy: 6 Money: 805
Food:
Bacon
Beef
Potion:
Middle Heal
Phoenix
Ore:
Gold
Silver
Enter any string to go to next step
```

7. Show all information of monster

This option will show all information of specific monster. You must choose the monster to show detail information. Then the system will show the monster's name, status, food dropped by the monster, and potion dropped by the monster.

```
=====================================================
7
=====================================================
Choose monster
<0> Monster: Boar
Hp: 12 Durability: 2 Attack: 1 Magic: 2
<1> Monster: Basilisk
Hp: 7 Durability: 4 Attack: 6 Magic: 0
1
Monster: Basilisk
Hp: 7 Durability: 4 Attack: 6 Magic: 0
Drop Food: Basilisk meat, Drop Potion: Medium Potion
Enter any string to go to next step
```

8. Buy Food or Potion From Market

This option will buy food or potion from the market. You must choose whether to buy potion or buy food.

```
=====================================================
Choose one choice
<1> Buy Potion
<2> Buy Food
```

If you choose to buy potion, you must choose a player to be the buyer and choose a market. Then choose the potion you want to buy.

```
8
=====================================================
Choose one choice
<1> Buy Potion
<2> Buy Food
1
Choose player who is buyer
<0> Akira
Money: 100
<1> Ayane
Money: 1200
1
Choose Market
<0> Taiyo
0
Choose potion you want to buy
<0> Middle Heal
Increasing Status Hp: 4 Durability: 0 Attack: 0 Magic: 0
Price: 15
<1> Middle Attack
Increasing Status Hp: 0 Durability: 0 Attack: 4 Magic: 0
Price: 15
1
Buy potion completed
Enter any string to go to next step
```

If you choose to buy food, you must choose a player to be your buyer and choose a market. Then choose food you want to buy.

```
8
=====================================================
Choose one choice
<1> Buy Potion
<2> Buy Food
2
Choose player who is buyer
<0> Akira
Money: 100
<1> Ayane
Money: 1185
1
Choose Market
<0> Taiyo
0
Choose food you want to buy
<0> Bacon
Price: 10 Energy: 5
<1> Chicken
Price: 15 Energy: 7
<2> Beef
Price: 20 Energy: 12
2
Buy food completed
Enter any string to go to next step
```

9. Player Attack Monster

This option makes a player attack a monster. You must choose a player (your attacker) and a monster being attacked. Then choose the type of attack. Normal attack damage gets

reduced according to the monster's durability but Magic attack damage cannot be reduced. If hp of the monster becomes 0, you get food and potion.

```
==========================================================
9
==========================================================
Choose Player which is attacker
<0> Akira
<1> Ayane
1
Choose Monster which is attacked
<0> Boar
<1> Basilisk
0
Choose type of attack
<1> Normal Attack
<2> Magic Attack
2
Monster: Boar now have hp 6
Enter any string to go to next step
```

10. Monster Attack Player

This option makes a monster attack a player. You must choose the attacking monster and the player. Then choose the type of attack. Normal attack damage will be decreased according to the player's durability value, but Magic attack damage will not decrease.

```
==========================================================
10
==========================================================
Choose Monster which is attacker
<0> Boar
<1> Basilisk
1
Choose Player which is attacked
<0> Akira
<1> Ayane
1
Choose type of attack
<1> Normal Attack
<2> Magic Attack
1
Player: Ayane now have hp 0
Enter any string to go to next step
```

11. Buy Ore

This option player will buy ore. You must choose the player and the ore. If the player possesses Gold, Silver, Platinum, Bronze ores, then that player wins the game.

```
==========================================================
11
==========================================================
Choose player
<0> Akira
Money: 100
Ores:
<1> Ayane
Money: 1200
Ores:
1
Choose ore you want to buy
<0> Gold Cost: 180
<1> Silver Cost: 140
<2> Platinum Cost: 250
<3> Bronze Cost: 40
0
Buy ore success
Enter any string to go to next step
```

12. Drink Potion

This option makes a player drink a bottle of specific potion (it must belong to the player). You must choose the player and the potion.

```
==========================================================
12
==========================================================
Choose player
<0> Akira
<1> Ayane
1
Choose Potion you want to drink
<0> Middle Heal
Increasing Status Hp : 4 Durable : 0 Attack : 0 Magic : 0
Price : 15
<1> Middle Attack
Increasing Status Hp : 0 Durable : 0 Attack : 4 Magic : 0
Price : 15
1
Drink Potion completed
Enter any string to go to next step
```

13. Eat Food

This option makes a player eat specific food (which belongs to the player). You must choose the player and the food.

```
================================================
13
================================================
Choose player
<0> Akira
<1> Ayane
1
Choose Food you want to eat
<0> Bacon
Energy : 5 Price : 10
<1> Beef
Energy : 12 Price : 20
0
Enter any string to go to next step
```

14. Sell Potion

This option makes a player sell a specific potion (which belongs to the player). You must choose the player and the potion.

```
======================================================
14
======================================================
Choose player
<0> Akira
<1> Ayane
1
Choose Potion you want to sell
<0> Middle Heal
Increasing Status Hp: 4 Durability: 0 Attack: 0 Magic: 0
Price: 15
0
Enter any string to go to next step
```

15. Sell Food

This option makes a player sell specific food (which belongs to the player). You must choose the player and the food.

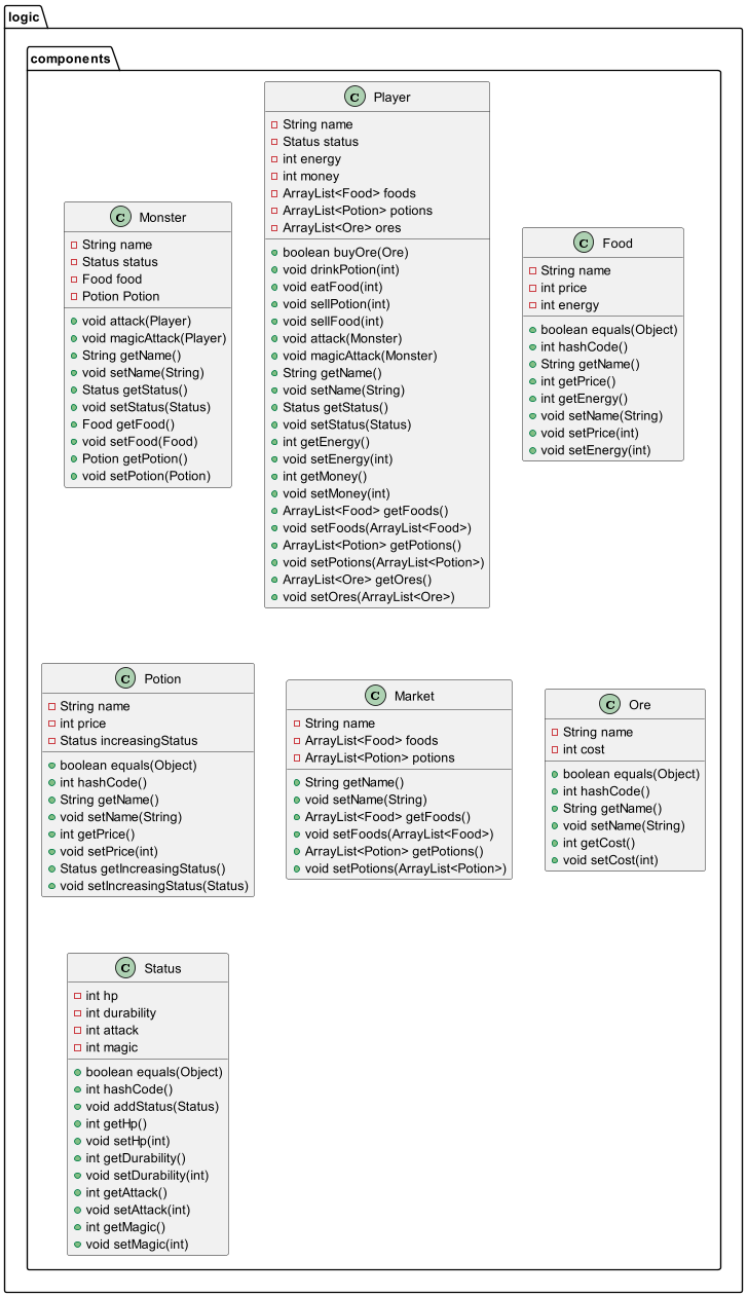
```
======================================================
15
======================================================
Choose player
<0> Akira
<1> Ayane
1
Choose Food you want to sell
<0> Beef
Energy: 12 Price: 20
0
Enter any string to go to next step
```

16. End Day

This option will end the current day. All players' energy will decrease by 3. If a player's energy becomes lower than 0, the player's hp will decrease.

2. Implementation Details:

The diagram of program is illustrated below. There are 7 classes: Food, Market, Monster, Ore, Player, Potion and Status.



** Note that Access Modifier Notations can be listed below*
** Also note that while other methods have been provided, only methods that you have to implement are listed here*

+ (public)
 # (protected)
- (private)

2.1 Package logic.components /* You must implement this package from scratch */
2.1.1 Class Food /* You must implement this class from scratch */

Fields

Name	Description
- String name	Food's name.
- int price	Food's price. Price must be positive.
- int energy	Food's energy. Energy must be positive.

Method

Name	Description
+ Food(String name,int price,int energy)	Initiate this food with the given value. If price is less than 1, set it to 1. If energy is less than 1, set it to 1.
+ boolean equals(Object o)	Return true if o is a Food that has the same name, price, and energy as **this**. (You need this method, otherwise Food comparison will only compare memory addresses of Food)
+ getter, setter of all fields	Getter and Setter of all fields.

2.1.2 Class Market /* You must implement this class from scratch */

Fields

Name	Description
- String name	Market's name.
- ArrayList<Food> foods	All foods in the market.
- ArrayList<Potion> potions	All potions in the market.

Method

Name	Description
+ Market(String name)	Initiate this market with the given value. Initiate foods and potions as new ArrayLists.
+ getter, setter of all fields	Getter and Setter of all fields.

2.1.3 Class Monster /* You must implement this class from scratch */

Fields

Name	Description
- String name	Monster's name.
- Status status	Monster's status.
- Food food	Food which drops when the monster is dead.
- Potion potion	Potion which drops when the monster is dead.

Method

Name	Description
+ Monster(String name, Status status)	Initiate this monster with the given value. - The monster hp () must be at least equal to 1. - Set food and potion to null. - Deal with exception inside constructor only (no exception is actually possible).
+ void attack(Player player)	Deals damage to player equals monster's attack. The damage is reduced by player's durability (damage is not less than 0). If the player's hp will become less than 0, find a way to set it to 0. Deal with exception inside this method only (no exception is actually possible).
+ void magicAttack(Player player)	Deals damage to player equals monster's magic. If the player's hp will become less than 0, find a way to set it to 0. Deal with exception inside this method only (no exception is actually possible).
+ getter, setter of all fields	Getter and Setter of all fields. Assume all values to set are legal.

2.1.4 Class Ore /* You must implement this class from scratch */

Fields

Name	Description
- String name	Ore's name.
- int cost	Ore's cost. Cost must be positive.

Method

Name	Description
+ Ore(String name, int cost)	Initiate this ore with the given value. If cost is less than 1, set it to 1.
+ equals(Object o)	Return true if o is an Ore with the same name, and cost as **this**. (You need this method, otherwise Ore comparison will only compare memory addresses of Ore)
+ getter, setter of all fields	Getter and Setter of all fields.

2.1.5 Class Player /* You must implement this class from scratch */

Fields

Name	Description
- String name	Player's name.
- Status status	Player's status.
- int energy	Player's energy. Energy must never be less than 0.
- int money	Player's money. Money must never be less than 0.
- ArrayList<Food> foods	Player's all foods.
- ArrayList<Potion> potions	Player's all potions.
- ArrayList<Ore> ores	Player's all ores.

Method

Name	Description
+ Player(String name, Status status)	Initialize this player with the given values. For the status: • hp can never go below 1 at the time of creation. Set energy to 10. Set money to 100. Initilize foods, potions, and ores as new ArrayLists. Deal with exception inside this constructor (no exception is actually possible). Note: Shorter code will get you more marks.
+ Player(String name, Status status, int energy, int money)	Initialize this player with the given values. For the status: • hp can never go below 1 at the time of creation. Initialize foods, potions, and ores as new ArrayLists. Deal with exception inside this constructor (no exception is actually possible).

+ boolean buyOre(Ore ore)	If money is more than or equals ore's cost:
	• then
	  ○ decrease the money by ore's cost.
	  ○ add ore to ores.
	  ○ return true.
	• Else
	  ○ return false.

+ void drinkPotion(int index)	Increase the player's status equals to potion's increasingStatus and remove potion from potions.

	index is a position of the potion to be drunk from potions ArrayList. If index indicates position outside the ArrayList, your code must do nothing. Deal with exception inside this method (no exception is actually possible).
+ void eatFood(int index)	Increase the player's energy equals food's energy and remove food from foods. index refers to the position of the food inside foods ArrayList. If index indicates position outside the ArrayList, your code must do nothing.
+ void sellPotion(int index)	Increase money equals potion's price and remove potion from potions. index refers to position in potions ArrayList. If index indicates position outside the ArrayList, your code must do nothing.
+ void sellFood(int index)	Increase money equals to food's price and remove food from foods. index refers to the position of the food in foods ArrayList. If index indicates position outside the ArrayList, your code must do nothing.
+ void attack(Monster monster)	Deal damage to monster equals player's attack and decrease the damage with monster's durability (damage is not less than 0). If the monster's hp will become less than 0, try to find a way to set it to 0. Deal with exception inside this method (no exception is actually possible).
+ void magicAttack(Monster monster)	Deal damage to monster equals player's magic. If the monster's hp will become less than 0, find a way to set it to 0. Deal with exception inside this method (no exception is actually possible).
+ getter, setter of all fields	Getter and Setter of all fields

2.1.6 Class Potion /* You must implement this class from scratch */

Fields

Name	Description
- String name	Potion's name.
- int price	Potion's price. Price must be positive.

- Status increasingStatus	Potion's increasing status.

Method

Name	Description
+ Potion(String name,int price, Status increasingStatus)	Initialize this potion with the given value. Make use of setters.
+ equals(Object o)	Return true if o is a Potion with the same name, price, and increasingStatus as **this**.
+ getter, setter of all fields	Getter and Setter of all fields. If price is less than 1, set it to 1.

2.1.7 Class Status /* You must implement this class from scratch */

Fields

Name	Description
- int hp	Health point. hp must not be less than 0.
- int durability	Durability value. It must not be less than 0.
- int attack	Attack value. It must not be less than 0.
- int magic	Magic value. It must not be less than 0.

Method

Name	Description
+ Status(int hp, int durability, int attack, int magic) throws BadStatusException	Initiate this status with the given value by using setter.
+ equals(Object o)	Return true if o is a Status with the same hp, durability, attack, and magic as **this**.
+ void addStatus(Status another) throws BadStatusException	Increasing all fields by adding values from another Status.
+ getter, setter of all fields throws BadStatusException	Getter and Setter of all fields. For Setter of all fields, if less than 0 then throws BadStatusException, else set it as parameter.

2.2 logic.game

2.2.1 Class GameController

Fields

Name	Description
+ ArrayList<Market> markets	All markets in the game.
+ ArrayList<Player> players	All alive players in the game.
+ ArrayList<Monster> monsters	All alive monsters in the game.
+ ArrayList<Ore> ores	All type of ores.
+ *GameController instance*	Game's Instance.
+ boolean gameEnd	Game's Status (whether the game is over or not).

Method

Name	Description
+ *GameController getInstance()*	If instance is null, then initialize GameController. Return the instance.
+ GameController()	Initialize all fields.
+ void addMarket(Market market)	Add market to markets field.
+ void addMonster(Monster monster)	Add monster to monsters field.
+ void addPlayer(Player player)	Add player to players field.
+ void endDay()	/*FILL CODE*/ Decrease all players' energy by 3. If a player's energy will change to below 0, deal damage to that player equals to the absolute value of the new energy, then set the energy to 0. If the player's hp will become less than 0, set it to 0. Deal with exception inside this method (no exception is actually possible).
+ void removeDeadPlayerAndMonster()	Remove all players and monsters with hp less than or equals to 0 from players and monster fields.
+ boolean buyOre(int iplayer, int jnumber)	ith player buys jth ore.
+ boolean checkGameEnd()	If gameEnd is false, check all players' ores, if any players have four ores, set gameEnd to true. Return gameEnd.
+ *Status createNewStatus(int hp, int durability, int attack,int magic)*	/*FILL CODE*/ Initialize new status. If any status is less than 0, return null. Else return new status.

Score criteria (45 -> will be scaled to 2.5)
FoodTest (2.5)

- testConstructor (1)
- testSetName (0.5)
- testSetPrice (0.5)
- testSetEnergy (0.5)

MarketTest (2.5)

- testConstructor (1)
- testSetName (0.5)
- testSetFoods (0.5)
- testSetPotions (0.5)

MonsterTest (4)

- testConstrutor (1)
- testAttack (0.5)
- testMagicAttack (0.5)
- testSetName (0.5)
- testSetStatus (0.5)
- testSetFood (0.5)

- testSetPotion (0.5)

OreTest (2.5)
- testConstructor (1)
- testSetName (0.5)
- testSetCost (1)

PlayerTest (9)
- testConstructor (1)
- testDetailedConstructor (1)
- testBuyOre (0.5)
- testDrinkPotion (0.5)
- testEatFood (0.5)
- testSellPotion (0.5)
- testSellFood (0.5)
- testAttack (0.5)
- testMagicAttack (0.5)
- testSetName (0.5)
- testSetStatus (0.5)
- testSetFoods (0.5)
- testSetPotions (0.5)
- testSetMoney (0.5)
- testSetOres (0.5)
- testSetEnergy (0.5)

PotionTest (2.5)
- testConstructor (1)
- testGetPrice (0.5)
- testSetPrice (0.5)
- testSetIncreasingStatus (0.5)

StatusTest (3.5)
- testConstructor (0.5)
- testConstructorException (0.5)
- testAddStatus (0.5)
- testSetHp (0.25)
- testSetHpException (0.25)
- testSetDurability (0.25)
- testSetDurabilityException (0.25)
- testSetAttack (0.25)
- testSetAttackException (0.25)
- testSetMagic (0.25)
- testSetMagicException (0.25)

GameController (2.5)
- testEndDay (0.5)
- testCreateNewStatus (1)
- testCreateNewStatusException (1)

UML (2)

Exception (4) (Run in main)

- When creating new player or monster, if any status is less than 0, that player or monster will not be created (3)
- When creating new player or monster, if all status more than or equals 0 then that monster or player is created successfully (1)

Fix main (10) such that:

- any invalid keyboard input in the first page of the menu (the one with choice 0 - 16) (for example, String input) prompt user to re-enter data. (5)
- wrong names and incorrect accesses are corrected. (5)