**Lab 3: Abstraction**

**Instruction**

1. Click the provided link on CourseVille to create your own repository.
2. Open Eclipse and then “File > new > Java Project” and set project name in this format **2110215\_Lab3\_2019\_2\_{ID}\_{FIRSTNAME}**
   * Example: **2110215\_Lab3\_2019\_2\_6131234521\_Thanos**.
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 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.
   * There will be additional test cases to test your code after you submit the final version, **make sure you follow the specifications in this document**.
6. After finishing the program, create a UML diagram using **ObjectAid** and put the result image (**UML.png**) at the root of your project folder.
7. Export your project into a jar file called **Lab3\_2019\_2\_{ID}** and place it at the root directory of your project.
   * Example: **Lab3\_2019\_2\_6131234521.jar**
8. Push all other commits to your GitHub repository

**1. Problem Statement : Children’s Card Game**

After your last success with creating solar system, You got an invite from the toy company and they need your help to create children’s card game. But this company has no experience making one before (but that didn’t stop them cause they want cash money), So you decided to make a simple card game for them.

## Gameplay

Player (You) Opponent



2.Player Turn Attack

3.Opponent Turn Attack

1.Player Play One Card

To simplify the game, Player will fight fixed opponent who never played any card, which mean most of the card function will only be used by player.

## **Game Component**

1.Player Character

- At the start of the game, both player will have a default character as an avatar, You choose Player Character before you choose your deck. Each have different stat of..

- Life Point = if Life Point of the character in played reach 0, that player lose the game.

- Attack Point = The amount of damage this card deal to the opponent when attack

- Defense Point = The amount of damage this card reduce when defend against opponent attack

2.Deck

- Contains as many card as you want, you can insert and remove any card at the card shop. However, you can have up to 4 of the same cards

3. Card

3.1 Character Card

- Similar to player character, Character card contains Life Point, Attack Point, and Defense Point. Player can choose to perform 1 of 2 actions when play this card.

1. Switch Character

- Replace current character in played with this card.

- Remaining Life Point percentage from previous character will be transfer to new character. (Ex Current character LP = 75/100 -> New character = 112/150)

- All of current character equipped item will be retired.

2. Sacrifice

- With the cost of this card, you can increase current character Life Point (by percentage). Different type of card increase different percentage of Life Point.

3.1.1 Basic Character Card

- Character card with only 2 previously mention actions.

- Sacrifice heal current character for 1/6 of current character max LP.

3.1.2 Main Character Card

- Character card that can level up when this card is in played and Player use the same card. Level up increase all 3 of character stats

- Sacrifice heal current character for 1/8 of current character max LP.

3.1.3 Exodia Character Card

- Character card that when equip with 4 Exodia Part Cards (Item Card). Player will immediately win the game

- Sacrifice heal current character for 1/10 of current character max LP.

3.2 Item Card

- Item Card is used to increased character stats, it contain LP bonus, Attack bonus, and Defense bonus.

- Item Card only has 1 action when played, equip item, which increase current character stats. Character can have any number of item card equipped, but all equipped item card will be lost when player switch character.

3.2.1 Basic Item Card

- Item card with only previously mention function.

3.2.2 Exodia Part Card

- Item card that only increase character Defense Point, if this card is equipped on to Exodia Character Card, Defense Point increase will be double.

## **Game Flow**

At the start of the game, player shuffle his deck and draw 5 card. And player character is played as a placeholder character.

1.Start with the player turn, player draw a card.

2.Player choose to play one card and pick its action.

3.Player character automatically attack opponent character

- Damage dealt = Attacker’s Total Attack Point – Defender’s Total Defense Point (Damage can’t be negative)

4.Start the opponent turn, opponent character automatically attack player character

5.Repeat until win condition is met by either player.

# **2. Implementation Details:**

To complete this assignment, you need to understand about **Abstract Classes** and **Exception Handling**.

To test your understanding about abstraction, **we will not provide class diagram for this assignment, and we will not indicate which methods and classes are abstract.** Try your best to figure out. There are **two** abstract classes and **four** abstract methods.

There are **four** packages in the provided files: application, card, deck and test.

You will be implementing most of the class in the card and deck package (Every class is partly given, while PremadeDeck class doesn’t need to modify)

There are some test cases given in package test.grader. These will help test your code whether it will be able to run or not. However, **some conditions are not tested** in these test cases. Look for those conditions in the following details, feel free to create your own test case in the package test.student.

There is an optional challenge for those who like some extra coding. It does not give you any extra score, so do not worry if you find it hard.

**You can define any additional number of private (but not public, protected or package) fields and methods** in addition to the fields and methods specified below. You are encouraged to try to group your logic into private methods to **reduce duplicate code as much as possible**.

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

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

## 

## **2.1** package deck

## **Class: Deck**

This class represents a deck player will used. Deck have 3 attributes: **name, deckSize, and deckList**. Player can assign deck to use in gameplay. **Deck can** **only have at most 4 of the same card**

- name = name of the deck

- deckSize = number of cards the deck contains, need to change when Deck is created, when card is inserted in to the Deck, and when card is removed from the Deck

- deckList = list of the cards this deck contains, each card must assign to each array without any empty slot.

### 2.1.1.1 Constructors

|  |  |
| --- | --- |
| + Deck(String name, Card[] deckList) | Construct a Deck object with the given name , deckList, and initialize deckSize to be the same size as deckList  (Hint: You should use Array (Card[]) to implement inventory.). |

### 2.1.1.2 Methods

|  |  |
| --- | --- |
| + int insertCard(Card card)  throws InsertCardFailedException | Inserts the given card into **the bottom of card list (Recommend create a new copy of deckList arrays with one more length)** and modified deckSize to be accurate to the new deckList  Throws an InsertCardFailedException with message *“*You can only put 4 of the same cards into the deck*”* if there are already 4 of same card in the deck. otherwise return the new deckSize |
| + Card removeCard(int slotNumber)  throws RemoveCardFailedException | Unequips a card in the given slot number from the deck, rearrange the card from every slots after to replace empty slot, modified deckSize to be accurate to the new deckList, and returns the removed card.  Throws an RemoveCardFailedException with message “Number you insert exceed deck size” when the slotNumber is greater than or equal deckSize, and with message “There is no card in that slot” if the slot is an empty slot. |
| + String toString() | You do **not** have to edit this method. |
| + int Card[] getDeckList()  + int String getName()  + int getDeckSize() | Getter methods. |
| + void setDeckSize(int deckSize) | Setter method. |

# **2.1.2 Class InsertCardFailedException**

This class is an **Exception** object thrown when failing to insert a card in to a hero. There can be multiple reasons why inserting a card fails (but in this assignment it is only when there is already 4 of the same card in the deck), so the Exception object will contain information about the reason.

### 2.1.2.1 Fields

|  |  |
| --- | --- |
| + String message | A string indicating the reason why the inserting failed to happen. |

### 2.1.2.2 Constructors

|  |  |
| --- | --- |
| + InsertCardFailedException(String message) | Initializes the Exception with the given message. |

# **2.1.3 Class RemoveCardFailedException**

Similar to the InsertCardFailedException class, this class is an Exception thrown when removing a card from the deck fails.

### 2.1.3.1 Fields

|  |  |
| --- | --- |
| + String message | A string indicating the reason why the removing failed to happen. |

### 2.1.3.2 Constructors

|  |  |
| --- | --- |
| + RemoveCardFailedException(String message) | Initializes the Exception with the given message. |

# **2.2** package card.base

## **2.2.1 Class Card**

This class is the base class of all card in the card shop. Each card has its own name and description. Card **should never be instantiated into objects**, as it is only designed to be a base class so that consumer classes (like Deck) can easily use their subclasses.

### 2.2.1.1 Constructors

|  |  |
| --- | --- |
| + Card(String name, String Description) | Initialize the card with the given name and description. |

### 2.2.1.2 Methods

|  |  |
| --- | --- |
| + String toString() | **this method should never be called with from this card, it can only be called from other class** |
| + String getName()  + String getDescription() | Getter methods. |

**2.2.2 Class CharacterCard**

This class is a base class of character card, used as an avatar for player to use in gameplay. It has lifePoint, attackPoint, and defensePoint. It provides shared implementation of displaying names across all upgradable items. CharacterCard **should never be instantiated to objects,** as it is only designed to be a base class so that consumer classes (like the Deck and Player) can easily use their subclasses.

### 2.2.2.1 Constructors

|  |  |
| --- | --- |
| + CharacterCard (String name, String Description , int lifePoint, int attackPoint, int defensePoint) | Initialize the character card with the given name ,description, lifePoint, attackPoint, and defensePoint |

### 2.2.2.2 Methods

|  |  |
| --- | --- |
| + void switchCharacter(Player player) | Switch character of the player, which also include changing player lifePoint, attackPoint, and defensePoint.  Hint:  Use setNewCharacterLifePoint(int)  ,setAttack(int)  , setDefense (int)  , setAssignedCharacter (CharacterCard)  from Player class |
| *+* int sacrifice(Player player) | Called when using CharacterCard to increase player life point. Life point gain is different for each class **this method should never be called from this card, it can only be called from other class** |
| + String toString() | You do **not** have to edit this method. |

**2.2.3 Class ItemCard**

This class is a base class of item card, player can equip item to increase lifePoint, attackPoint, and/or defensePoint. It has lpBonus, attackBonus, and defenseBonus. It provides shared implementation of displaying names across all item card. ItemCard **should never be instantiated to objects,** as it is only designed to be a base class so that consumer classes (like the Deck and Player) can easily use their subclasses.

### 2.2.2.1 Constructors

|  |  |
| --- | --- |
| + ItemCard(String name, String description, int lpBonus , int attackBonus, int defenseBonus) | Initialize the item card with the given name ,description, lpBonus, attackBonus, and defenseBonus |

### 2.2.2.2 Methods

|  |  |
| --- | --- |
| *+* void equipItem(Player player) | Called when using ItemCard to increase player stats, **this method should never be called with from this card, it can only be called from other class** |
| + String toString() | You do **not** have to edit this method. |

# **2.3** package card.cards

This package contains implementation of the **concrete** card in card shop. Some classes should extend classes from card.base package while some should extend classes from this package. In any case, every classes must be extended from Card class at its root.

**2.3.1 Class BasicCharacterCard**

This class represents “basic character card” that has no additional logic or properties

### 2.3.1.1 Constructors

|  |  |
| --- | --- |
| + BasicCharacterCard (String name, String description, int lifePoint, int attackPoint, int defensePoint) | Initialize the basic character card with the given name, description, and the attribute. |

### 2.3.1.2 Methods

|  |  |
| --- | --- |
| *+* int sacrifice(Player player) | Called when using BasicCharacterCard to increase player life point. Life point gain is different for each class  **Return life point increased from this method** (don’t have to consider whether healing lifePoint exceed maxLifePoint)  **In BasicCharacterCard case, player heal equal to 1/6 of player maxLifePoint**  Hint:  Use getMaxLifePoint()  , getCurrentLifePoint()  , healPlayer(int)  from Player class |

**2.3.2 Class BasicItemCard**

This class represents “basic items card” that has no additional logic or properties

### 2.3.1.1 Constructors

|  |  |
| --- | --- |
| + BasicItemCard(String name, String description, int lpBonus, int attackBonus, int defenseBonus) | Initialize the item card with the given name, description, and the attribute bonuses. |

### 2.3.1.2 Methods

|  |  |
| --- | --- |
| *+* void equipItem(Player player) | Called when using ItemCard to increase player stats **and add this ItemCard to player inventory.**  **In case of lpBonus, it must apply on both player’s currentLifePoint and maxLifePoint.**  Hint:  Use addInventory(Itemcard)  , setMaxLifePoint(int)  , setCurrentLifePoint(int)  , setAttack(int)  , setDefense(int)  from Player class |

**2.3.3 Class MainCharacterCard**

This class represents “main character card”. Very similar to BasicCharacterCard but with 2 more properties and 3 or methods. Main character start with level 0, if player currently played main character card, it can **levelUp** which increase level by 1 and multiplier player stats equal to **levelUpBonus**. (levelUp can be done by playing the same card as currently played main character card, but you don’t have to implement that part)

- int level = current level of this character, start with 0 when initialized

- float levelUpBonus = stat increase multiplier when levelUp,

Ex: If levelUpBonus = 0.3, when levelUp, player lifePoint, attackPoint, and defensePoint increase by 30% (round down to int)

### 2.3.3.1 Constructors

|  |  |
| --- | --- |
| + MainCharacterCard (String name, String description, int lifePoint, int attackPoint, int defensePoint, float levelUpBonus) | Initialize the main character card with the given name, description, levelUpBonus, and the attribute. |

### 2.3.3.2 Methods

|  |  |
| --- | --- |
| *+* int sacrifice(Player player) | Called when using MainCharacterCard to increase player life point. Life point gain is different for each class  **Return life point increased from this method** (don’t have to consider whether healing lifePoint exceed maxLifePoint)  **In BasicCharacterCard case, player heal equal to 1/8 of player maxLifePoint**  Hint:  Use getMaxLifePoint()  , getCurrentLifePoint()  , healPlayer(int)  from Player class |
| + float levelUp(Player player) | Called when using MainCharacterCard to **increase lifePoint, attackPoint, and defensePoint equal to its multiplier with levelUpBonus. Also increase level by 1**  **Return levelUpBonus**  Ex: If levelUpBonus = 0.3, when levelUp, player lifePoint, attackPoint, and defensePoint increase by 30% (round down to int)  Hint:  Use setNewCharacterLifePoint (int)  , setAttack (int)  , setDefense (int)  from Player class |
| + String getName() | You do **not** have to edit this method. |
| + int getLevel() | Getter methods. |
| + void setLevel(int level) | Setter methods. |

**2.3.4 Class ExodiaCharacterCard**

This class represents “exodia character card”. Exodia is a CharacterCard with..

name = “"Exodia the Forbidden One"”

description = "With 4 or more Exodia Part Card equiped, you win the game"

lifePoint = 800

attackPoint = 0

defensePoint = 25

Additionally, it has one more method called winConditionCheck that check if Exodia is equip with 4 ExodiaPartCard or not. If it’s true, player win the game immediately (You don’t have to implement winning part, you only have to return true or false)

### 2.3.4.1 Constructors

|  |  |
| --- | --- |
| + ExodiaCharacterCard() | Initialize the main character card with the given name, description, and the attribute.  (See above) |

### 2.3.4.2 Methods

|  |  |
| --- | --- |
| *+* int sacrifice(Player player) | Called when using ExodiaCharacterCard to increase player life point. Life point gain is different for each class  **Return life point increased from this method** (don’t have to consider whether healing lifePoint exceed maxLifePoint)  **In BasicCharacterCard case, player heal equal to 1/10 of player maxLifePoint**  Hint:  Use getMaxLifePoint()  , getCurrentLifePoint()  , healPlayer(int)  from Player class |
| + boolean winConditionCheck(ItemCard[] inventory) | Called to check if inventory contents 4 or more ExodiaPartCard, return true if it’s 4 or more, false otherwise  Hint:  Use instanceof to check if ItemCard is ExodiaPartCard |

**2.3.5 Class ExodiaPartCard**

This class represents “exodia items card” Exodia is a BasicItemCard with..

description = "Assemble 4 of Exodia part card to win the game"

lifeBonus = 0

attackBonus = 0

name and defenseBonus is depend during initialization

### 2.3.5.1 Constructors

|  |  |
| --- | --- |
| + ExodiaPartCard(String name, int defense) | Initialize the main character card with the given name, description, and the attribute.  (See above) |

### 2.3.5.2 Methods

|  |  |
| --- | --- |
| *+* void equipItem(Player player) | Called when using ItemCard to increase player stats **(only defensePoint in this case) and add this ItemCard to player inventory.**  **If player character is ExodiaCharacterCard, increase player stats by twice of this bonus stats instead**  Hint:  Use addInventory(Itemcard)  , setMaxLifePoint(int)  , setDefense(int)  , getAssignedCharacter()  from Player class  Use instanceof to check if CharacterCard is ExodiaCharacterCard |

# 