North South University

Department of Electrical and Computer Engineering

CSE 215L (Programming Language II Lab)

Lab 11: Polymorphism

Objective:

• Get introduced to the concept of Polymorphism.

Polymorphism – The Basics:

In theory, Polymorphism means 'many forms', meaning the same task can be done in multiple approaches. Remember the time when we did method overriding? That ACT is one of the main ways of achieving Polymorphism, as we could bring our twists of the inherited methods in the subclasses.

For example, we create three classes, one being the superclass to the other two; all three classes have a common method. The name of the method is TestMethod() (just for testing purposes), and then the two subclasses have decided to use their use visions of the implementation of that certain TestMethod().

Do you know that you can pass an instance of a subclass to a parameter of its superclass type?

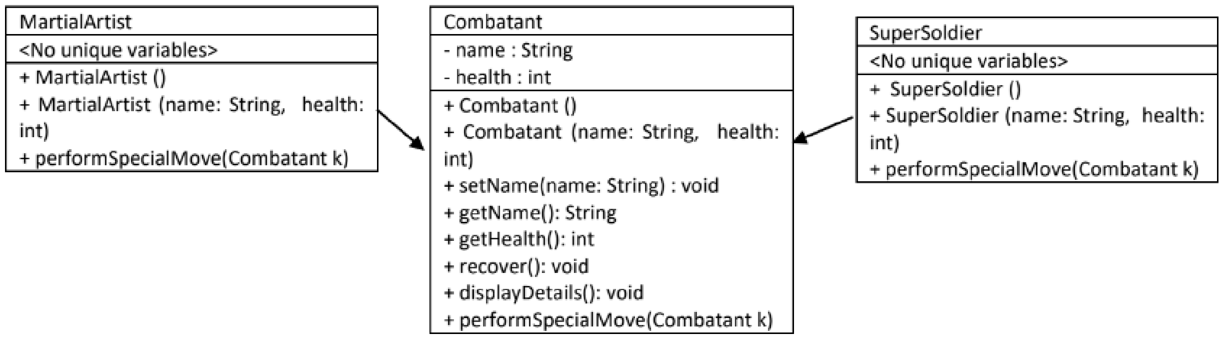
Let me give you an example here:

```
Superclass s1 = new Subclass1() ☑     Superclass s2 = new Subclass2() ☑     Subclass s1 = new Superclass() ☒
```

Technically, we can pass an instance of a subclass to a superclass parameter but not vice versa.

Classwork:

1. Consider the following UML Diagram(s) as given below:


MartialArtist
<No unique variables>
+ MartialArtist ()
+ MartialArtist (name: String, health: int)
+ performSpecialMove(Combatant k)
Combatant
- name : String
- health : int
+ Combatant ()
+ Combatant (name: String, health: int)
+ setName(name: String) : void
+ getName(): String
+ getHealth(): int
+ recover(): void
+ displayDetails(): void
+ performSpecialMove(Combatant k)
SuperSoldier
<No unique variables>
+ SuperSoldier ()
+ SuperSoldier (name: String, health: int)
+ performSpecialMove(Combatant k)

"Earthrealm's Protector, Atlee Kang had recruited **Vikram Rathore**, the Super Soldier from **Earthrealm**, to fight against **Sujon Majhi**, the formidable Martial Artist from **Outworld** who Jhantu Kahn picked."

These UML class diagrams are self-explanatory in that the SuperSoldier and MartialArtist class inherits the Combatant class, where the Combatant Class has the data fields name and health.

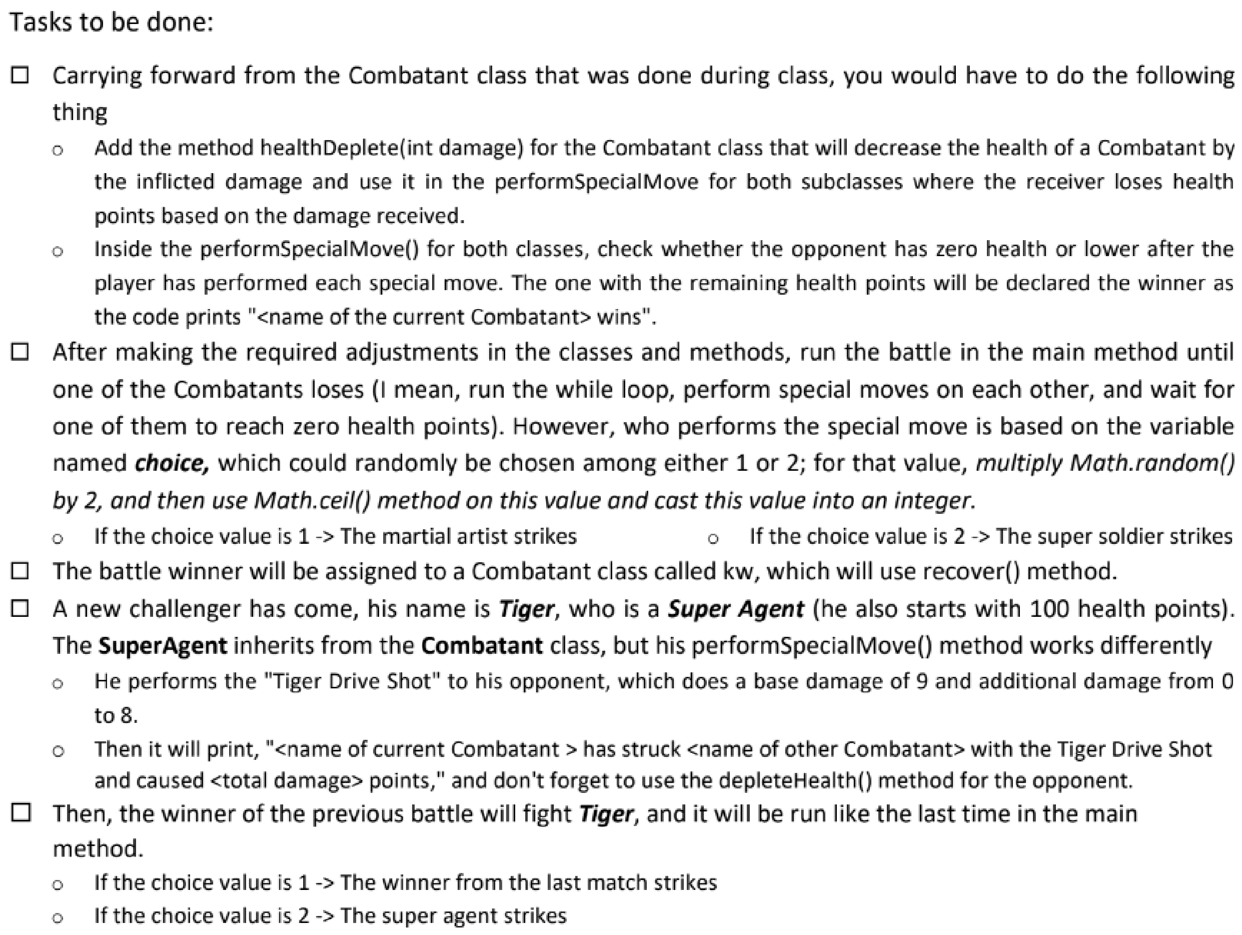
Here are some of the notes you need to consider:

- The health data field does not have mutator methods.
- displayDetails() method prints the name and health in the different lines.
- recover() method sets the health back to 100
- performSpecialMove(Combatant k) is implemented differently through all classes:
    - o From the Combatant class, it just prints, "Run it from the instance of the Child Class!"
    - o In the MartialArtist class, the character uses the "Majhi Smacker". It causes a base damage of 10 and an additional damage from 0 to 5.
        - Then it will print, "<name of current Combatant > has performed the Majhi Smacker to <name of other Combatant> by <total damage> points."
    - o In the SuperSoldier class, the character uses the "Jawan Punch". It causes a base damage of 8 and an additional damage from 0 to 10.
        - Then it will print, "<name of current Combatant> has caused <total damage> points to <name of other Combatant> by Jawan Punch"
    - o The damage has been calculated as follows:
        - Total damage = Base Damage + Additional Damage
        - Additional damage = x*Math.random() (Where x is the maximum damage)

Tasks to be done:

- Write the Classes of Combatant, SuperSoldier, and MartialArtist, and put the data fields and methods accordingly.
- Read the first paragraph after the UML diagrams to create classes of necessary objects. (k1 for the martial artist, and k2 for the super soldier), both will start with 100 health points.
- For both the martial artist and the super soldier, run the displayInfo() method and then the performSpecialMove method on each other.

Homework:



Tasks to be done:

- ☐ Carrying forward from the Combatant class that was done during class, you would have to do the following thing
  - Add the method healthDeplete(int damage) for the Combatant class that will decrease the health of a Combatant by the inflicted damage and use it in the performSpecialMove for both subclasses where the receiver loses health points based on the damage received.
  - Inside the performSpecialMove() for both classes, check whether the opponent has zero health or lower after the player has performed each special move. The one with the remaining health points will be declared the winner as the code prints "<name of the current Combatant> wins".
- ☐ After making the required adjustments in the classes and methods, run the battle in the main method until one of the Combatants loses (I mean, run the while loop, perform special moves on each other, and wait for one of them to reach zero health points). However, who performs the special move is based on the variable named ***choice,*** which could randomly be chosen among either 1 or 2; for that value, *multiply Math.random() by 2, and then use Math.ceil() method on this value and cast this value into an integer.*
  - If the choice value is 1 -> The martial artist strikes
  - If the choice value is 2 -> The super soldier strikes
- ☐ The battle winner will be assigned to a Combatant class called kw, which will use recover() method.
- ☐ A new challenger has come, his name is ***Tiger***, who is a ***Super Agent*** (he also starts with 100 health points). The **SuperAgent** inherits from the **Combatant** class, but his performSpecialMove() method works differently
  - He performs the "Tiger Drive Shot" to his opponent, which does a base damage of 9 and additional damage from 0 to 8.
  - Then it will print, "<name of current Combatant > has struck <name of other Combatant> with the Tiger Drive Shot and caused <total damage> points," and don't forget to use the depleteHealth() method for the opponent.
- ☐ Then, the winner of the previous battle will fight ***Tiger***, and it will be run like the last time in the main method.
  - If the choice value is 1 -> The winner from the last match strikes
  - If the choice value is 2 -> The super agent strikes