

Course: CSC-220.02

Student: Mya Phyu, SFSU ID: 921759134

Teammate: Jasmine Torres, SFSU ID: 921137929

Assignment Number: Assignment_02

Assignment Due Date & Time: 09-23-2022 at 11:59 PM

Part A- OOP Class Design Guidelines

Question: Please choose 3 guidelines and discuss them in-depth.

OOP- object-oriented programming is a paradigm which is a way of programming that organize data and modifiers to design products or applications.

1. Encapsulation

Encapsulation is the process of encapsulating something in a capsule. It has to do with being able to hide data in a class, to hide it from direct access of a client or user, “data hiding”. Variables of the class are declared private, to hide its data from users.

Encapsulation – Encapsulation is to use for hiding complexity not details. (you want to hide complexity from any user). Grouping functionality into a class that you will reuse and complicated processing can be encapsulated into a class and simple methods exposed to your other class For example, by implementing getters and setters, attributes are accessible from any other class.

With getters and setters we have publicly accessible methods that clients are able to call. Getters and setters protect your data, it allows control over values, and how you can access them and update your code to make it more efficient and less complex. The first letter of the variable in getter and setter should be capital. Getter methods are used when we want the data to be readable. It returns the value. Setter methods are put in place when you want the data to be ‘set’ or updated. With encapsulation in use we are making data inside inaccessible from the outside, there are no unexpected modifications from other parts of the program, or external code. It allows you to hide specific information and have more control over the program and methods. Encapsulation protects data and code from external intervention.

2. Inheritance vs. Aggregation

By using “extends” keyword in child class, child class can access everything from parent class. In other words, inheritance is saving programmers’ time by using “DRY” don’t repeat yourself method.

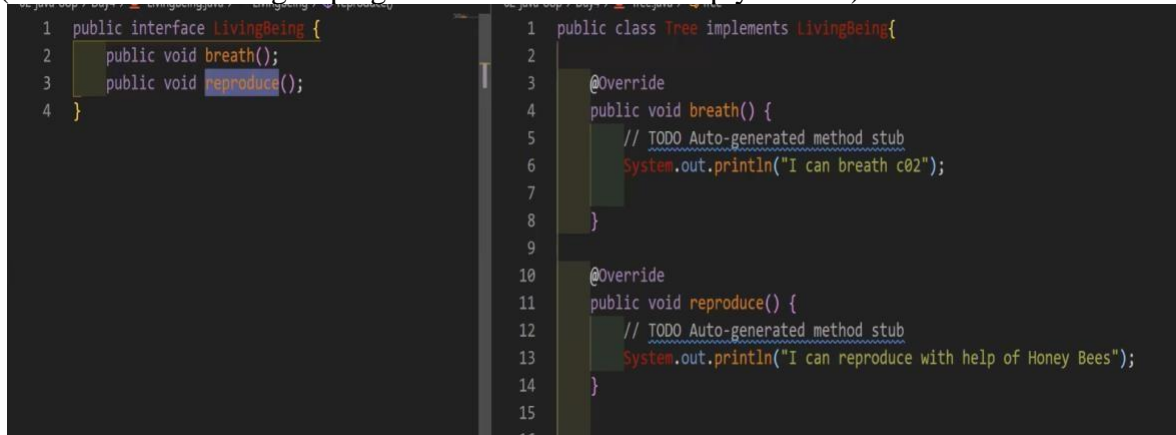
With inheritance we’re inheriting properties. Classes may inherit methods and properties of other classes. A subclass is a class derived from another class, and a superclass is the class from which a subclass is derived. Inheritance makes it easy to be able to reuse code, features, and methods. It makes it easier to minimize the amount of code we use to not over complicate and make it easier to navigate, easier to work with. When we use “extends” in our code for

inheritance, the superclass is being extended to the sub-class. In java there are different types of inheritance which are single, multiple, multilevel and hybrid Single level inheritance is an inheritance that only occurs for one class. Multi-level Inheritance includes the involvement of two or more classes. Multiple inheritance is when a subclass can inherit from more than one parent class. New derived classes can have more than one superclass. Hybrid Inheritance is more of a combination of inheritances. Aggregation represents a has-a relationship.

3. Interface

Interface cannot contain constructor and interface methods do not have a body.

(Interface is used for grouping related methods with empty bodies.)



```
1 public interface LivingBeing {
2     public void breath();
3     public void reproduce();
4 }
5
6 public class Tree implements LivingBeing{
7
8     @Override
9     public void breath() {
10         // TODO Auto-generated method stub
11         System.out.println("I can breath c02");
12     }
13
14     @Override
15     public void reproduce() {
16         // TODO Auto-generated method stub
17         System.out.println("I can reproduce with help of Honey Bees");
18     }
19 }
```

According to the above screenshot, Tree class can access methods and overriding those from the LivingBeing class with “implements” keyword in Tree class.

Interfaces are another way to achieve abstraction. It is an abstract class used to get related methods together with empty bodies. These interface methods have no body.

An interface must be implemented by another class using “implements”. The body of the interface is provided by the implement class. An interface can’t have a constructor. Interfaces can’t be used to create objects, it specifies what a class does. It’s a blueprint of the class.

All of the methods within the implementation of an interface must be overridden.

By default interface methods are public and abstract. As for its attributes, by default it’s public, static and final. Interfaces are used in instances when you want to hide specific details and only show necessary information.

4. Polymorphism

Polymorphism method overloading and constructor overloading. Can be used by other class, not only for one class.



```
Phone.java > Phone
1 public class Phone {
2     public void sound(){
3         System.out.println("There are
4             ringtones in phone.");
5     }
6 }
7
8

Iphone.java
1 public class Iphone extends Phone{
2     public void sound() {
3         System.out.println("Iphone has a
4             ringtone!");
5     }
6 }
7

Android.java > ...
1 public class Android extends Phone{
2     public void sound() {
3         System.out.println("Android
4             has a ringtone as well!");
5     }
6 }
7
```

Iphone class and Android class are allowed to use the same method and implement code as they desire in their class without creating a new method.

Polymorphism, meaning ‘numerous forms’, can be achieved using interfaces. Polymorphism occurs when there are many classes that are connected by inheritance. It allows you to perform a single action in many different ways, different forms. Polymorphism allows you to use the inherited attributes of these classes for different purposes.