

Part A- OOP Class Design Guidelines

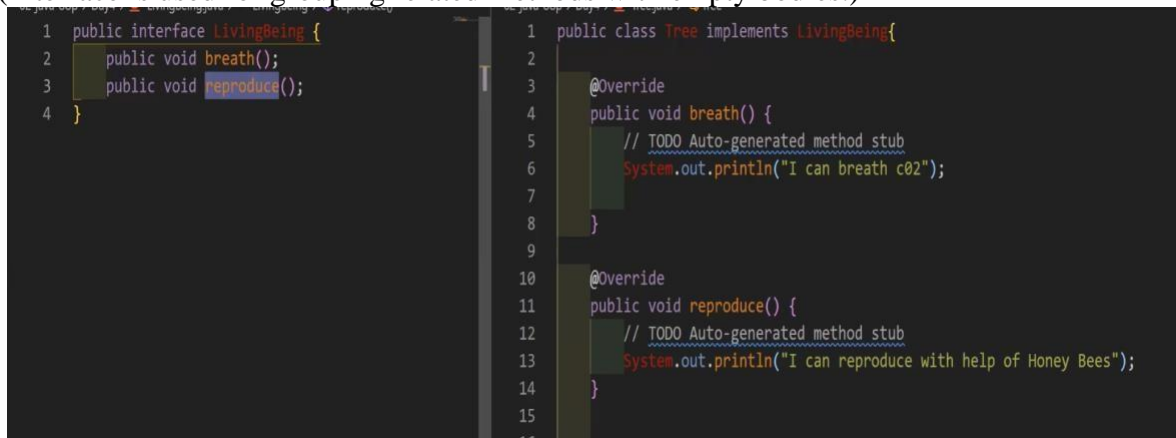
Question: Please choose 3 guidelines and discuss them in-depth.

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

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.

Inheritance- 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.

Interface- Interface cannot contain constructor and interface methods do not have a body. (Interface is used for grouping related methods with empty bodies.)

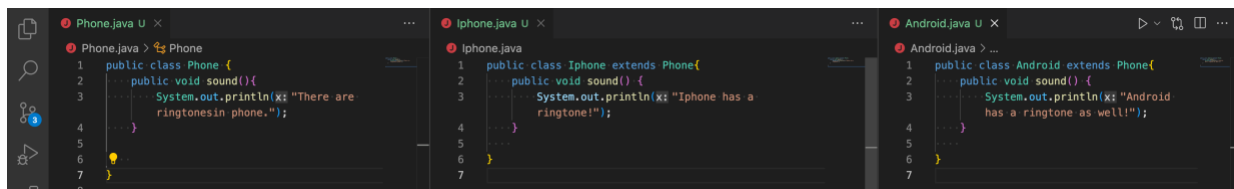


```
1 public interface LivingBeing {
2     public void breath();
3     public void reproduce();
4 }

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

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

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



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

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

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 }
```

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.