**Assignment 1**

**Question 1 - Explain abstract class and interface with the real-time scenario.**

**Ans-** A basic mobile phone is a type of interface, you deal with the buttons to operate it, Even if we have no idea what kind of implementation it have. Here we can create a interface mobile in which we add all the features ,We only get to use the feature without knowing what method it is using and how.

1. abstract class MobilePhone
2. {
3. public abstract void model();
4. public abstract void calculator();
5. public abstract void call();
6. public abstract void switchOff();
7. }

And we can see car as an example of abstract class, there are some pre-defined feature but also we can increase or decrease the features , there’s a flexibility to update the new feature . Here we can define some abstract methods and also some methods according to our need.

1. abstract class Car
2. {
3. public abstract void model();
4. public abstract void Engine();
5. public abstract void SwitchOn();
6. public static void DrivingMode() {
7. //Logic
8. }
9. public static void Speed() {

//Logic

1. }

**Question 2:** **How to decide when to use abstract class when to use interface. Justify with an example scenario**

**Answer:**

* Abstract class can be used where we want to have the flexibility in the program like if we want the derived class to include some particular methods and also some other methods too.

For example:

A banking System Home page , there are some methods which are must like Create account, check balance and also there is a flexibility if we want to add some other features on the Home page like some offers, scheme, or other new feature in the future.

* While we can use interface when we want all the feature to be included in the program , it is used when there is no plan for the future expansion in that section and feature are fixed and must be included. Any class that extend the interface have to include those abstract methods.

For Example:

In the banking website there should be a change password section in which there should be fixed four sections that is current password, new password, confirm password and otp verification . And also we can use interface for the login page where the sections are fixed that is username and password. We can use these feature without knowing what is inside them.

**Question 3: Examples on overriding rules**

**Answer:**

1. We can override the public method of Parent class

If we have the same method in the child class and the method in the parent class is public then the child class method will override the parent class method.

Example :

A.java

**package** com.rapipay.a;

**import** com.rapipay.b.B;

**public** **class** A **extends** B{

**public** **int** a1 =10;

**public** **static** **int** *count*;

**public** A() {

// System.out.println("default constructor of class A");

}

// @Override

**public** **void** methodB() { // for overriding purpose with Class B.

System.***out***.println("methodB in class A");

}

**public** A(**int** i) {

System.***out***.println("parametarized constructor of class A");

}

**public** **void** methodA() {

System.***out***.println("method in class A");

}

**public** **static** **void** main(String[] args) {

A a = **new** A();

a.methodB();

}

}

B.java

**package** com.rapipay.b;

//import com.rapipay.c.C;

**public** **class** B {

**public** B() {

System.***out***.println("default constructor of class B");

}

**public** B(**int** i) {

System.***out***.println("parametarized constructor of class B");

}

**public** **void** methodB() {

System.***out***.println("methodB in class B");

}

**public** **void** methodBB() {

System.***out***.println("method in class BB");

}

}

Output:

methodB in class A

1. We can override the protected method of the parent class in the same package.

If the method in the parent class is protected and in the same package then it can be override by the child class’s method

A.java

**package** com.rapipay.a;

**import** com.rapipay.b.B;

**public** **class** A **extends** B{

**public** **int** a1 =10;

**public** **static** **int** *count*;

**public** A() {

// System.out.println("default constructor of class A");

}

// @Override

**public** **void** methodB() { // for overriding purpose with Class B.

System.***out***.println("methodB in class A");

}

**public** A(**int** i) {

System.***out***.println("parametarized constructor of class A");

}

**public** **void** methodA() {

System.***out***.println("method in class A");

}

**public** **static** **void** main(String[] args) {

A a = **new** A();

a.methodB();

}

}

B.java

**package** com.rapipay.b;

//import com.rapipay.c.C;

**public** **class** B {

**public** B() {

System.***out***.println("default constructor of class B");

}

**public** B(**int** i) {

System.***out***.println("parametarized constructor of class B");

}

**protected** **void** methodB() {

System.***out***.println("methodB in class B");

}

**public** **void** methodBB() {

System.***out***.println("method in class BB");

}

}

Output:

methodB in class A