ASSIGNMENT-2

1) WAP TO PERFORM OVERLOADING

```
class OverloadExample { int add(int a, int b) {
return a + b;
}
int add(int a, int b, int c) { return a + b + c;
}
double add(double a, double b) { return a + b;
}
public static void main(String[] args) {
OverloadExample example = new OverloadExample();
System.out.println("Sum of two integers: " + example.add(5, 10));
System.out.println("Sum of three integers: " + example.add(5, 10, 15));
System.out.println("Sum of two doubles: " + example.add(5.5, 10.5));
}
}
```

2) WAP TO PRFORM OVERRIDING.

```
class Animal { void sound() {
System.out.println("Animal makes a sound");}} class Dog extends Animal {
@Override void sound() {
System.out.println("Dog barks");
}}class Cat extends Animal { @Override
void sound() {
System.out.println("Cat meows");
}}public class MethodOverridingExample { public static void main(String[] args)
{
Animal myAnimal; // Declare an Animal reference myAnimal = new Dog();
myAnimal.sound(); // Calls the Dog's sound method myAnimal = new Cat();
myAnimal.sound(); }}
```

3) WAP TO PRFORM INHERITANCE

- SINGLE

```
class Animal { void eat() {
System.out.println("Animal is eating");
}
}
class Dog extends Animal { void bark() {
System.out.println("Dog is barking");
}
}
public class SingleInheritanceDemo { public static void main(String[] args) {
Dog myDog = new Dog(); myDog.bark();
myDog.eat();
}
}
```

MULTIPLE

```
interface Printable { void print();
}
interface Shareable { void share();
}
class Document implements Printable, Shareable { @Override
public void print() {
System.out.println("Printing document...");
}
@Override
public void share() {
System.out.println("Sharing document...") }
}public class Main {
public static void main(String[] args) {
Document document = new Document(); document.print();
document.share(); }}
```

MULTILEVEL

```
class Animal { void eat() {
System.out.println("Animal is eating");
}}
class Dog extends Animal { void bark() {
System.out.println("Dog is barking");
}}
class Puppy extends Dog { void weep() {
System.out.println("Puppy is weeping");
}}
public class MultilevelInheritanceDemo { public static void main(String[] args)
{
Puppy myPuppy = new Puppy(); // Create a Puppy object myPuppy.weep(); // Calls
Puppy’s weep method myPuppy.bark(); // Calls Dog’s bark method (inherited)
myPuppy.eat(); // Calls Animal’s eat method (inherited)
}}
```

HYBRID

```
class Animal { void eat() {
System.out.println("Animal is eating");
}}
class Dog extends Animal { void bark() {
System.out.println("Dog is barking");
}}
class Puppy extends Dog { void weep() {
System.out.println("Puppy is weeping");
}}
public class MultilevelInheritanceDemo { public static void main(String[] args)
{
Puppy myPuppy = new Puppy(); // Create a Puppy object myPuppy.weep(); // Calls
Puppy’s weep method myPuppy.bark(); // Calls Dog’s bark method (inherited)
myPuppy.eat(); // Calls Animal’s eat method (inherited)
}}
```

HIERARCHICAL

```
class Animal { void eat() {
System.out.println("Animal is eating"); }} class Dog extends Animal {
void bark() {
System.out.println("Dog is barking");
}}class Cat extends Animal { void meow() {
System.out.println("Cat is meowing");
}}class Bird extends Animal { void chirp() {
System.out.println("Bird is chirping"); }} public class
HierarchicalInheritanceDemo {
public static void main(String[] args) {
Dog myDog = new Dog(); // Create a Dog object Cat myCat = new Cat(); // Create a
Cat object
Bird myBird = new Bird();
myDog.eat(); // Calls Animal's eat method myDog.bark();
myCat.eat(); // Calls Animal's eat method myCat.meow();
myBird.eat(); // Calls Animal's eat method myBird.chirp(); // Calls Bird's chirp
method
}}
```