JBK1008- Polymorphism

Method overloading by changing data type of Arguments

```
class Calculate{
 void sum (int a, int b){
  System.out.println("sum is"+(a+b)) ;
 }
 void sum (float a, float b){
  System.out.println("sum is"+(a+b));
 }
 Public static void main (String[] args) {
  Calculate  cal = new Calculate();
  cal.sum (8,5);     //sum(int a, int b) is method is called.
  cal.sum (4.6, 3.8); //sum(float a, float b) is called.
 }    }
```

Simple Method Overloading example

```
public class Overload {
            void demo(int a) {
                System.out.println("a: " + a);
        }
        void demo(int a, int b) {
                System.out.println("a and b: " + a + "," + b);
        }
        double demo(double a) {
                System.out.println("double a: " + a);
                return a * a;
        }
        int demo(int a ,int b,int c) {
                return a+b+c;
            }    }
public class MethodOverloading {
      public static void main(String[] args) {
                 Overload Obj = new Overload();
              double result;
              int add;
              Obj .demo(10);
              Obj .demo(10, 20);
              result = Obj .demo(5.5);
```

```
            System.out.println("O/P : " + result);
            add=Obj.demo(5, 5, 5);
            System.out.println("O/P : " + add);
        }           }
```

Method overloading by changing no. of argument.

```
class Area{
 void find(int l, int b) {
  System.out.println("Area is"+(l*b)) ;
 }
 void find(int l, int b,int h) {
  System.out.println("Area is"+(l*b*h));
 }
 public static void main (String[] args) {
  Area  ar = new Area();
  ar.find(8,5);     //find(int l, int b) is method is called.
  ar.find(4,6,2);    //find(int l, int b,int h) is called.
 } }
```

Program for method overloading

```
public class Student {
        String name;
        int age;
        String email;
        public void setData(String name,int age){
                this.name=name;  this.age=age;
        }
        public void setData(String name,int age, String email){
                this.name=name;  this.age=age;  this.email=email;
        }
        public void display(){
                System.out.println(name);
                System.out.println(age);
                System.out.println(email);
        }
        public static void main(String[] args) {
                Student s1=new Student();
                s1.setData("Shanthi", 20);
                s1. display();
                Student s2=new Student();
```

```
                s2.setData("Veera", 25,"veera@candidjava.com");
                s2. display();
                }   }
```

Example Of method Override:

```
public class BaseClass {

        public void method() //Base class method{
      System.out.println ("I'm the method of BaseClass");
   } }
public class DerivedClass extends BaseClass {
        public void method() //Base class method   {
      System.out.println ("I'm the method of DerivedClass");
   } }
public class Override {
        public static void main(String[] args) {
                // method calling from sub class object
                DerivedClass der = new DerivedClass();
                der.method();
                // method calling from super class object
                BaseClass base = new BaseClass();
                base.method();

                BaseClass base1 = new DerivedClass();
                base1.method();
           }       }
```

Create a Simple Method Overriding(Dynamic Binding) in Java

```
public class Bind_Dynamic {
        protected String val;
        void display (String str) {
                val = "Base Class Fuction ".concat(str);
                System.out.println(val);
        } }
class SubClass extends Bind_Dynamic{
        void display (String str) {
                if(val == null) {
                        str = "Derived Class Fuction ".concat(str);
                        System.out.println(str);
```

```
        } } }
class MainClass {
        public static void main(String args[]) {
                SubClass obj = new SubClass();
                obj.display("Called");
        }       }
```

Now rewrite the Code in SubClass and check changes of output

```
class SubClass extends Bind_Dynamic{

        void display (String str) {
                super.display(str);
                if(val == null) {
                        str = "Derived Class Fuction ".concat(str);
                        System.out.println(str);
                }     }       }
```

Complex Method Overriding (Dynamic Binding) example

```
public class Bind_Ex1 {
        String text = "Bind_Ex1's";
        void display() {
                System.out.println(text + " function called");
        } }
class SubClass1 extends Bind_Ex1 {
        void display() {
                super.display();
                text = "SubClass1's";
                System.out.println(text + " function called");
        } }
class SubClass2 extends SubClass1 {
        void display() {
                super.display();
                text = "SubClass2's";
                System.out.println(text + " function called");
        } }
class MainClass {
        public static void main(String args[]) {
                SubClass2 obj = new SubClass2();
```

```
                obj.display();
        } }
```

Method Overriding in hierarchical type

```
public class Bank {
        int getRateOfInterest() {
                return 0;
        } }
public class SBI extends Bank {
        int getRateOfInterest() {
                return 8;
        } }
public class ICICI extends Bank {
        int getRateOfInterest(){
                return 10;
                } }
public class Axis extends Bank {
        int getRateOfInterest() {
                return 11;
        } }
public class Override_Test {
        public static void main(String[] args) {
                Bank b=new Bank();
                System.out.println("Bank Rate of Interest :
"+b.getRateOfInterest()+"%");
Bank b1=new SBI();  Bank b2=new ICICI();
Bank b3=new Axis();
                System.out.println("SBI Rate of Interest :
"+b1.getRateOfInterest()+"%");
                System.out.println("ICICI Rate of Interest :
"+b2.getRateOfInterest()+"%");
                System.out.println("AXIS Rate of Interest :
"+b3.getRateOfInterest()+"%");
}}
```

Static and final

Program for Static variable Demo

```
class VariableDemo {
```

```
  static int count=0;
  public void increment()  {
     count++;
  } public static void main(String args[]) {
     VariableDemo obj1=new VariableDemo();
     VariableDemo obj2=new VariableDemo();
     obj1.increment();
     obj2.increment();
     System.out.println("Obj1: count is="+obj1.count);
     System.out.println("Obj2: count is="+obj2.count);
  } }
```

Program for Static Method Demo

```
public class StaticMethod_Demo {
        public static void copyArg(String str1, String str2) {
            //copies argument 2 to arg1
            str2 = str1;
            System.out.println("First String arg is: "+str1);
            System.out.println("Second String arg is: "+str2);
         } public static void main(String[] args) {
StaticMethod_Demo.copyArg("PQR", "DEF");    //this is first
method to call static method
           copyArg("XYZ", "ABC");      //this is second  method to
call static method
       } }
```

Program for static variables and methods.

```
public class MyStaticMethods {

   private String name;
   private static String staticStr = "STATIC-STRING";
   public MyStaticMethods(String n){
      this.name = n;
   }
   public static void testStaticMethod(){
      System.out.println("Hey... I am in static method...");
      //you can call static variables here
      System.out.println(MyStaticMethods.staticStr);
      //you can not call instance variables here.
```

```
  }
  public void testObjectMethod(){
    System.out.println("Hey i am in non-static method");
    //you can also call static variables here
    System.out.println(MyStaticMethods.staticStr);
    //you can call instance variables here
    System.out.println("Name: "+this.name);
  }
  public static void main(String a[]){
    //By using class name, you can call static method
    MyStaticMethods.testStaticMethod();
MyStaticMethods msm
= new MyStaticMethods("Java2novice");
    msm.testObjectMethod();
  } }
```

Program final Variable Demo

```
public class FinalEx {
final int a=10;
final void JBK(){
final int i=0;
for(i=0;i<5;i++)        //compile time error final variable's
value can't
be change
System.out.println("value of I+"+i);}
public static void main(String[] args) {
FinalEx finalEx=new FinalEx();
finalEx.JBK();
}
```

Program for Blank final variable

```
class Demo{
  //Blank final variable
  final int MAX_VALUE;
  Demo(){
    //It must be initialized in constructor
    MAX_VALUE=100;
  }  void myMethod(){
    System.out.println(MAX_VALUE);
```

```
  }  public static void main(String args[]){
    Demo obj=new  Demo();
    obj.myMethod();
  } }
```

Program for Final Method Demo

```
public class FinalEx1 {
FinalEx1() {
System.out.println ("This is a default constructer of
FinalEx2");
}
final int a = 100;
final void show() {
System.out.println (a);}
void welcome() {
System.out.println("Welcome to java by kiran,Pune");
} }
public class FinalExTest  extends  FinalEx1{
void show(){//compile time error because method cannot
override
System.out.println(This is method of FinalEx1Test"");
}
public static void main(String[] args) {
FinalEx1 finalEx1=new FinalEx1();
finalEx1.show();
} }
```

Program for final Class Demo

```
final class FinalClass {
void KiranShow(){
System.out.println("BY Kiran's way final class can not be
Inherite");
}}
public class  FinalClassTest  extends  FinalClass{
 //Here compile  time error because final class cannot be
extended
public static void main(String[] args) {
FinalClass finalClass=new FinalClass();
finalClass.KiranShow();
}}
```