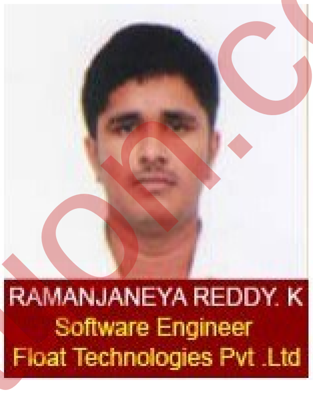
PROGRAMS ON OVERLOADING

Example problem on overloading:



```
public class Overload2
{
        void add(int m, int n)
        {
                int sum = m + n;
                System.out.println( "Sum of a+b is "
        +sum);
        }
        void add(int a, int b, int c)
        {
                int sum = a + b + c;
                System.out.println("Sum of a+b+c is " +sum);
        }
        void add(double a, double b)
        {
                double sum = a + b;
                System.out.println("Sum of a+b is "+sum);
        }
        void add(String s1, String s2)
        {
                String s = s1+s2;
                System.out.println(s);
        }
}
class overloadfunc
{
        public static void main(String args[])
        {
                Overload2 obj = new Overload2();
                Obj.add(4,19);
                obj.add(4,17,11);
                obj.add(1.5,21.5);
                obj.add("Life at"," the speed of rail ");
        }
}
```

Output will be:

```
Sum of a+b is 23
Sum of a+b+c is 32
Sum of a+b is 23.0
```

Q) Program of Method Overloading with Runtime Values

Solution:

```
import java.util.Scanner;
class cal
{
        void add(int a,int b)
        {
                System.out.print("Addition of First 2No.s is : " +(a+b));
        }
        void add(int a,int b,int c)
        {
                System.out.print("Addition of Three No.s is : " +(a+b+c));
        }
}
class mthdovrng
{
public static void main (String args[])
        {
                int a,b,c;
                Scanner input=new Scanner(System.in);
                System.out.print("Enter First No. : ");
                a=input.nextInt();
                System.out.print("Enter Second No. : ");
                b=input.nextInt();
                System.out.print("Enter Third No. : ");
                c=input.nextInt();
                cal ob=new cal();
                ob.add(a,b);
                ob.add(a,b,c);
        }
}
```

Q) You cannot overload the private method in Test class.

Solution: public class A

```
{
          int aa,bb,addd,pp,qq,rr,add2,ss,tt,uu,vv,add3;
          void overload(int a,int b)//===========1
          {
                    aa=a;
                    bb=b;
                    addd=aa+bb;
                    System.out.println("Addition is="+addd);
          }
private void overload(int p,int q,int r)//============2
          {
                    pp=p;
                    qq=q;
                    rr=r;
                    add2=pp+qq+rr;
                    System.out.println("Addition is="+add2);
          }
          protected void overload(int s,int t,int u,int v)//========3
          {
                    ss=s;
                    tt=t;
                    uu=u;
                    vv=v;
                    add3=ss+tt+uu+vv;
                    System.out.println("Addition is="+add3);
          }
```

```
                }
                class test
                {
                        public static void main(String a[])
                        {
                                A a1=new A();
                                a1.overload(23,55);
                                A a2=new A();
                                a2.overload(20,40,60);
                                A a3=new A();
                                a3.overload(20,40,6,100);
                        }
                }
```

Difference between Classes, Interfaces in Java

Class: A Java class is a virtual construct in Programming world, which can be instantiated to create Logical object to represent a Physical entity in the Virtual / Programming environment.

Every Object instantiated from a Class will have a state, and a behavior. These objects will communicate with other Objects in their virtual world using this behavior. We will take a deep dive about this communication between the objects in a short while.

Class Definition & Syntax:

```
   1: public class Vehicle
```

```
{ // syntax of class

        int steering; // these are the instance variables
        int wheels;
        Vehicle(int wheelCpunt)

        {

        //Isaconstructorofthelass
        wheels=wheelCount;

        }

        int getWheels()

        {
         return wheels;
        }

}
```

Interface: A Java Interface is also a virtual construct in programming world, but in contrary to the Java Class, Interface cannot be instantiated or created as an object. An Interface denotes a group of logical entities. It can also act as a contract between two sub systems.

Example program/Sample Program:

```
public interface Vehicle

{

        int steeringCount=2;
        int getWheels();
}
```

The above code snippet, defines an interface called Vehicle. An interface will not have a constructor and cannot be instantiated or an object cannot be created for this type. The variables in an Interface are called Class variables / Static variables, since they cannot have instances they should be able to communicate using the Class Name itself. In addition to being static variables these are also final by default which makes them not to be altered by any other objects.

Some of the differences between a class and an Interface

Property	**Class**	**Interface**
Instantiation	Can Be Instantiated	Cannot be instantiated
State	Each Object created will have its own state	Each objected created after implementing will have the same state
Behavior	Every Object will have the same behavior unless overridden.	Every Object will have to define its own behavior by implementing the contract defined.
Inheritance	A Class can inherit only one Class and can implement many interfaces	An Interface cannot inherit any classes while it can extend many interfaces
Variables	All the variables are instance by default unless otherwise specified	All the variables are static final by default, and a value needs to be assigned at the time of definition
Methods	All the methods should be having a definition unless	All the methods are abstract by default and they will not have a definition.

	decorated with an abstract keyword	

**When to use abstract methods in Java?

Why you would want to declare a method as abstract is best illustrated by an example.

Sample Program

```
/* the Figure class must be declared as abstract
   because it contains an abstract method  */
public abstract class Figure
        {
                /* because this is an abstract method the body will be blank*/
                public abstract float getArea();
        }
public class Circle extends Figure
        {
                private float radius;
                public float getArea()
                {
                        return (3.14 * (radius ^ 2));
                }
        }
public class Rectangle extends Figure
```

```
        {
                private float length, width;
                public float getArea(Figure other)
                {
                        return length * width;
                }
        }
```

Execution:

Both the Circle and Rectangle classes provide definitions for the getArea method, as you can see in the code above.

****Java interface versus abstract class**

An interface differs from an abstract class because an interface is not a class. An interface is essentially a type that can be satisfied by any class that implements the interface.

Any class that implements an interface must satisfy 2 conditions:

It must have the phrase "implements Interface_Name" at the beginning of the class definiton.

It must implement all of the method headings listed in the interface definition.

This is what an interface called "Dog" would look like:

Sample Example

```
public interface Dog
        {
                public boolean Barks();
                public boolean isGoldenRetriever();
        }
```

Now, if a class were to implement this interface, this is what it would look like:

Implementation:

```
public class SomeClass implements Dog
        {
                public boolean Barks
                {
                        // method definition here
                }
                public boolean is GoldenRetriever
                {
                        // method definition here
                }
        }
```

Test Your Skill

Q) What is the output of this program?

Input:

```
        Interface calculate
        {
                void cal(int item);
        }
        Class display implements calculate
        {
                int x;
        Public void call (int item)
                {
                        x = item * item;
```

```
            }
        }
        Class interfaces
        {
            Public static void main (String args[])
            {
                display arr = new display;
                arr.x = 0;
                arr.cal(2);
                System.out.print(arr.x);
            }
        }
```

Output:

Q) What is the output of this program?

Input:

```
    Interface calculate
        {
            void cal(int item);
        }
    Class displayA implements calculate
        {
            int x;
        Public void cal(int item)
            {
                x = item * item;
            }
```

```
      }
   Class displayB implements calculate
      {
             int x;
       Public void cal(int item)
             {
                   x = item / item;
             }
      }
   class interfaces
      {
     public static void main(String args[])
             {
             displayA arr1 = new displayA;
               displayB arr2 = new displayB;
               arr1.x = 0;
               arr2.x = 0;
             arr1.cal(2);
               arr2.cal(2);
               System.out.print(arr1.x + " " + arr2.x);
      }
   }
```

Output:

Q) What is the output of this program?

Input:

```
Interface calculate
        {
                int VAR = 0;
                Void cal (int item);
        }
       Class display implements calculate
        {
                int x;
                        Public void cal(int item)
                {
                        if (item<2)
                                x = VAR;
                 else
                                x = item * item;
                }
        }
 Class interfaces
        {
                        Public static void main(String args[])
                {
                        display [] arr=new display[3];
                        for(int i=0;i<3;i++)
                          arr[i]=new display();
                        arr[0].cal(0);
                        arr[1].cal(1);
                        arr[2].cal(2);
                        System.out.print(arr[0].x+" " + arr[1].x + " " + arr[2].x);
                }
        }
```

Output: