5]

```
public class ArraySum {
    public static int calculate(int[] array)
    { int sum=0;
        for(int num : array){
            sum+=num;
        }
        return Sum;
    }
public static void main(string[]arg
{ int[] number={10,12,30,40,50}
  int res = calculate(number);
  System.out.println("The sum= "+res);
  }
}
```

IT23012

6] Access modifiers in Java are keywords thats define the visibility or accessibility of classes, methods, constructors and variables with in the program. Java provides four access modifiers:

1. Public
2. Private
3. Protected
4. Default

Access Modifier	Accessible in class	Accessible in Package	Accessible Outside Packg by subclass	Accessible outside Package
Public	Yes	Yes	Yes	Yes
Private	Yes	No	No	No
Protected	Yes	Yes	Yes	No
Default	Yes	Yes	No	No

Public: Accessible from anywhere.

Example - public int a=25;

Private: Accessible in only the class w
its declared.

Example - private String name= N

IT23012

protected: Accessible within the same package and by subclasses

Example: protected int a = 8;

Default: Accessible only within the same package

Example: int count = 10;

IT23012

9) Method Overriding:

Method overriding is a feature in java that allows a subclass to provide a new implemention for a method that is already define in its superclass. How its works in inheritance -

i) When a subclass overrides a method, the subclass version of the method gets executed even if the method called on a superclass reference holding a superclass object.

ii) This process is called runtime polymorphism, because the method call is resolved at runtime.

1/23/2

What happens when a subclass overrides a method-

1. Subclass method executes replacing the superclass method
2. Runtime polymorphism determines method execution at runtime.
3. Supperclass method is hidden unless called using super
4. Overriding rules apply.
5. Super can call the overridde superclass method.

Using multiple classes.

Main.java,

```
public class Main{
    int x = 5;
}
```

Second.java,

```
class Second{
public static void main(
        String[] args){
    Main obj = new Main();
    System.out.println(obj.x);
}
}
```

run =>
```
>javac Main.java
>javac second.java
>java second
> 5
```

Class attributes,

```
public class Main {
    int x = 5;
    int y = 3;
}
```

here x and y is two attributes of Main class.

Modify Attributes,

```
public class Main {
    int x;
  public static void main(String[] arg)
    {
      Main obj = new Main();
       obj.x = 40;
      System.out.println(obj.x);
    }
}
```

IT23012

10] Difference between static and non-static members in Java.

feature	Static member	Non-static member
Definati-on	Belong to the class and are shared among all objects.	Belong to the individual objects, each instance has its own copy.
Access	Accessed using class name of instance	Access only through an object of the class.
memory allocation	Stored in the method area	stored in the heap memory
Invoeati-on	Can be called without creating an object.	Requires object creation before calling.
usage	Used for constants utility methods and shared properties.	used for object specific behavior and instance data.

IT23012

11] Difference between abstract class and interface -

Abstract	Interface
1] Abstract class have both abstract and concrete method.	1] Interface contains only abstract method.
2] Can have instance variable with any access modifier.	2] Can have only public, static, final constants.
3] Can have constructors	3] Cannot have constructor.
4] Can include both implemented and non-implemented method	4] All methods are public and abstract by default.
5] Example: abstract class Animal {}	Example: Interface Animal {}

12)
```
import java.util.scanner;
class Baseclass {
    void print result (string msg, Object n)
    { System.out.println(msg+n);
    } }
class SumClass extends BuseClass{
    double sumseries() {
        double sum = 0.0;
        for(double i=1.0; i>0.1; i-=0.1)
        { sum+=i;
        } return sum;
    }
Class DivisonMultipleClass BaseClass{
    int get (int a, int b) {
    while (b!=0) {
    int temp=b;
    b=a%.b;
```

```
        a = temp;}
        return a;
    }
Class NumberConversionClass extend
                    BaseClass{
    string toBinary(int num){
    return Integer.toBinaryString(num);
    }
    String toHex(int num){
    return Integer.toHexString(num).
        toUpperCase();
    }
    String toOctal(int num){
    return.toOctalString(num);
    }.
}
```

IT23012

```
class customPrintClass {
    void pr(String msg)
    System.out.println(">>" +msg);
    }
}
public class Mainclass {
    Public static void main(String[] args){
    Scanner sc=new scanner(System.in);
    SumClass sumobj = new sumClass();
    DivisonmultipleClass dmObj = new
                Divisonmultiple Class();
    NumberconversionClass noObj = new
                Numberconversionclass();
    Custompnint Class pnint Obj = new
                Custompnintclasspnint();
    }
}
```

```
    a = temp;}
    return a;
}
Class NumberConversionClass extend
                    BaseClass{
    string toBinary(int num){
    return Integer.toBinaryString(num
    }
    String toHex(int num){
    return Integer.toHexString(num).
        toUpperCase();
    }
    String toOctal(int num){
    return.toOctalString(num);
}.
```

```
class customPrintClass {
    void pr(String msg)
    System.out.println(">>" +msg);
    }
}
public class Mainclass {
    Public static void main(String [] args){
        Scanner sc = new scanner(System.in);
        SumClass sumobj = new sumClass();
        DivisonmultipleClass dmObj = new
                    DivisonmultipleClass();
        NumberconversionClass noObj = new
                    NumberconversionClass();
        CustompnintClass pnint Obj = new
                    CustompnintclassPnint();
    }
}
```

IT23012

14] import java.math.BigInteger;
import java.util.Scanner;

Public class FactorialCalculation {

Public static BigInteger f(int n){

BigInteger r = BigInteger.ONE;

for(int i=2; i<=n; i++){

r = r.multiply(BigInteger.valueof(i));

}

return r;

}

Public static void main(String[] args){

Scanner scn = new Scanner(System.in);

int n = scn.nextInt();

if (number<0){

System.out.println("Factorial is not

for this number");

IT23012

```
        }
        else{
            BigInteger fr=factorial(number);
            System.out.println("Factorial of "+number
                    +" is: "+factorialresult);
        }
        scn.close();
    }
}
```

IT23012

18] Public class cRge {

Public int myR(int i, int n){

int [] nandom = new int [n+i];

long cf = System.currentTimeMillis();

int it=1;

for(int j=1; j<=n; j++)

nandom[j] = it+2;

int nan = (int)(cf * nandom[i]) % 1000;

return nan;

}

}

import java.util.scanner;

public class RandNumber {

public static void main(String[] args)

int n = obj.nextInt();

cRge obj2 = new cRge();

for (int i=1; i<=n; i++)

```
{   int num = obj2.myRand(i,n);
    if(num<0)
    System.out.println(-1*num);

    else
    system.out.println(num);

    }
}
}
```