EXERCISE 1

a) Write a JAVA program to display default value of all primitive data type of JAVA

program:

```
class Demo
{
	static byte b;
	static short s;
	static int i;
	static long l;
	static float f;
	static double d;
	static char c;
	static boolean bl;
	public static void main(String[] args)
	{
		System.out.println("The default values of primitive data types are:");
		System.out.println("Byte :"+b);
		System.out.println("Short :"+s);
		System.out.println("Int :"+i);
		System.out.println("Long :"+l);
		System.out.println("Float :"+f);
		System.out.println("Double :"+d);
		System.out.println("Char :"+c);
		System.out.println("Boolean :"+bl);
	}
}
```

Output:

```
The default values of primitive data types are:
Byte :0
Short :0
Int :0
Long :0
Float :0.0
Double :0.0
Char :
Boolean :false
```

b) Write a java program that display the roots of a quadratic equation ax2 +bx=0. Calculate the discriminate D and basing on value of D, describe the nature of root.

Program:

```
import java.util.Scanner;
public class QuadraticEquationExample1
{
	public static void main(String[] Strings)
	{
		Scanner input = new Scanner(System.in);
		System.out.print("Enter the value of a: ");
		double a = input.nextDouble();
		System.out.print("Enter the value of b: ");
		double b = input.nextDouble();
		System.out.print("Enter the value of c: ");
		double c = input.nextDouble();
		double d= b * b - 4.0 * a * c;
		if (d> 0.0)
		{
			double r1 = (-b + Math.pow(d, 0.5)) / (2.0 * a);
			double r2 = (-b - Math.pow(d, 0.5)) / (2.0 * a);
			System.out.println("The roots are " + r1 + " and " + r2);
		}
		else if (d == 0.0)
		{
			double r1 = -b / (2.0 * a);
			System.out.println("The root is " + r1);
		}
		else
		{
		System.out.println("Roots are not real.");
		}
	}
}
```

Output:

```
Enter the value of a: 1
Enter the value of b: 1
Enter the value of c: 1
Roots are real not real.
```

c) Five Bikers Compete in a race such that they drive at a constant speed which may or may not be the same as the other. To qualify the race, the speed of a racer must be more than the average speed of all 5 racers. Take as input the speed of each racer and print back the speed of qualifying racers

Program:

```
import java.util.Scanner;
class Bike_Racers
{
	public static void main(String args[])
	{
		Scanner sc=new Scanner(System.in);
		int racer1_Speed,racer2_Speed,racer3_Speed,racer4_Speed,racer5_Speed;
		int sum;
		float avg_Speed;
		System.out.println("Enter 5 Bike Racers Speeds");
		racer1_Speed=sc.nextInt();
		racer2_Speed=sc.nextInt();
		racer3_Speed=sc.nextInt();
		racer4_Speed=sc.nextInt();
		racer5_Speed=sc.nextInt();
		 sum=racer1_Speed+racer2_Speed+racer3_Speed+racer4_Speed+racer5_Speed;
		avg_Speed=(float)sum/5;
		System.out.println("Average Speed is:"+avg_Speed);
		System.out.println("The Qualified Racers are:");
		if(racer1_Speed>avg_Speed)
		System.out.println(racer1_Speed);
		if(racer2_Speed>avg_Speed)
		System.out.println(racer2_Speed);
		if(racer3_Speed>avg_Speed)
		System.out.println(racer3_Speed);
		if(racer4_Speed>avg_Speed)
		System.out.println(racer4_Speed);
		if(racer5_Speed>avg_Speed)
		System.out.println(racer5_Speed);
	}
}
```

Output:

```
Enter 5 Bike Racers speed :255
247
289
```

220
236
Average speed is: 249.4
The qualified Racers are:
255
289

EXERCISE 2

a) Write a JAVA program to search for an element in a given list of elements using binary search mechanism

Program:

```
import java.util.*;
class BinarySearchExample
{
        public static void binarySearch(int arr[], int first, int last, int key)
       {
               int mid = (first + last)/2;
               while( first<= last )
               {
               if ( arr[mid] < key )
               {
               first = mid + 1;
               }
               else if(arr[mid] == key )
               {
               System.out.println("Element is found at index: " + mid);
               break;
               }
               else
               {
               last = mid - 1;
               }
               mid = (first + last)/2;
               }
               if ( first> last )
               {
               System.out.println("Element is not found!");
               }
       }
        public static void main(String args[])
       {
                Scanner sc=new Scanner(System.in);
               int n,key,arr[];
               System.out.println("Enter the number of elements");
               n=sc.nextInt();
               arr=new int[n];
               System.out.println("Enter "+n+" elements");
               for(int i=0;i<n;i++)
```

```
            arr[i]=sc.nextInt();
            System.out.println("Enter the number to search");
            key=sc.nextInt();
             int last=n-1;
            binarySearch(arr,0,last,key);
        }
}
```

Output:

```
Enter the number of elements
8
Enter 8 elements
11 22 33 44 55 66 77 88
Enter the number to search
44
Element is found at index: 3
```

b) Write a JAVA program to sort for an element in a given list of elements using bubble sort

Program:

```
import java.util.Scanner;
class BubbleSortExample
{
        public static void bubbleSort(int[] arr)
       {
              int n=arr.length,temp;
              for(int i=0;i<n-1;i++)
              {
              for(int j=0;j<n-i-1;j++)
              {
              if(arr[j]>arr[j+1])
              {
              temp=arr[j];
              arr[j]=arr[j+1];
              arr[j+1]=temp;
              }
              }
              }
        }
        public static void main(String[] args)
{
               int arr[],n;
               Scanner sc=new Scanner(System.in);
               n=sc.nextInt();
              arr=new int[n];
              for(int i=0;i<n;i++)
              arr[i]=sc.nextInt();
              System.out.println("Array Before Bubble Sort");
              for(int i=0; i<arr.length; i++)
               {
              System.out.print(arr[i] + " ");
               }
              System.out.println();
              bubbleSort(arr); //sorting array elements using bubble sort
```

```
            System.out.println("Array After Bubble Sort");
            for(int i=0; i<arr.length; i++)
            {
            System.out.print(arr[i] + " ");
             }
        }
}
```

Output:

```
8
1 4 2 3 5 9 8 6
Array Before Bubble Sort
1 4 2 3 5 9 8 6
Array After Bubble Sort
1 2 3 4 5 6 8 9
```

c) Write a JAVA program to sort for an element in a given list of elements using merge sort.

Program:

```
class Merge_Sort
{
	void merge(int a[], int beg, int mid, int end)
	{
		int i, j, k;
		int n1 = mid - beg + 1;
		int n2 = end - mid;
		int LeftArray[] = new int[n1];
		int RightArray[] = new int[n2];
		for (i = 0; i< n1; i++)
		LeftArray[i] = a[beg + i];
		for (j = 0; j < n2; j++)
		RightArray[j] = a[mid + 1 + j];
		i = 0;
		j = 0;
		k = beg;

		while (i< n1 && j < n2)
		{
		if(LeftArray[i] <= RightArray[j])
		{
		a[k] = LeftArray[i];
		i++;
		}
		else
		{
		a[k] = RightArray[j];
		j++;
		}
		k++;
		}
		while (i<n1)
		{
		a[k] = LeftArray[i];
		i++;
		k++;
		}

		while(j<n2)
		{
		a[k]=RightArray[j];
```

```
            j++;
             k++;
             }
        }
         void mergeSort(int a[], int beg, int end)
        {
            if(beg < end)
             {
             int mid = (beg + end) / 2;
            mergeSort(a, beg, mid);
            mergeSort(a, mid + 1, end);
            merge(a, beg, mid, end);
             }
            }
            void printArray(int a[], int n)
            {
             int i;
             for (i = 0; i< n; i++)
            System.out.print(a[i] + " ");
        }
        public static void main(String args[])
        {
             int a[] = { 11, 30, 24, 7, 31, 16, 39, 41 };
             int n = a.length;
            Merge_Sort m1 = new Merge_Sort();
            System.out.println("\nBefore sorting array elements are - ");
             m1.printArray(a, n);
             m1.mergeSort(a, 0, n - 1);
            System.out.println("\nAfter sorting array elements are - ");
             m1.printArray(a, n);
            System.out.println("");
        }
}
```

Output:

```
Before sorting array elements are -
11 30 24 7 31 16 39 41
After sorting array elements are -
7 11 16 24 30 31 39 41
```

d) Write a JAVA program using StringBuffer to delete, remove character.

Program:

```
class StringBuffer_Demo
{
	public static void main(String args[])
	{
		StringBuffer s1=new StringBuffer();
		System.out.println(s1.capacity());
		System.out.println(s1.length());
		StringBuffer s2=new StringBuffer("Welcome ");
		System.out.println(s2.capacity());
		System.out.println(s2.charAt(4));
		s2.setCharAt(4,'a');
		System.out.println(s2);
		s2.deleteCharAt(4);
		System.out.println(s2);
		s2.append(" Srinu");
		System.out.println(s2);
		s2.insert(4,"a");
		System.out.println(s2);
		s2.delete(8,13);
		System.out.println(s2);
		s2.append(true);
		System.out.println(s2);
		s2.reverse();
		System.out.println(s2);
	}
}
```

Output:

```
16
0
24
o
Welcame
Welcme
Welcme Srinu
Welcame Srinu
Welcame u
Welcameutrue
eurtuemacleW
```

EXERCISE 3

a) Write a JAVA program to implement class mechanism. Create a class, methods and invoke them inside main method.

Program:

```
class A
{
	int l=10,b=20;
	void display()
	{
		System.out.println(l);
		System.out.println(b);
	}
}
class methoddemo
{
	public static void main(String args[])
	{
		A a1=new A();
		a1.display();
	}
}
```

Output:

```
10
20
```

b) Write a JAVA program to implement constructor.

Program:

```
class Box_Demo
{
        int l,b,area;
        public Box_Demo()
       {
       System.out.println("Default");
       l=b=1;
       }
       public void Cal_Area()
       {
        area=l*b;
       System.out.println("Area is: "+area);
       }
       public static void main(String args[])
       {
              Box_Demo b1=new Box_Demo(5);
              b1.Cal_Area();
       }
}
```

Output:

```
Default
Area is: 1
```

EXERCISE 4

a) Write a JAVA program to implement constructor overloading.

Program:

```
class Box_Demo
{
        int l,b,area;

        public Box_Demo()
       {
       System.out.println("Default");
       l=b=1;
       }
        public Box_Demo(int m)
       {
       System.out.println("SQUARE");
        l=m;
       b=m;
       }
       public void Cal_Area()
       {
        area=l*b;
       System.out.println("Area is: "+area);
       }
       public static void main(String args[])
       {
              Box_Demo b1=new Box_Demo(5);
              b1.Cal_Area();
              Box_Demo b3=new Box_Demo();
              b3.Cal_Area();
       }
}
```

Output:

```
SQUARE
Area is: 25
Default
Area is: 1
```

b) Write a JAVA program implement method overloading.

Program:

```
class Method_Overloading
{
	public void methodOne()
	{
	System.out.println("no argument");
	}
	public void methodOne(int x,int y)
	{
	System.out.println(x+y);
	}
	public void methodOne(int d)
	{
	System.out.println(d);
	}
	 public void methodOne(double d)
	{
	System.out.println(d);
	}
	public static void main(String args[])
	{
		Method_Overloadingmo=new Method_Overloading();
		mo.methodOne();
		mo.methodOne(10);
		mo.methodOne(10,20);
		mo.methodOne(3.14);
	}
}
```

Output:

```
no argument
10
30
3.14
```

EXERCISE 5

a) Write a JAVA program to implement Single Inheritance

Program:

```
class A
{
        int x=10;
        public void showX()
        {
       System.out.println("X = "+x);
        }
}
class B extends A
{
        int y=20;
        public void showY()
        {
       System.out.println("Y = "+y);
        }
}
class SingleLevel
{
        public static void main(String args[])
                {
                A a=new A();
               a.showX();
               System.out.println("=========================");
                B b=new B();
               b.showX();
               b.showY();
        }
}
```

Output:

```
X = 10
=========================
X = 10
Y = 20
```

b) Write a JAVA program to implement multi level Inheritance

Program:

```
class Car{
        public Car()
        {
        System.out.println("Class Car");
        }
        public void vehicleType()
        {
        System.out.println("Vehicle Type: Car");
        }
}
class Maruti extends Car{
        public Maruti()
        {
        System.out.println("Class Maruti");
        }
        public void brand()
        {
        System.out.println("Brand: Maruti");
        }
        public void speed()
        {
        System.out.println("Max: 90Kmph");
        }
}
public class Maruti800 extends Maruti
{
        public Maruti800()
        {
        System.out.println("Maruti Model: 800");
        }
        public void speed()
        {
        System.out.println("Max: 80Kmph");
        }
        public static void main(String args[])
        {
                 Maruti800 obj=new Maruti800();
                obj.vehicleType();
                obj.brand();
                obj.speed();
        }
```

```
}
```

Output:

```
Class Car
Class Maruti
Maruti Model: 800
Vehicle Type: Car
Brand: Maruti
Max: 80Kmph
```

c) Write a java program for abstract class to find areas of different shapes

Program:

```
import java.util.*;
abstract class Shape
{
	Scanner sc=new Scanner(System.in);
	float s1,s2,a;
	final float pi=3.14f;
	public abstract void get_Input();
	public abstract void Cal_Area();
	public void show_Area()
	{
	System.out.println("Area is:"+a);
	}
}
class Rect extends Shape
{
	public void get_Input()
	{
	System.out.println("Enter L and B values");
	s1=sc.nextFloat();
	s2=sc.nextFloat();
	}
	public void Cal_Area()
	{
	a=s1*s2;
	}
}
class Circle extends Shape
{
	public void get_Input()
	{
	System.out.println("Enter radius of the Circle");
	s1=sc.nextFloat();
	}
	public void Cal_Area()
	{
	a=pi*s1*s1;
	}
}
class Mainclass
{
```

```
public static void main(String args[])throws Exception
	{
		Shape s;
		s=new Rect();
		System.out.println("Rectangle:");
		s.get_Input();
		s.Cal_Area();
		s.show_Area();
		System.out.println("=============");
		s=new Circle();
		System.out.println("Circle:");
		s.get_Input();
		s.Cal_Area();
		s.show_Area();
		System.out.println("=============");
	}
}
```

Output:

```
Rectangle:
Enter L and B values
14
16
Area is:224.0
=============
Circle:
Enter radius of the Circle
5.6
Area is:98.4704
```

EXERCISE 6

a) Write a JAVA program give example for “super” keyword.

Program:

```
class A
{
	int x=10;
	public void show()
	{
	System.out.println("A: X = "+x);
	}
}
class B extends A
{
	int x=20;
	public void show()
	{
	super.show();
	System.out.println("B: X = "+x);
	}
}
class C extends B
{
	int x=30;
	public void show()
	{
	super.show();
	System.out.println("C: X = "+x);
	}
}
class MainDemo
{
	public static void main(String args[])
	{
		C c=new C();
		c.show();
	}
}
```

Output:

```
A: X = 10
B: X = 20
C: X = 30
```

b) Write a JAVA program to implement Interface. What kind of Inheritance can be achieved?

Program:

```
interface AnimalEat
{
        void eat();
}
interface AnimalTravel
{
         void travel();
}
class Animal implements AnimalEat, AnimalTravel
{
        public void eat()
        {
        System.out.println("Animal is eating");
        }
        public void travel()
        {
        System.out.println("Animal is travelling");
        }
}
public class Demo
 {
        public static void main(String args[])
        {
         Animal a = new Animal();
        a.eat();
         a.travel();
        }
}
```

Output:

```
Animal is eating
Animal is travelling
```

EXERCISE 7

a) Write a JAVA program that describes exception handling mechanism

Program:

```
class ExceptionDemo
{
	public static void main(String[] args)
	{
		int m, n, o=0;
		try
		{
			m = Integer.parseInt(args[0]);
			n = Integer.parseInt(args[1]);
			o = m/n;
		}
		catch(ArrayIndexOutOfBoundsException ae)
		{
		System.out.println(ae.getMessage());
		}
	}
}
```

Output:

```
D:\>java ExceptionDemo 10
Index 1 out of bounds for length 1
Cleanup code
0
```

b) Write a JAVA program Illustrating Multiple catch clauses

Program:

```
class ExceptionDemo
{
	public static void main(String[] args)
	{
		int m, n, o=0;
		try
		{
		m = Integer.parseInt(args[0]);
		n = Integer.parseInt(args[1]);
		o = m/n;
		}
		catch(ArrayIndexOutOfBoundsException ae)
		{
		System.out.println(ae.getMessage());
		}
		catch(NumberFormatException ne)
		{
		System.out.println(ne.getMessage());
		}
		catch(ArithmeticException are)
		{
		are.printStackTrace();
		}
		catch(Exception e)
		{
		System.out.println(e);
		}
		finally
		{
		System.out.println("Cleanup code");
		System.out.println(o);
		}
	}
}
```

Output:

```
D:\>java ExceptionDemo 10 2
Cleanup code
5
```

```
D:\>java ExceptionDemo 10
Index 1 out of bounds for length 1
Cleanup code
0

D:\>java Exception1 10 0
java.lang.ArithmeticException: / by zero at Exception1.main(Exception1.java:9)
Cleanup code
0

D:\>java Exception1 10 a
For input string: "a"
Cleanup code
0
```

EXERCISE 8

a) Write a JAVA program that implements Runtime polymorphism

Program:

```
abstract class Vehicle
{
        public abstract int get_No_Wheels();
        public abstract int seating_Capacity();
}
class Bike extends Vehicle
{
        public int get_No_Wheels()
        {
        return 2;
        }
        public int seating_Capacity()
        {
        return 2;
        }
}
class Auto extends Vehicle
{
        public int get_No_Wheels()
        {
        return 3;
        }
        public int seating_Capacity()
        {
        return 4;
        }
}
class Car extends Vehicle
{
        public int get_No_Wheels()
        {
        return 4;
        }
        public int seating_Capacity()
        {
        return 5;
        }
```

```
}
class Mainclass
{
	public static void main(String args[])
	{
		Vehicle v;
		int w,c;
		v=new Bike();
		System.out.println("==============");
		System.out.println("Bike:");
		w=v.get_No_Wheels();
		c=v.seating_Capacity();
		System.out.println("No of Wheels: "+w);
		System.out.println("Seating Capacity: "+c);
		System.out.println("==============");
		v=new Auto();
		System.out.println("==============");
		System.out.println("Auto:");
		w=v.get_No_Wheels();
		c=v.seating_Capacity();
		System.out.println("No of Wheels: "+w);
		System.out.println("Seating Capacity: "+c);
		System.out.println("==============");
		v=new Car();
		System.out.println("==============");
		System.out.println("Car:");
		w=v.get_No_Wheels();
		c=v.seating_Capacity();
		System.out.println("No of Wheels: "+w);
		System.out.println("Seating Capacity: "+c);
		System.out.println("==============");
	}
}
```

Output:

```
==============
Bike:
No of Wheels: 2
Seating Capacity: 2
==============
==============
Auto:
No of Wheels: 3
Seating Capacity: 4
```

==========
==========
Car:
No of Wheels: 4
Seating Capacity: 5

b) Write a Case study on run time polymorphism, inheritance that implements in above problem

Program:

```
abstract class Vehicle
{
        public abstract int get_No_Wheels();
        public abstract int seating_Capacity();
}
class Bike extends Vehicle
{
        public int get_No_Wheels()
        {
        return 2;
        }
        public int seating_Capacity()
        {
        return 2;
        }
}
class Auto extends Vehicle
{
        public int get_No_Wheels()
        {
        return 3;
        }
        public int seating_Capacity()
        {
        return 4;
        }
}
class Car extends Vehicle
{
        public int get_No_Wheels()
        {
        return 4;
        }
        public int seating_Capacity()
        {
        return 5;
        }
}
class Mainclass
```

```
{
	public static void main(String args[])
	{
		Vehicle v;
		int w,c;
		v=new Bike();
		System.out.println("==============");
		System.out.println("Bike:");
		w=v.get_No_Wheels();
		c=v.seating_Capacity();
		System.out.println("No of Wheels: "+w);
		System.out.println("Seating Capacity: "+c);
		System.out.println("==============");
		v=new Auto();
		System.out.println("==============");
		System.out.println("Auto:");
		w=v.get_No_Wheels();
		c=v.seating_Capacity();
		System.out.println("No of Wheels: "+w);
		System.out.println("Seating Capacity: "+c);
		System.out.println("==============");
		v=new Car();
		System.out.println("==============");
		System.out.println("Car:");
		w=v.get_No_Wheels();
		c=v.seating_Capacity();
		System.out.println("No of Wheels: "+w);
		System.out.println("Seating Capacity: "+c);
		System.out.println("==============");
	}
}
```

Output:

```
==============
Bike:
No of Wheels: 2
Seating Capacity: 2
==============
==============
Auto:
No of Wheels: 3
Seating Capacity: 4
==============
==============
```

Car:
No of Wheels: 4
Seating Capacity: 5

EXERCISE 9

a) Write a JAVA program for creation of Illustrating throw

Program:

```
class ThrowExcep
{
	static void fun()
	{
		try
		{
			throw new NullPointerException("demo");
		}
		catch(NullPointerException e)
		{
			System.out.println("Caught inside fun().");
			throw e; // rethrowing the exception
		}
	}

	public static void main(String args[])
	{
		try
		{
			fun();
		}
		catch(NullPointerException e)
		{
			System.out.println("Caught in main.");
		}
	}
}
```

Output:

```
Caught inside fun().
Caught in main.
```

b) Write a JAVA program for creation of Illustrating finally

Program:

```
import java.io.*;
class Demo
{
	public static void main(String[] args)
	{
		try {
			System.out.println("inside try block");
			System.out.println(34 / 2);
		}
		catch (ArithmeticException e)
		{
			System.out.println("Arithmetic Exception");
		}
		finally
		{
			System.out.println("finally : i execute always.");

		}
	}
}
```

Output:

```
inside try block
17
finally :i execute always.
```

c) Write a JAVA program for creation of Java Built-in Exceptions

Program:

```
class NegativeValException extends Exception
{
        public NegativeValException(String msg)
        {
        super(msg);
        }
}
class Excep3
{
        public static void main(String args[])
        {
                String name=null;
                int m1=0,m2=0,m3=0;
                try
                {
                name=args[0];
                m1=Integer.parseInt(args[1]);
                m2=Integer.parseInt(args[2]);
                m3=Integer.parseInt(args[3]);
                if(m1<0 || m2<0 || m3<0)
                 throw new NegativeValException("Marks should be greater than 0");
                }
                catch(ArrayIndexOutOfBoundsExceptionaoe)
                {
                System.out.println("Minimum of 4 arguments you need to pass");
                }
                catch(NumberFormatException ne)
                {
                System.out.println("Marks should be Integers only");
                }
                catch(NegativeValExceptionnve)
                {
                System.out.println("Marks should be greater than zero");
                System.exit(0);
                }
                catch(Exception e)
                {
                System.out.println(e);
                }
                System.out.println("Name = "+name);
                System.out.println("Average Marks="+(m1+m2+m3)/3);
```

```
        }
}
```

Output:

```
D:\Practice>java Excep3 Ramesh 45 78 65
Name = Ramesh
Average Marks=62

D:\Practice>java Excep3 Ramesh 45 78
Minimum of 4 arguments you need to pass
Name = Ramesh
Average Marks=41

D:\Practice>java Excep3 Ramesh 45 78 -52
Marks should be greater than zero
```

d)Write a JAVA program for creation of User Defined Exception

Program:

```
class NegativeValException extends Exception
{
        public NegativeValException(String msg)
        {
        super(msg);
        }
}
class Excep3
{
        public static void main(String args[])
        {
                String name=null;
                int m1=0,m2=0,m3=0;
                try
                {
                name=args[0];
                m1=Integer.parseInt(args[1]);
                m2=Integer.parseInt(args[2]);
                m3=Integer.parseInt(args[3]);
                if(m1<0 || m2<0 || m3<0)
                 throw new NegativeValException("Marks should be greater than 0");
                }
                catch(ArrayIndexOutOfBoundsExceptionaoe)
                {
                System.out.println("Minimum of 4 arguments you need to pass");
                }
                catch(NumberFormatException ne)
                {
                System.out.println("Marks should be Integers only");
                }
                catch(NegativeValExceptionnve)
                {
                System.out.println("Marks should be greater than zero");
                System.exit(0);
                }
                catch(Exception e)
                {
                System.out.println(e);
                }
                System.out.println("Name = "+name);
                System.out.println("Average Marks="+(m1+m2+m3)/3);
```

```
        }
}
```

Output:

```
D:\Practice>java Excep3 Ramesh 45 78 65
Name = Ramesh
Average Marks=62

D:\Practice>java Excep3 Ramesh 45 78
Minimum of 4 arguments you need to pass
Name = Ramesh
Average Marks=41

D:\Practice>java Excep3 Ramesh 45 78 -52
Marks should be greater than zero
```

EXERCISE 10

a) Write a JAVA program that creates threads by extending Thread class .First thread display "Good Morning "every 1 sec, the second thread displays "Hello "every 2 seconds and the third display "Welcome" every 3 seconds ,(Repeat the same by implementing Runnable)

Program:

(i)Creating multiple threads using Thread class

```
class A extends Thread
{
	public void run()
	{
		try
		{
			for(int i=1;i<=10;i++)
			{
			sleep(1000);
			System.out.println("good morning");
			}
		}
		catch(Exception e)
		{
		    System.out.println(e);
		}
	}
}
class B extends Thread
{
	public void run()
	{
		try
		{
			for(int j=1;j<=10;j++)
			{
			sleep(2000);
			System.out.println("hello");
			}
		}
		catch(Exception e)
		{
		System.out.println(e);
		}
```

```
        }
}
class C extends Thread
{
        public void run()
        {
                try
                {
                        for(int k=1;k<=10;k++)
                        {
                        sleep(3000);
                        System.out.println("welcome");
                        }
                }
                catch(Exception e)
                {
                System.out.println(e);
                }
        }
}
class threaddemo
{
        public static void main(String args[])
        {
                A a1=new A();
                B b1=new B();
                C c1=new C();
                a1.start();
                b1.start();
                c1.start();
        }
}
```

Output:

```
good morning
hello
good morning
good morning
welcome
hello
good morning
good morning
hello
good morning
```

```
welcome
good morning
hello
good morning
good morning
welcome
hello
good morning
hello
welcome
hello
welcome
hello
hello
welcome
hello
welcome
welcome
welcome
welcome
```

(ii)Creating multiple threads using Runnable interface

```
class A implements Runnable
{
	public void run()
	{
		try
		{
			for(int i=1;i<=10;i++)
			{
			Thread.sleep(1000);
			System.out.println("good morning");
			}
		}
		catch(Exception e)
		{
		System.out.println(e);
		}
	}
}
class B implements Runnable
{
	public void run()
	{
```

```
            try
            {
                for(int j=1;j<=10;j++)
                {
                Thread.sleep(2000);
                System.out.println("hello");
                }
            }
            catch(Exception e)
            {
            System.out.println(e);
            }
        }
}
class C implements Runnable
{
        public void run()
        {
            try
            {
                for(int k=1;k<=10;k++)
                {
                Thread.sleep(3000);
                System.out.println("welcome");
                }
            }
            catch(Exception e)
            {
            System.out.println(e);
            }
        }
}
class runnabledemo
{
        public static void main(String args[])
        {
            A a1=new A();
            B b1=new B();
            C c1=new C();
            Thread t1=new Thread(a1);
            Thread t2=new Thread(b1);
            Thread t3=new Thread(c1);
            t1.start();
            t2.start();
            t3.start();
```

```
        }
}
```

Output:

```
good morning
good morning
hello
good morning
welcome
good morning
hello
good morning
good morning
welcome
hello
good morning
good morning
hello
good morning
welcome
good morning
hello
welcome
hello
hello
welcome
hello
welcome
hello
hello
welcome
welcome
welcome
welcome
```

b) Write a program illustrating isAlive and join ()

Program:

```
class A extends Thread
{
	public void run()
	{
		try
		{
			for(int i=1;i<=10;i++)
			{
			sleep(1000);
			System.out.println("good morning");
			}
		}
		catch(Exception e)
		{
		    System.out.println(e);
		}
	}
}
class B extends Thread
{
	public void run()
	{
		try
		{
			for(int j=1;j<=10;j++)
			{
			sleep(2000);
			System.out.println("hello");
			}
		}
		catch(Exception e)
		{
		System.out.println(e);
		}
	}
}
class C extends Thread
{
	public void run()
	{
		try
```

```
            {
                for(int k=1;k<=10;k++)
                {
                sleep(3000);
                System.out.println("welcome");
                }
            }
            catch(Exception e)
            {
            System.out.println(e);
            }
        }
}
class threaddemo
{
        public static void main(String args[])
        {
            A a1=new A();
            B b1=new B();
            C c1=new C();
            a1.start();
            b1.start();
            c1.start();
            System.out.println(a1.isAlive());
            System.out.println(b1.isAlive());
            System.out.println(c1.isAlive());
            try
            {
                a1.join();
                b1.join();
                c1.join();
            }
            catch(InterruptedException e)
            {
                System.out.println(e);
            }
            System.out.println(a1.isAlive());
            System.out.println(b1.isAlive());
            System.out.println(c1.isAlive());
        }
}
```

Output:

true good morning

true hello
true welcome
good morning hello
good morning hello
hello welcome
good morning hello
welcome welcome
good morning hello
hello hello
good morning welcome
good morning welcome
welcome welcome
hello welcome
good morning false
good morning false
hello false
good morning
welcome

c) Write a Program illustrating Daemon Threads.

Program:

```
class DaemonThreadExample extends Thread
{
	public void run()
	{
		if(Thread.currentThread().isDaemon())
		{
		System.out.println("Daemon thread executing");
		}
		else
		{
		System.out.println("user(normal) thread executing");
		}
 }
	public static void main(String[] args)
	{
		DaemonThreadExample t1=new DaemonThreadExample();
		DaemonThreadExample t2=new DaemonThreadExample();
		t1.setDaemon(true);
		t1.start();
		t2.start();
	}
}
```

Output:

```
Daemon thread executing
user(normal) thread executing
```

EXERCISE 11

a) Write a JAVA program Producer Consumer Problem

Program:

```
import java.util.*;
class Buffer
{
	String data;
	boolean avail=false;
	public synchronized void put(String data)
	{
		while (avail==true)
		{
			try
			{
			wait ();
			}
			catch(InterruptedExceptionie)
			{
			System.out.println(ie);
			}
		}
		this.data=data;
		System.out.println("Produced:"+data);
		avail = true;
		notify();
	}
	public synchronized String get()
	{
		while(avail==false)
		{
			try
			{
			wait ();
			}
			catch (InterruptedExceptionie)
			{
			System.out.println(ie);
			}
		}
		avail = false;
		notify ();
```

```
                return data;
        }
}
class Producer extends Thread
{
        String data;
        Scanner sc=new Scanner(System.in);
        Buffer buf;
        public Producer(Buffer buf)
        {
                super("Producer");
                this.buf=buf;
        }
        public void run ()
        {
                try
                {
                while (true)
                {
                        System.out.println("Enter data");
                        data=sc.nextLine();
                        buf.put(data);
                        Thread.sleep(500);
                }
                }
                catch (InterruptedException e)
                {
                System.out.println(e);
                }
        }
}
class Consumer extends Thread
{
        Buffer buf;
        public Consumer(Buffer buf)
        {
                super ("Consumer");
                this.buf=buf;
        }
        public void run ()
        {
                try
                {
                while (true)
                {
```

```
                System.out.println("Consumed:"+ buf.get());
                Thread.sleep (500);
            }
            }
            catch(InterruptedException e)
            {
            System.out.println(e);
            }
        }
}
class MainDemo
{
        public static void main (String args [])
        {
            Buffer buf = new Buffer ();
            Producer p = new Producer (buf);
            Consumer c = new Consumer (buf);
            p.start();
            c.start();
        }
}
```

Output:

```
Enter data
CSE
Produced:CSE
Consumed:CSE
Enter data
ECE
Produced:ECE
Consumed:ECE
Enter data
EEE
Produced:EEE
Consumed:EEE
Enter data
MECH
Produced:MECH
Consumed:MECH
Enter data
```

b) Write a case study on thread Synchronization after solving the above producer consumer problem

Program:

```
import java.util.*;
class Buffer
{
	String data;
	boolean avail=false;
	public synchronized void put(String data)
	{
		while (avail==true)
		{
			try
			{
			wait ();
			}
			catch(InterruptedExceptionie)
			{
			System.out.println(ie);
			}
		}
		this.data=data;
		System.out.println("Produced:"+data);
		avail = true;
		notify();
	}
	public synchronized String get()
	{
		while(avail==false)
		{
			try
			{
			wait ();
			}
			catch (InterruptedExceptionie)
			{
			System.out.println(ie);
			}
		}
		avail = false;
		notify ();
		return data;
	}
}
class Producer extends Thread
```

```
{
        String data;
        Scanner sc=new Scanner(System.in);
        Buffer buf;
        public Producer(Buffer buf)
        {
                super("Producer");
                this.buf=buf;
        }
        public void run ()
        {
                try
                {
                while (true)
                {
                        System.out.println("Enter data");
                        data=sc.nextLine();
                        buf.put(data);
                        Thread.sleep(500);
                }
                }
                catch (InterruptedException e)
                {
                System.out.println(e);
                }
        }
}
class Consumer extends Thread
{
        Buffer buf;
        public Consumer(Buffer buf)
        {
                super ("Consumer");
                this.buf=buf;
        }
        public void run ()
        {
                try
                {
                while (true)
                {
                        System.out.println("Consumed:"+ buf.get());
                        Thread.sleep (500);
                }
                }
```

```
                catch(InterruptedException e)
                {
                System.out.println(e);
                }
        }
}
class MainDemo
{
        public static void main (String args [])
        {
                Buffer buf = new Buffer ();
                Producer p = new Producer (buf);
                Consumer c = new Consumer (buf);
                p.start();
                c.start();
        }
}
```

Output:

```
Enter data
CSE
Produced:CSE
Consumed:CSE
Enter data
ECE
Produced:ECE
Consumed:ECE
Enter data
EEE
Produced:EEE
Consumed:EEE
Enter data
MECH
Produced:MECH
Consumed:MECH
Enter data
```