LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING
(An Autonomous Institution since 2010)
Accredited by NAAC with 'A' Grade & NBA (Under Tier - I),
An ISO 21001:2018,14001:2015,50001:2018 Certified Institution
Approved by AICTE, New Delhi and Affiliated to JNTUK, Kakinada
L.B. REDDY NAGAR, MYLAVARAM, NTR DIST., A.P.-521 230.
hodcse@lbrce.ac.in, cseoffice@lbrce.ac.in, Phone: 08659-222 933, Fax: 08659-222931
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

OBJECT-ORIENTED PROGRAMMING LAB MANUAL

II B.TECH III SEM

(R20 Regulations)

Estd: 1998

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Course Name & Code : **Object Oriented Programming lab (20CS57)**
L-T-P Structure : **0-0-3** **Credits:** 1.5
Program/Sem/Sec : B.Tech-CSE / III SEM / A,B & C SEC. **A.Y :** 2023-24

PREREQUISITE : **C Programming Language**

Course Educational Objectives:

The objective of the course is to apply the constructs of Java programming language along with built-in facilities to create different applications such as console & graphical user interfaces. They will be applying knowledge of object-oriented programming, collection framework to perform all operations on data.

Course Outcomes (COs): ***At the end of this course, the student will be able to***

- **CO1:** Solve Basic mathematical problems using fundamentals of Java and its object-oriented principles. **(Apply – L3)**
- **CO2:** Implement multithreading and exception handling mechanisms. **(Apply – L3)**
- **CO3:** Develop GUI applications and basic data structures using collection framework. **(Apply – L3)**
- **CO4:** Improve individual / teamwork skills, communication & report writing skills with ethical values.

COURSE ARTICULATION MATRIX (Correlation of Cos & POs, PSOs):

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	-	3	2	-	-	-	-	-	-	-	-	-	3	-	-
CO2	-	3	2	-	-	-	-	-	-	-	-	-	3	-	-
CO3	-	3	3	-	-	-	-	-	-	-	-	-	3	-	-
CO4	-	-	-	-	-	-	-	2	2	2	-	-	-	-	-

Note: Enter Correlation Levels 1 or 2 or 3. If there is no correlation, put '-'
1- Slight (Low), 2 – Moderate (Medium), 3 - Substantial (High).

Vision of the Department

The Computer Science & Engineering aims at providing continuously stimulating educational environment to its students for attaining their professional goals and meet the global challenges.

Mission of the Department

- **DM1:** To develop a strong theoretical and practical background across the computer science discipline with an emphasis on problem solving.
- **DM2:** To inculcate professional behaviour with strong ethical values, leadership qualities, innovative thinking and analytical abilities into the student.
- **DM3:** Expose the students to cutting edge technologies which enhance their employability and knowledge.
- **DM4:** Facilitate the faculty to keep track of latest developments in their research areas and encourage the faculty to foster the healthy interaction with industry.

Program Educational Objectives (PEOs)

- **PEO1:** Pursue higher education, entrepreneurship and research to compete at global level.
- **PEO2:** Design and develop products innovatively in computer science and engineering and in other allied fields.
- **PEO3:** Function effectively as individuals and as members of a team in the conduct of interdisciplinary projects; and even at all the levels with ethics and necessary attitude.
- **PEO4:** Serve ever-changing needs of society with a pragmatic perception.

PROGRAMME OUTCOMES (POs):

PO 1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO 2	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO 3	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO 4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO 5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO 6	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO 7	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO 8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO 9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO 10	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO 11	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO 12	**Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAMME SPECIFIC OUTCOMES (PSOs):

PSO 1	The ability to apply Software Engineering practices and strategies in software project development using open-source programming environment for the success oforganization.
PSO 2	The ability to design and develop computer programs in networking, web applications and IoT as per the society needs.
PSO 3	To inculcate an ability to analyse, design and implement database applications.

Modules

Module 1:

a. Develop a java program to create class, which contains data & methods, create an object to access those members.
b. Develop a java program, which implements all types of java variables (local, class level: static, instance).
c. Develop a java program to calculate the sum of diagonal elements of given n x n matrix.

Module 2:

a. Develop a java program, which contains both static and non-static methods.
b. Develop a java program to find area of geometrical figures using method.
c. Develop a java program to initialize instance variables by using constructors.
d. Develop a java program, which implements constructor overloading by passing different number of parameters of different types.

Module 3:

a. Develop a java program to count the words, characters in the given line of text.
b. Develop a java program for sorting a given list of names in ascending order.
c. Develop a java program that reads a line of integers separated by commas and then displays each integer, find the sum of the integers (using StringTokenizer).
d. Develop a java program to implement multi-level inheritance.

Module 4:

a. Develop a java program to create and access user-defined package.
b. Develop a java program to identify the accessibility of a variable by means of different access specifies within and outside the package.
c. Develop a java program to implement the concept of method overloading.
d. Develop a java program to implement the concept of method overriding.

Module 5:

a. Develop a java program for abstract class to find areas of different shapes.
b. Develop a java program to achieve multiple inheritance using interfaces.
c. Develop a java program to create an interface named Vehicle which contains two abstract methods (Specifications (), Display ()). Provide two classes named Two-wheeler, Four wheeler that is implemented by that interface.

Module 6:

a. Develop a java program that implements a multi-threaded program, which has three threads. First thread generates a random integer for every 1 second, if the generated integer is even the second thread computes the square of the number and print it. If the generated integer is odd the third thread will print the value of cube of the number.
b. Develop a java program to identify the use of synchronized blocks, synchronized methods and static synchronized methods in threads concept.
c. Develop a java program to illustrate the concept of inter thread communication.

Module 7:

a. Develop a java program that creates a user interface to perform integer divisions with possible validations (Divide by Zero, NumberFormatException).
b. Develop a java program to implement mouse events like mouse pressed, mouse released, and mouse moved by means of adapter classes.

Module 8:

a. Develop a java program that works as a simple calculator. Use a Grid Layout to arrange Buttons for digits and for the + - * % operations. Add a text field to display the result. Handle any possible exceptions like divide by zero.
b. Develop a java program to simulate a traffic light, user can select any one of the three buttons with: red, yellow, and green color. On selecting a button, an appropriate message with “Stop” or “Ready” or “Go” should appear with the selected button color.

Module 9:

a. Develop a java program to print the collection data by using the following ways
   i) for loop ii) for-each loop iii) Iterator iv) ListIterator

b. Develop a java program to perform all the operations in Collection interface.

Module 10:

a. Develop a java program to implement and perform all the operations in List, Set Interface.

b. Develop a java program to implement and perform all the operations in Map interface.

Module 1:

a). Develop a java program to create class, which contains data & methods, create an object to access those members.

```
import java.io.*;
import java.util.*;
class Student
{
    int rollno;
    String name;
    void insertRecord(int r, String n)
   {
        rollno=r;
        name=n;
    }
    void displayInformation()
  {
        System.out.println(rollno+" "+name);
   }
    public static void main(String args[])
   {
        Student s1=new Student();
        Student s2=new Student();
        s1.insertRecord(550,"nazma");
        s2.insertRecord(152,"reshma");
        s1.displayInformation();
        s2.displayInformation();
    }
   }
```

Output:

```
F:\java\nazma>javac std.java

F:\java\nazma>java Student
550 nazma
152 reshma

F:\java\nazma>
```

b. Develop a java program, which implements all types of java variables (local, class level: static, instance).

```
import java.io.*;
import java.util.*;
public class Student
   {
      static  int stid=50;
      String stdname;
      void display()
      {
            int stdage=18;
            System.out.println("student id is:"+ stid);
            System.out.println("student name is:"+ stdname);
            System.out.println("student age is:"+ stdage);
      }
      public static void main(String args[])
      {
            Student obj=new Student();
            Scanner s=new Scanner(System.in);
            System.out.println("enter student name is:");
            obj.stdname=s.nextLine();
            obj.display();
      }
}
```

Output:

```
F:\java\nazma>javac student.java

F:\java\nazma>java student
enter student name is:
nazma
student id is:50
student name is:nazma
student age is:18
```

c) Develop a java program to calculate the sum of diagonal elements of given n x n matrix.

```
import java.util.*;
class sumofdiagonalelements
{
	public static void main(String args[])
	{
		Scanner sc = new Scanner(System.in);
		int i,j,row,col,sum=0;
		System.out.println("Enter the number of rows:");
		row = sc.nextInt();
		System.out.println("Enter the number of columns:");
		col = sc.nextInt();
		int[][] mat = new int[row][col];
		System.out.println("Enter the elements of the matrix") ;
		for(i=0;i<row;i++)
		{
			for(j=0;j<col;j++)
			{
				mat[i][j] = sc.nextInt();
			}
		}
		System.out.println("The elements of the matrix") ;
		for(i=0;i<row;i++)
		{
			for(j=0;j<col;j++)
			{
				System.out.print(mat[i][j]+"\t");
			}
			System.out.println("");
		}
		for(i=0;i<row;i++)
		{
			for(j=0;j<col;j++)
			{
				if(i==j)
				{
					sum = sum + mat[i][j];
				}
			}
		}
		System.out.printf("SUM of DIAGONAL elements of the matrix = "+sum) ;
	}
}
```

Output:

```
Enter the number of rows:
3
Enter the number of columns:
3
Enter the elements of the matrix
1
2
3
4
5
6
7
8
9
The elements of the matrix
1       2       3
4       5       6
7       8       9
SUM of DIAGONAL elements of the matrix = 15
```

Module 2:

a) Develop a java program, which contains both static and non-static methods.

```
import java.io.*;
import java.util.*;
class method
{
	static void st_met()
	{
		String name = “Pavani”;
		System.out.println(“Hi “+name);
	}
	void sum(int a, int b)
	{
		int c = a+b;
		System.out.println(“Sum:”+c);
	}
	public static void main(String args[])
	{
		method ob = new method();
		ob.sum(3,4);
		st_met();
	}
}
```

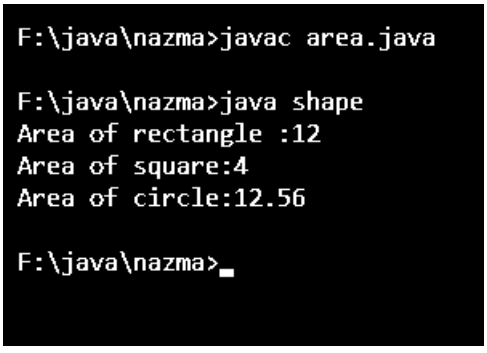
Output:

```
F:\java\nazma>javac Staticnonstatic.java

F:\java\nazma>java method
Sum:7
Hi Pavani

F:\java\nazma>
```

b. Develop a java program to find area of geometrical figures using method.

```
import java.io.*;
import java.util.*;
class shape
{
	int a,l,b,r;
	static int area(int side)
	{
		int a = side*side;
		return a;
	}
	void area(int l, int b)
	{
		a = l*b;
		System.out.println("Area of rectangle :"+a);
	}
	double area(double r)
	{
		double a = 3.14*r*r;
		return a;
	}
	public static void main(String args[])
	{
		shape s = new shape();
		int x = area(2);
		s.area(3,4);
		double y = s.area(2.000);
		System.out.println("Area of square:"+x);
		System.out.println("Area of circle:"+y);
	}

}
```

Output:



c. Develop a java program to initialize instance variables by using constructors.

```
import java.util.*;
import java.io.*;
class ConstructorEx
{
	int rollno;
	String name;
	ConstructorEx()
	{
		rollno=550;
		name="Nazma";
		System.out.println("Rollno of student is :  "+rollno);
		System.out.println("name of student is :  "+name);
	}
	public static void main(String args[])
	{
		ConstructorEx obj=new ConstructorEx();
	}
}
```

Output:

```
F:\java\nazma>javac constructor.java

F:\java\nazma>java constructor
Rollno of student is :  550
name of student is :  Nazma

F:\java\nazma>
```

d. Develop a java program, which implements constructor overloading by passing different number of parameters of different types.

```
import java.io.*;
import java.util.*;
class cons
{
	cons()
	{
		System.out.println(“Hi “);
	}
	cons(int num)
	{
		if(num%2==0)
		{
			System.out.println(num+” is even number”);
		}
		else
		{
			System.out.println(num+” is a add number”);
		}
	}
	cons(int a, int b)
	{
		int sum = a+b;
		System.out.println(“sum:”+sum);
	}
}
class mains
{
	public static void main(String args[])
	{
		cons ob = new cons();
		cons ob1 = new cons(3);
		cons ob2 = new cons(4,5);
	}
}
```

Output:

```
F:\java\nazma>javac consover.java

F:\java\nazma>java mains
Hi
3 is a odd number
sum:9

F:\java\nazma>
```

Module 3:

a. Develop a java program to count the words, characters in the given line of text.

```
import java.util.*;
import java.lang.*;

public class Main
{
	public static void main(String[] args)
	{
		int chcount=0,wcount=0,i,j;
		Scanner s= new Scanner(System.in);
		System.out.println(" enter string" );
		String s1 = s.nextLine();
		System.out.println(   "  length of string is " + s1.length());
		for( i=0; i<s1.length(); i++)
		{
			if( s1.charAt(i)!=' ')
			{
				chcount++;
			}
		}
		System.out.println(" no of characters " + chcount);

		 if( (s1.charAt(0))!=' ')
		 {
			wcount++;
		  }

		for(j=0; j<s1.length(); j++)
		{
			if( (s1.charAt(j) ==' ' )&& (s1.charAt(j+1)!=' '))
			{
				wcount++;
			}
		}
		System.out.println(" no of words" +wcount);
	}
}
```

Output:

```
 enter string
welcome to java
  length of string is 15
 no of characters 13
 no of words3

[Program finished]
```

b. Develop a java program for sorting a given list of names in ascending order.

```
import java.util.*;
import java.lang.*;

public class Main
{
	public static void main(String[] args)
	{
		int n,i,j;
		Scanner s1=new Scanner(System.in);
		System.out.println("enter the no of words");
		n=s1.nextInt();
		String s[] =new String[n];
		System.out.println("enter the  names");
		for(i=0;i<n;i++)
		{
			s[i]= s1.next();
		}
		for(i=0;i<n;i++)
		{
			for(j=i+1;j<n;j++)
			{
				if (s[i].compareTo(s[j])>0)
				{
					String temp =s[i];
					s[i]=s[j];
					s[j]=temp;
				}
			}
		}
		System.out.println("sorted list of names");
		for(i=0;i<n;i++)
		{
			System.out.println(s[i]);
		}
	}
}
```

Output:

```
enter the no of words
5
enter the  names
ramu
venky
suresh
gopi
abdul
sorted list of names
abdul
gopi
ramu
suresh
venky
```

c. Develop a java program that reads a line of integers separated by commas and then displays each integer, find the sum of the integers (using StringTokenizer).

```
import java.util.*;
import java.lang.*;

public class Main
{
	public static void main(String[] args)
	{
		int sum=0;
		Scanner sc=new Scanner(System.in);
		System.out.println("enter the string");

		String s1=sc.nextLine();

		StringTokenizer str= new StringTokenizer(s1,",");
		while(str.hasMoreTokens())
		{
			String s2 =str.nextToken();
			int n= Integer.parseInt(s2);
			sum+=n;
		}
		System.out.println("sum of all intergers is"+sum);
	}
}
```

Output:

```
enter the string
20,87,34,06
sum of all intergers is147
```

d. Develop a java program to implement multi-level inheritance.

```
import java.util.*;
class A
{
	int x,y;
	void sum( int a ,int b)
	{
			x=a;
		y=b;
		System.out.println("sum is "  +(x+y));
	}
 }
class B extends A
{
	int x,y;
	void mult( int a ,int b)
	{
		x=a;
		y=b;
		System.out.println("mult is "  +(x*y));
	}
 }
class C extends B
{
	int x,y;
	void diff( int a ,int b)
	{
		x=a;
		y=b;
		System.out.println("diff is "  +(x-y));
	}
 }
class Main
{
	public static void main( String args[])
	{
		C o = new C();
		o.diff(5,6);
		o.mult(6,8);
		o.sum(9,2);
	}
 }
```

Output:

```
C:\Users\Admin\OneDrive - Lakireddy Bali Reddy College of Engineering\Desktop>java Main
diff is -1
mult is 48
sum is 11
```

Module 4:

a. Develop a java program to create and access user-defined package.

```
package a;
public class Main
{
	int p,q;
	public void display(int x,int y)
	{
		p=x;
		q=y;
		int r=p+q;
		System.out.println("result" +r);
	}
	public static void main(String args[])
	{
		Main obj= new Main();
		obj.display(5,6);
	}
}
```

Output:

```
C:\Users\Admin\OneDrive - Lakireddy Bali Reddy College of Engineering\Desktop>javac Main.java

C:\Users\Admin\OneDrive - Lakireddy Bali Reddy College of Engineering\Desktop>java Main
Error: Could not find or load main class Main
Caused by: java.lang.NoClassDefFoundError: a/Main (wrong name: Main)

C:\Users\Admin\OneDrive - Lakireddy Bali Reddy College of Engineering\Desktop>javac -d . Main.java

C:\Users\Admin\OneDrive - Lakireddy Bali Reddy College of Engineering\Desktop>java a.Main
result:11

C:\Users\Admin\OneDrive - Lakireddy Bali Reddy College of Engineering\Desktop>
```

b. Develop a java program to identify the accessibility of a variable by means of different access specifies within and outside the package.

```
package b;
import a.Main;
class Main1
{
	int m,n;
	void disp(int x,int y)
	{
		m=x;
		n=y;
		int k = m-n;
		System.out.println("differnce is: "+ k);
	}
	public static void main(String args[])
	{
		Main1 obj2= new Main1();
		obj2.disp(7,2);
		Main obj3=new Main();
		obj3.display(9,3);
	}
}
```

Output:

```
C:\Users\Admin\OneDrive - Lakireddy Bali Reddy College of Engineering\Desktop>javac -d . Main1.java

C:\Users\Admin\OneDrive - Lakireddy Bali Reddy College of Engineering\Desktop>java b.Main1
differnce is: 5
result:12

C:\Users\Admin\OneDrive - Lakireddy Bali Reddy College of Engineering\Desktop>
```

c. Develop a java program to implement the concept of method overloading.

```
class Main
{
	static int a;
	static double b;
	void sum(int x,int y)
	{
		a=x+y;
		System.out.println("sum is "+a);
	}
	void sum(double x,double y)
	{
		b=x+y;
		System.out.println("product is "+b);
	}
	public static void main(String args[])
	{
		a obj = new a();
		obj.sum(7,8);
		obj.sum(9.0d,7.0d);
	}
}
```

Output:

```
C:\Users\Admin\Desktop\java>javac over.java

C:\Users\Admin\Desktop\java>java a
sum is 15
product is 16.0
```

d. Develop a java program to implement the concept of method overriding.

```
class A
{ static int a;
  void sum(int x, int y)
  { a=x+y;
    System.out.println(" parent sum is " +a);
  }
}
 class b extends A
 { static int b;
   void sum(int x, int y)
  { b=x+y;
    System.out.println(" child sum is " +a);
  }
  public static void main(String args[])
  { b obj = new b();
    obj.sum(8,9);
  }
```

Output:

```
C:\Users\Admin\Desktop\java>javac override.java
C:\Users\Admin\Desktop\java>java b
 child sum is 0
```

Module 5:

a. Develop a java program for abstract class to find areas of different shapes.

```
abstract class shape
{
        abstract void area();
}
class triangle extends shape
{
        void area()
          { int a=8,b=9;
            System.out.println("area of triangle is " +0.5*a*b) ;
          }
}

class rectangle extends shape
{
        void area()
          { int a=8,b=9;
            System.out.println("area of triangle is " +a*b) ;
          }
}

 class square extends shape
{
        void area()
          { int a=8;
            System.out.println("area of triangle is " +a*a) ;
          }
}
class a
{
public static void main(String args[])
             {
              triangle t = new triangle();
              t.area();
              rectangle r= new rectangle();
              r.area();
              square s= new square();
              s.area();
             }
}
```

Output:

```
avac module5a.java
oem@INDIA:~/pavan$ java a
area of triangle is 36.0
area of triangle is 72
area of triangle is 64
```

b. Develop a java program to achieve multiple inheritance using interfaces.

```
interface a
{ void speed();
}
interface b
{ void distance();
}
class c implements a,b
{ public void speed()
  { System.out.println("go slow");
  }
  public void distance()
  { System.out.println("walk more distance");
  }
}
class A
{ public static void main(String args[])
  { c o =new c ();
    o.speed();
    o.distance();
  }
}
```

Output:

```
em@INDIA:~/pavan$ javac module5b.java
oem@INDIA:~/pavan$ java A
go slow
walk more distance
```

c. Develop a java program to create an interface named Vehicle which contains two abstract methods (Specifications (), Display ()). Provide two classes named Two-wheeler, Fourwheeler that is implemented by that interface.

```
interface vehicle
{ void specification();
  void display();
}
class twowheeler implements vehicle
{ public void specification()
  { System.out.println("herohonda");
  }
  public void display()
  {System.out.println("price is 60000");
  }
}
class fourwheeler implements vehicle
{ public void specification()
  { System.out.println("bmw");
  }
  public void display()
  {System.out.println("price is 160000");
  }
}
class a
{ public static void main(String args[])
  { twowheeler o =new twowheeler();
    o.specification();
    o.display();
    fourwheeler o2 =new fourwheeler();
    o2.specification();
    o2.display();
  }

}
```

Output:

```
oem@INDIA:~/pavan$ javac module5c.java
oem@INDIA:~/pavan$ java a
herohonda
price is 60000
bmw
price is 160000
```

Module 6:

a). Develop a java program that implements a multi-threaded program, which has three threads. First thread generates a random integer for every 1 second, if the generated integer is even the second thread computes the square of the number and print it. If the generated integer is odd the third thread will print the value of cube of the number.

```
import java.util.Random;
class randomnumber extends Thread
{ public void run()
{ Random rand = new Random();
  for(int i= 0; i<4;i++)
  { int randominteger = rand.nextInt(5);
    System.out.println( " random integer generated" + randominteger);
    if( randominteger%2==0)
    { squarethread sqt = new squarethread(randominteger);
    sqt.start();
    }
    else
    { cubethread cut = new cubethread(randominteger);
    cut.start();
    }
    try
    { Thread.sleep(1000);
    }
    catch(InterruptedException e)
    { System.out.println(e);
    }

  }
}
}

 class squarethread extends Thread
 { int number;
    squarethread(int x)
    { number = x;
    }
    public void run()
    { System.out.println(" square of number is " +(number*number));
    }
 }

  class cubethread extends Thread
 { int number;
    cubethread(int x)
    { number = x;
    }
    public void run()
    { System.out.println(" cube of number is " +(number*number* number));
    }
```

```
 }
 class test
 { public static void main(String args[])
   { randomnumber rn = new randomnumber();
   rn.start();
   }
 }
```

Output:

```
random integer generated4
square of number is 16
random integer generated1
cube of number is 1
random integer generated2
square of number is 4
random integer generated0
square of number is 0

[Program finished]
```

6 b) Develop a java program to identify the use of synchronized blocks, synchronized methods and static synchronized methods in threads concept

```
class SharedResource {
    // Synchronized method
    public synchronized void synchronizedMethod(String threadName) {
        System.out.println(threadName + " entered synchronized method.");
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted: " + e.getMessage());
        }
        System.out.println(threadName + " exiting synchronized method.");
    }

    // Method with synchronized block
    public void methodWithSynchronizedBlock(String threadName) {
        System.out.println(threadName + " trying to enter synchronized block.");
        synchronized (this) {
            System.out.println(threadName + " entered synchronized block.");
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                System.out.println("Thread interrupted: " + e.getMessage());
            }
            System.out.println(threadName + " exiting synchronized block.");
        }
    }

    // Static synchronized method
    public static synchronized void staticSynchronizedMethod(String threadName) {
        System.out.println(threadName + " entered static synchronized method.");
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted: " + e.getMessage());
        }
        System.out.println(threadName + " exiting static synchronized method.");
    }
}

class MyThread extends Thread {
    private SharedResource resource;
    private String methodType;

    public MyThread(SharedResource resource, String methodType, String threadName) {
        super(threadName);
        this.resource = resource;
        this.methodType = methodType;
    }

    @Override
    public void run() {
        switch (methodType) {
            case "synchronizedMethod":
                resource.synchronizedMethod(this.getName());
```

```
            break;
        case "synchronizedBlock":
            resource.methodWithSynchronizedBlock(this.getName());
            break;
        case "staticSynchronizedMethod":
            SharedResource.staticSynchronizedMethod(this.getName());
            break;
        default:
            System.out.println("Invalid method type.");
    }
  }
}

public class SynchronizedDemo {
  public static void main(String[] args) {
    SharedResource resource = new SharedResource();

    // Creating threads to demonstrate synchronized methods
    MyThread thread1 = new MyThread(resource, "synchronizedMethod", "Thread-1");
    MyThread thread2 = new MyThread(resource, "synchronizedMethod", "Thread-2");

    // Creating threads to demonstrate synchronized blocks
    MyThread thread3 = new MyThread(resource, "synchronizedBlock", "Thread-3");
    MyThread thread4 = new MyThread(resource, "synchronizedBlock", "Thread-4");

    // Creating threads to demonstrate static synchronized methods
    MyThread thread5 = new MyThread(null, "staticSynchronizedMethod", "Thread-5");
    MyThread thread6 = new MyThread(null, "staticSynchronizedMethod", "Thread-6");

    // Starting threads
    thread1.start();
    thread2.start();
    thread3.start();
    thread4.start();
    thread5.start();
    thread6.start();
  }
}
```

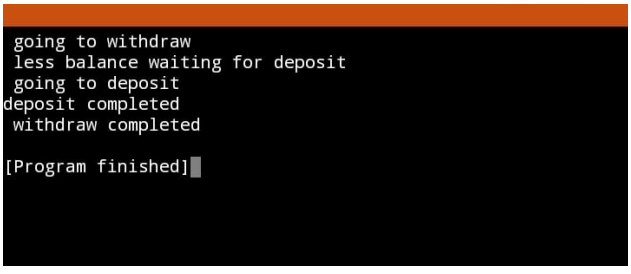
Output:

```
Thread-1 entered synchronized method.
Thread-3 trying to enter synchronized block.
Thread-5 entered static synchronized method.
Thread-2 waiting to enter synchronized method.
Thread-4 trying to enter synchronized block.

Thread-1 exiting synchronized method.
Thread-2 entered synchronized method.
Thread-2 finished execution

Thread-3 entered synchronized block
Thread-4 entered synchronized block
-- completion block
Thread-6 entered static synchronized method
```

6c). Develop a java program to illustrate the concept of inter thread communication.

```
class customer
{ int amount =10000;
  synchronized void withdraw( int amount)
  { System.out.println(" going to withdraw");    if (this.amount< amount)
    { System.out.println(" less balance waiting for deposit");
    try
    { wait();
    }
    catch(InterruptedException e)
    {System.out.println(e) ;
    }
    }
    this.amount-=amount;
    System.out.println(" withdraw completed");
    }
    synchronized void deposit(int amount)
    { System.out.println(" going to deposit");
      this.amount+= amount;
      System.out.println("deposit completed");
      notify();
    }
}
class test
{ public static void main(String args[])
  { customer c = new customer();
    Thread t1 = new Thread() {
            public void run()
            { c.withdraw(15000);
            }
    };
    Thread t2= new Thread() {
            public void run()
            { c.deposit(10000);
            }
    };
     t1.start();
     t2.start();
  }
}
```

Output:



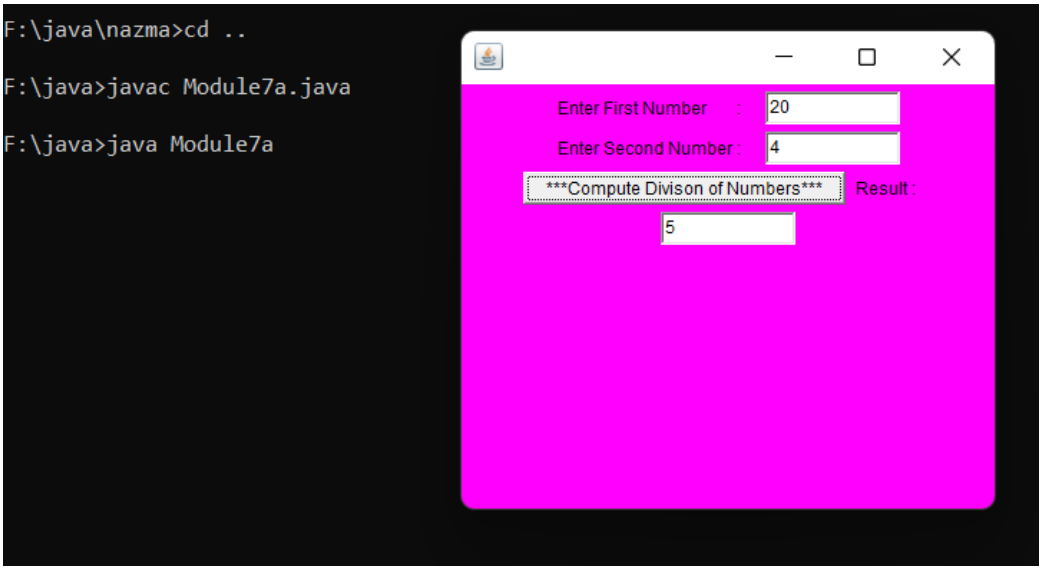
Module 7:

a) Develop a java program that creates a user interface to perform integer divisions with possible validations (Divide by Zero, NumberFormatException).

```
import java.awt.*;
import java.awt.event.*;
public class Module7a extends Frame implements ActionListener,WindowListener
{
        Label l1,l2,l3;
        TextField t1,t2,t3;
        Dialog d;
        Button btn;

        public Module7a()
        {
                setLayout(new FlowLayout(FlowLayout.CENTER));
                setSize(290,290);
                setVisible(true);
              setBackground(Color.MAGENTA);
                addWindowListener(this);
             // setResizable(false);

              l1= new Label("Enter First Number       :");
                add(l1);
                t1=new TextField(10);
                add(t1);

              l2= new Label("Enter Second Number :");
              add(l2);
                t2=new TextField(10);
                add(t2);

              btn=new Button("   **Compute Divison of Numbers**   ");
              add(btn);

                l3= new Label("Result :");
              add(l3);
                t3=new TextField(10);
              add(t3);

                btn.addActionListener(this);
        }
                        public void windowActivated(WindowEvent e) {}
        public void windowIconified(WindowEvent e) {}
        public void windowOpened(WindowEvent e) {}
        public void windowClosing(WindowEvent e) {
        dispose();
        }
        public void windowDeactivated(WindowEvent e) {}
        public void windowDeiconified(WindowEvent e) {}
```

```
	public void windowClosed(WindowEvent e) {}
	public void actionPerformed(ActionEvent ae)
	{
		int num1,num2;
		try
		    {
		    num1 = Integer.parseInt(t1.getText());
		      num2 = Integer.parseInt(t2.getText());
		      t3.setText(num1/num2+"");
		     }
		    catch(ArithmeticException aex)
		    {
			Dia d1=new Dia("Arithmetic Exception");
			d1.setVisible(true);
		    }
		    catch(NumberFormatException nfe)
		    {
			Dia d2=new Dia("Number Format Exception ");
			d2.setVisible(true);
		    }

	  }
		//addWindowListener(this);

	public static void main(String ar[])
	{
		new Module7a();
	}
}
class Dia extends Dialog implements ActionListener
{
	Button cancel;
	Dia(String str)
	{
		super(new Frame(),str,true);
		cancel=new Button("Cancel");
		setLayout(new FlowLayout());
		setSize(300,200);
		add(new Label("Press the Button"));
		add(cancel);
			Label l4=new Label("exception occurs");
			add(l4);
		cancel.addActionListener(this);

	}

	public void actionPerformed(ActionEvent ae)
	{
		setVisible(true);
	      System.exit(0);
	}
```

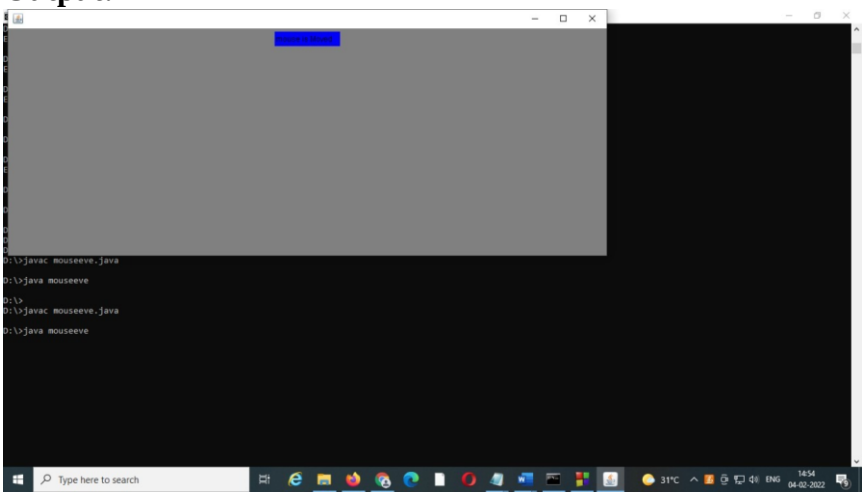
}


F:\java\nazma>cd ..
F:\java>javac Module7a.java
F:\java>java Module7a
Enter First Number :
20
Enter Second Number :
4
Compute Divison of Numbers
Result :
5

b. Develop a java program to implement mouse events like mouse pressed, mouse released, and mouse moved by means of adapter classes.

```
import java.awt.*;
import java.awt.event.*;
public class mouseeve extends Frame
{
	Label l;
	mouseeve()
	{
		setSize(400,400);
		setBackground(Color.gray);
		setVisible(true);

		l=new Label("");
		l.setBackground(Color.blue);
		add(l);
		setLayout(new FlowLayout());
		addMouseListener(new MouseAdapter(){

		public void mousePressed(MouseEvent e)
		{
			l.setText("Mouse is Pressed");
		}
		public void mouseReleased(MouseEvent e)
		{
			l.setText("Mouse is Released");
		}
		});
	addWindowListener(new WindowAdapter(){
	    public void windowClosing(WindowEvent e)
		 {
		    System.exit(0);
		 }
		 });
		addMouseMotionListener(new MouseMotionAdapter()
		{
			public void mouseMoved(MouseEvent e)
			{
				l.setText("mouse is Moved");
			}
		});
	}
		public static void main(String args[])
		{
			new mouseeve();
		}
}
```

Output:


D:\>javac mouseeve.java
D:\>java mouseeve
D:\>
D:\>javac mouseeve.java
D:\>java mouseeve
Type here to search
31°C
ENG
14:54
04-02-2022

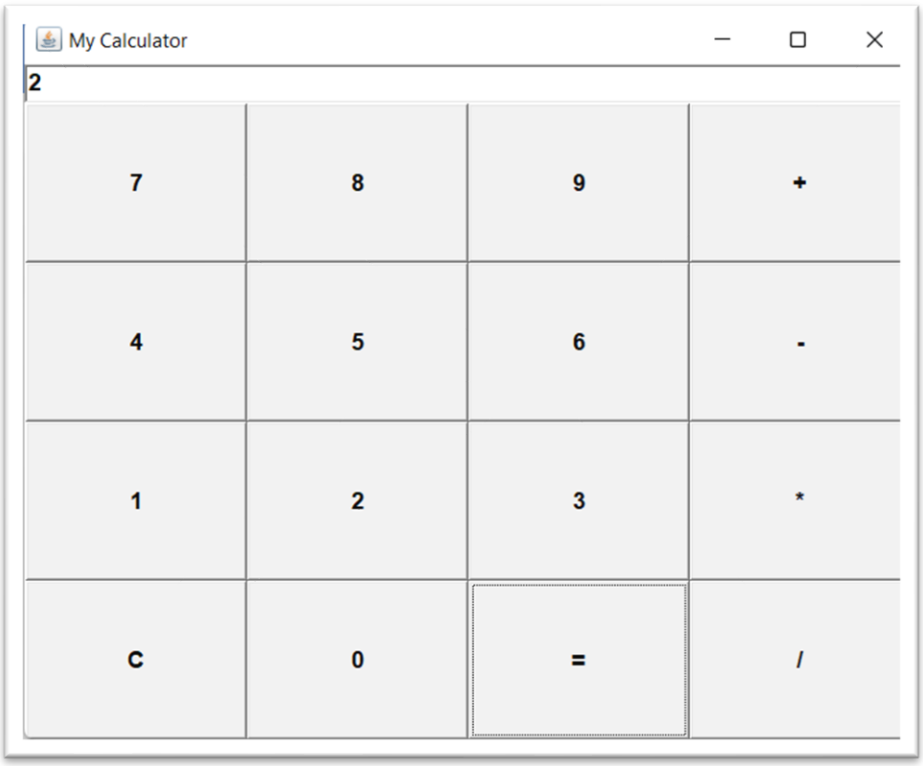
Module 8:

a) Develop a java program that works as a simple calculator. Use a Grid Layout to arrange Buttons for digits and for the + - * % operations. Add a text field to display the result. Handle any possible exceptions like divide by zero.

```
import java.awt.*;
import java.awt.event.*;

class MyCalculator extends Frame implements ActionListener {

 TextField tfInput;
 Panel panel;

 String btnString[] = {"7", "8", "9", "+", "4", "5", "6", "-","1", "2", "3", "*",  "C", "0", "=", "/"};
Button btn[] = new Button[16];
 int num1 = 0, num2 = 0, result = 0;
 char op;

 public MyCalculator() {

  Font f = new Font("Cambria", Font.BOLD, 18);

  tfInput = new TextField(10);
  tfInput.setFont(f);

  panel = new Panel();

  add(tfInput, "North");
  add(panel, "Center");

  panel.setLayout(new GridLayout(4,4));

  for(int i=0; i < 16; i++) {

   btn[i] = new Button(btnString[i]);
   btn[i].setFont(f);
   btn[i].addActionListener(this);
   panel.add(btn[i]);
  }

  addWindowListener(new WindowAdapter(){

   public void windowClosing(WindowEvent we) {
    System.exit(0);
   }
  });
 }

 public void actionPerformed(ActionEvent ae) {

  String str = ae.getActionCommand();

  if(str.equals("+")) {

   op = '+';
```

```
  num1 = Integer.parseInt(tfInput.getText());
  tfInput.setText("");
 }
 else if(str.equals("-")) {
  op = '-';
  num1 = Integer.parseInt(tfInput.getText());
  tfInput.setText("");
 }
 else if(str.equals("*")) {
  op = '*';
  num1 = Integer.parseInt(tfInput.getText());
  tfInput.setText("");
 }
 else if(str.equals("/")) {
  op = '/';
  num1 = Integer.parseInt(tfInput.getText());
  tfInput.setText("");
 }
 else if(str.equals("=")) {

  num2 = Integer.parseInt(tfInput.getText());

  switch(op) {

   case '+' : result = num1 + num2;
    break;
   case '-' : result = num1 - num2;
    break;
   case '*' : result = num1 * num2;
    break;
   case '/' : result = num1 / num2;
    break;
  }
  tfInput.setText(result + "");
  result = 0;
 }
 else if(str.equals("C")) {

  tfInput.setText("");
  num1 = num2 = result = 0;
 }
 else {
  tfInput.setText(tfInput.getText() + str);
 }
}

public static void main(String args[]) {

 MyCalculator m = new MyCalculator();
 m.setTitle("My Calculator");
 m.setSize(250,300);
 m.setVisible(true);
}
}
```

Output:

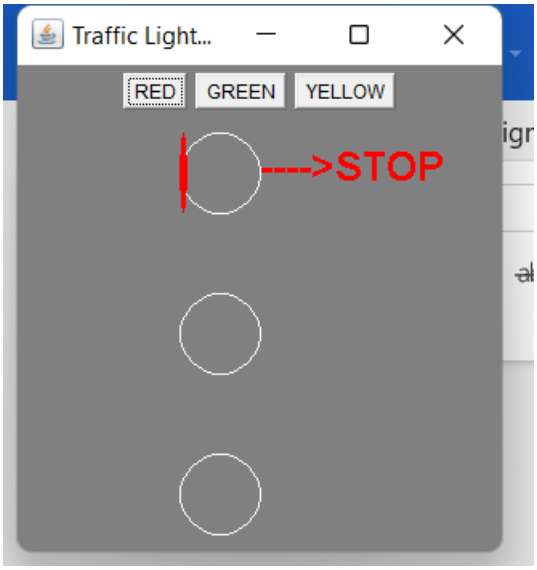


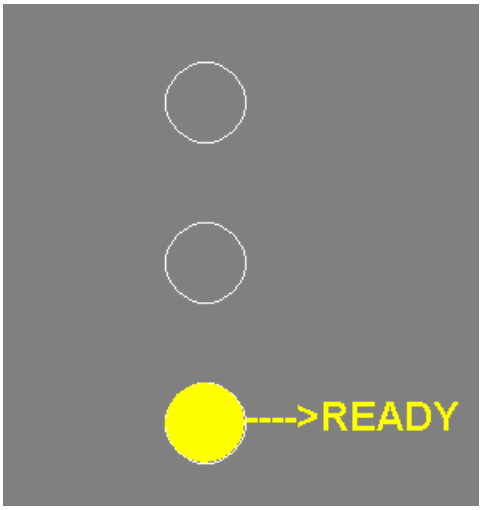
b)Develop a java program to simulate a traffic light, user can select any one of the three buttons with: red, yellow, and green color. On selecting a button, an appropriate message with "Stop" or "Ready" or "Go" should appear with the selected button color.

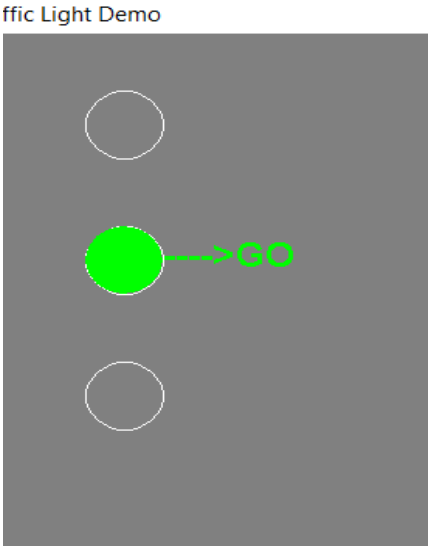
```
import java.awt.*;
import java.awt.event.*;
public class TrafficLightDemo extends Frame
implements ActionListener
{
Button redbt,greenbt,yellowbt;
String msg="";
TrafficLightDemo()
{
setLayout(new FlowLayout());
redbt=new Button("RED");
greenbt=new Button("GREEN");
yellowbt=new Button("YELLOW");
add(redbt);
add(greenbt);
add(yellowbt);
redbt.addActionListener(this);
greenbt.addActionListener(this);
yellowbt.addActionListener(this);
addWindowListener(new WindowAdapter() {
public void windowClosing(WindowEvent we)
{
System.exit(0);
} });
}
public void actionPerformed(ActionEvent ae)
{
msg=ae.getActionCommand();
repaint();
}
public static void main(String args[])
{
TrafficLightDemo tf=new TrafficLightDemo();
tf.setTitle("Traffic Light Demo");
tf.setSize(320,350);
tf.setVisible(true);
}
public void paint(Graphics g) {
setBackground(Color.gray);
setForeground(Color.white);
g.setFont(new Font("Arial",Font.BOLD,25));
g.drawOval(110,80,50,50);
g.drawOval(110,180,50,50);
g.drawOval(110,280,50,50);
if(msg.equals("RED"))
{
g.setColor(Color.red);
g.fillOval(110,80,50,50);
```

```
g.drawString("---->STOP",160,110);
}
else if(msg.equals("GREEN"))
{
g.setColor(Color.green);
g.fillOval(110,180,50,50);
g.drawString("---->GO",160,210);
}
else if(msg.equals("YELLOW"))
{
g.setColor(Color.yellow);
g.fillOval(110,280,50,50);
g.drawString("---->READY",160,310);
}
}
}
```

Output:







9 a) Develop a java program to print the collection data by using the following ways

i) for loop ii) for-each loop iii) Iterator iv) ListIterator

```
import java.util.*;
public class CollectionIterationDemo {
   public static void main(String[] args) {
      // Create a list of integers
      List<Integer> numbers = new ArrayList<>();
      numbers.add(10);
      numbers.add(20);
      numbers.add(30);
      numbers.add(40);
      numbers.add(50);

      // i) Using for loop
      System.out.println("Using for loop:");
      for (int i = 0; i < numbers.size(); i++) {
         System.out.println(numbers.get(i));
      }

      // ii) Using for-each loop
      System.out.println("\nUsing for-each loop:");
      for (Integer number : numbers) {
         System.out.println(number);
      }

      // iii) Using Iterator
      System.out.println("\nUsing Iterator:");
      Iterator<Integer> iterator = numbers.iterator();
      while (iterator.hasNext()) {
         System.out.println(iterator.next());
      }

      // iv) Using ListIterator
      System.out.println("\nUsing ListIterator (Forward direction):");
      ListIterator<Integer> listIterator = numbers.listIterator();
      while (listIterator.hasNext()) {
         System.out.println(listIterator.next());
      }

      System.out.println("\nUsing ListIterator (Backward direction):");
      while (listIterator.hasPrevious()) {
         System.out.println(listIterator.previous());
      }
   }
}
```

Output:

```
Using for loop:
10
20
30
40
50
```

Using for-each loop:
10
20
30
40
50

Using Iterator:
10
20
30
40
50

Using ListIterator (Forward direction):
10
20
30
40
50

Using ListIterator (Backward direction):
50
40
30
20
10

9 b) Develop a java program to perform all the operations in Collection interface.

```
import java.util.*;

public class CollectionDemo {
   public static void main(String[] args) {
      // Create a collection of integers
      Collection<Integer> collection = new ArrayList<>();

      // 1. Add elements
      collection.add(10);
      collection.add(20);
      collection.add(30);
      collection.add(40);
      collection.add(50);
      System.out.println("Elements after adding: " + collection);

      // 2. Remove an element
      collection.remove(30);
      System.out.println("Elements after removing 30: " + collection);

      // 3. Check if an element exists
      System.out.println("Does collection contain 20? " + collection.contains(20));

      // 4. Get the size of the collection
      System.out.println("Size of collection: " + collection.size());

      // 5. Check if collection is empty
      System.out.println("Is collection empty? " + collection.isEmpty());

      // 6. Iterate using for-each loop
      System.out.println("\nIterating using for-each loop:");
      for (Integer number : collection) {
         System.out.println(number);
      }

      // 7. Iterate using Iterator
      System.out.println("\nIterating using Iterator:");
      Iterator<Integer> iterator = collection.iterator();
      while (iterator.hasNext()) {
         System.out.println(iterator.next());
      }

      // 8. Clear the collection
      collection.clear();
      System.out.println("\nElements after clearing: " + collection);

      // 9. Check if collection is empty after clearing
      System.out.println("Is collection empty after clearing? " + collection.isEmpty());
   }
}
```

Output:

```
Elements after adding: [10, 20, 30, 40, 50]
Elements after removing 30: [10, 20, 40, 50]
Does collection contain 20? true
Size of collection: 4
Is collection empty? false
Iterating using for-each loop:
10
20
40
50
Iterating using Iterator:
10
20
40
50
Elements after clearing: []
Is collection empty after clearing? True
```

10 a) Develop a java program to implement and perform all the operations in List, Set Interface.

```
import java.util.*;

public class CollectionDemo {
   public static void main(String[] args) {
      // List Interface Implementation
      List<Integer> list = new ArrayList<>();
      // 1. Add elements to List
      list.add(10);
      list.add(20);
      list.add(30);
      list.add(40);
      list.add(50);
      System.out.println("Elements in List after adding: " + list);

      // 2. Remove an element from List
      list.remove(Integer.valueOf(30));
      System.out.println("Elements in List after removing 30: " + list);

      // 3. Access elements in List
      System.out.println("Element at index 2: " + list.get(2));

      // 4. Check if an element exists in List
      System.out.println("Does List contain 20? " + list.contains(20));

      // 5. Get the size of the List
      System.out.println("Size of List: " + list.size());

      // 6. Iterate over List using for-each loop
      System.out.println("\nIterating over List using for-each loop:");
      for (Integer number : list) {
         System.out.println(number);
      }

      // 7. Iterate over List using Iterator
      System.out.println("\nIterating over List using Iterator:");
      Iterator<Integer> listIterator = list.iterator();
      while (listIterator.hasNext()) {
         System.out.println(listIterator.next());
      }

      // 8. Clear the List
      list.clear();
      System.out.println("\nElements in List after clearing: " + list);

      // 9. Check if List is empty
      System.out.println("Is List empty after clearing? " + list.isEmpty());

      // Set Interface Implementation
      Set<Integer> set = new HashSet<>();
      // 1. Add elements to Set
      set.add(10);
      set.add(20);
      set.add(30);
```

```
        set.add(40);
        set.add(50);
        set.add(10); // Duplicate element
        System.out.println("\nElements in Set after adding (duplicates not allowed): " + set);

        // 2. Remove an element from Set
        set.remove(30);
        System.out.println("Elements in Set after removing 30: " + set);

        // 3. Check if an element exists in Set
        System.out.println("Does Set contain 20? " + set.contains(20));

        // 4. Get the size of the Set
        System.out.println("Size of Set: " + set.size());

        // 5. Iterate over Set using for-each loop
        System.out.println("\nIterating over Set using for-each loop:");
        for (Integer number : set) {
            System.out.println(number);
        }

        // 6. Iterate over Set using Iterator
        System.out.println("\nIterating over Set using Iterator:");
        Iterator<Integer> setIterator = set.iterator();
        while (setIterator.hasNext()) {
            System.out.println(setIterator.next());
        }

        // 7. Clear the Set
        set.clear();
        System.out.println("\nElements in Set after clearing: " + set);

        // 8. Check if Set is empty
        System.out.println("Is Set empty after clearing? " + set.isEmpty());
    }
}
```

Output:

```
Elements in List after adding: [10, 20, 30, 40, 50]
Elements in List after removing 30: [10, 20, 40, 50]
Element at index 2: 40
Does List contain 20? true
Size of List: 4
Iterating over List using for-each loop:
10
20
40
50
Iterating over List using Iterator:
10
20
40
50
```

Elements in List after clearing: []
Is List empty after clearing? true
Elements in Set after adding (duplicates not allowed): [50, 20, 40, 10, 30]
Elements in Set after removing 30: [50, 20, 40, 10]
Does Set contain 20? true
Size of Set: 4
Iterating over Set using for-each loop:
50
20
40
10
Iterating over Set using Iterator:
50
20
40
10
Elements in Set after clearing: []
Is Set empty after clearing? True

10 b) Develop a java program to implement and perform all the operations in Map interface.

```
import java.util.*;

public class MapDemo {
   public static void main(String[] args) {
      // Create a HashMap
      Map<Integer, String> map = new HashMap<>();

      // 1. Add elements to the Map
      map.put(1, "Apple");
      map.put(2, "Banana");
      map.put(3, "Cherry");
      map.put(4, "Date");
      map.put(5, "Elderberry");
      System.out.println("Map after adding elements: " + map);

      // 2. Remove an element from the Map
      map.remove(3); // Removes entry with key 3
      System.out.println("Map after removing key 3: " + map);

      // 3. Access an element by key
      String value = map.get(2); // Retrieves the value associated with key 2
      System.out.println("Value for key 2: " + value);

      // 4. Update an existing key-value pair
      map.put(2, "Blueberry"); // Updates the value for key 2
      System.out.println("Map after updating key 2: " + map);

      // 5. Check if a key exists in the Map
      boolean containsKey = map.containsKey(4);
      System.out.println("Does Map contain key 4? " + containsKey);

      // 6. Check if a value exists in the Map
      boolean containsValue = map.containsValue("Elderberry");
      System.out.println("Does Map contain value 'Elderberry'? " + containsValue);

      // 7. Get the size of the Map
      int size = map.size();
      System.out.println("Size of the Map: " + size);

      // 8. Iterate over Map entries using for-each loop
      System.out.println("\nIterating over Map entries:");
      for (Map.Entry<Integer, String> entry : map.entrySet()) {
         System.out.println("Key: " + entry.getKey() + ", Value: " + entry.getValue());
      }

      // 9. Iterate over keys only
      System.out.println("\nIterating over keys:");
      for (Integer key : map.keySet()) {
         System.out.println("Key: " + key);
      }

      // 10. Iterate over values only
      System.out.println("\nIterating over values:");
```

```
        for (String val : map.values()) {
            System.out.println("Value: " + val);
        }

        // 11. Clear all elements in the Map
        map.clear();
        System.out.println("\nMap after clearing all elements: " + map);

        // 12. Check if Map is empty
        boolean isEmpty = map.isEmpty();
        System.out.println("Is Map empty? " + isEmpty);
    }
}
```

Output:

```
Map after adding elements: {1=Apple, 2=Banana, 3=Cherry, 4=Date, 5=Elderberry}
Map after removing key 3: {1=Apple, 2=Banana, 4=Date, 5=Elderberry}
Value for key 2: Banana
Map after updating key 2: {1=Apple, 2=Blueberry, 4=Date, 5=Elderberry}
Does Map contain key 4? true
Does Map contain value 'Elderberry'? true
Size of the Map: 4

Iterating over Map entries:
Key: 1, Value: Apple
Key: 2, Value: Blueberry
Key: 4, Value: Date
Key: 5, Value: Elderberry

Iterating over keys:
Key: 1
Key: 2
Key: 4
Key: 5

Iterating over values:
Value: Apple
Value: Blueberry
Value: Date
Value: Elderberry

Map after clearing all elements: {}
Is Map empty? True
```

********** **The End** **********