Bachelor of Technology (B.Tech)

Department of Computer Science and Engineering

II year I sem- Object Oriented Programming through Java Laboratory Manual

SIDDHARTHA INSTITUTE OF TECHNOLOGY & SCIENCES

(Approved by AICTE, New Delhi & Affiliated to JNTUH,
Hyderabad)Accredited by NBA and NAAC with 'A+' Grade
Narapally, Korremula Road, Ghatkesar, Medchal- Malkajgiri (Dist)-501 301

Vision of the Institute

To be a reputed institute in technical education towards research, industrial and societal needs.

Mission of the Institute

Mission	Statement
IM_1	Provide state-of-the-art infrastructure, review, innovative and experiment teaching –learning methodologies.
IM_2	Promote training, research and consultancy through an integrated institute industry symbiosis
IM_3	Involve in activities to groom professional, ethical values and social responsibility

SIDDHARTHA INSTITUTE OF TECHNOLOGY AND SCIENCES

(*Approved by AICTE, New Delhi &Affiliated to JNTUH, Hyderabad*)
Narapally, Telangana – 500 088.

Department of Computer Science and Engineering

Vision of the Department

To be a recognized center of Computer Science education with values, and quality research

Mission of the Department

Mission	Statement
DM_1	Impart high quality professional training with an emphasis on basic principles of Computer Science and allied Engineering
DM_2	Imbibe social awareness and responsibility to serve the society
DM_3	Provide academic facilities, organize collaborated activities to enable overall development of stakeholders

SIDDHARTHA INSTITUTE OF TECHNOLOGY AND SCIENCES

(*Approved by AICTE, New Delhi &Affiliated to JNTUH, Hyderabad*)
Narapally, Telangana – 500 088.

Department of Computer Science and Engineering

Program Educational Objectives (PEOs)

PEO's	Statement
PEO1	Graduates will be able to solve Computer Science and allied Engineering problems, develop proficiency in computational tools.
PEO2	Graduates will be able to communicate and work efficiently in Multidisciplinary teams with a sense of professional and social responsibility.
PEO3	Graduates will be able to exhibit lifelong learning ability andpursue career as architects, software developers and entrepreneurs.

Programme Outcomes

PO1	**Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**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.
PO3	**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.
PO4	**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.
PO5	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
PO6	**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.
PO7	**Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental context, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team network:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**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.
PO12	**Life-Long learning:** Recognize the need for, and have the preparation and able to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes:

PSO1	**Program Applications:** Able to develop programs modules for cloud based applications.
PSO2	**Development Tools:** Able to use tools such as Weka, Rational Rose Raspberry-Pi, Sql and advanced tools

SIDDHARTHA INSTITUTE OF TECHNOLOGY & SCIENCES

(UGC - AUTONOMOUS)

2230579: OBJECT ORIENTED PROGRAMMING THROUGH JAVA LAB

(Common to CSE, DS, SE, CS)

B.Tech. II Year. I Sem

L	T	P	C
0	0	3	1.5

Course Objectives:

- To write programs using abstract classes.
- To write programs for solving real world problems using the java collection framework.
- To write multithreaded programs.
- To write GUI programs using swing controls in Java.
- To introduce java compiler and eclipse platform.
- To impart hands-on experience with java programming.

Course Outcomes:

- Able to write programs for solving real world problems using the java collection framework.
- Able to write programs using abstract classes.
- Able to write multithreaded programs.
- Able to write GUI programs using swing controls in Java.
- Able to write files program.

Note:

1. Use LINUX and MySQL for the Lab Experiments. Though not mandatory, encourage the useof the Eclipse platform.
2. The list suggests the minimum program set. Hence, the concerned staff is requested to addmore problems to the list as needed.

List of Experiments:

1. Write a Java program that checks whether a given string is a palindrome or not.

2. Write a Java program to find the Fibonacci value of a given number

3. Write a Java program to multiply two given matrices

4. Use Eclipse or Net bean platform and acquaint yourself with the various menus. Create a test project,add a test class, and run it. See how you can use auto suggestions, auto fill. Try code formatter and code refactoring like renaming variables, methods, and classes. Try debug step by step with a small program of about 10 to 15 lines which contains at least one if else condition and a for loop.

5. Write a Java program that works as a simple calculator. Use a grid layout to arrange

buttons for thedigits and for the +, -,*, % operations. Add a text field to display the result. Handle any possible exceptions like divided by zero.

6. A) Develop an applet in Java that displays a simple message.
B) Develop an applet in Java that receives an integer in one text field, and computes its factorialValue and returns it in another text field, when the button named "Compute" is clicked.

7. Write a Java program that creates a user interface to perform integer divisions. The user enters twonumbers in the text fields, Num1 and Num2. The division of Num1 and Num 2 is displayed in the Resultfield when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throwa Number Format Exception. If Num2 were Zero, the program would throw an Arithmetic Exception. Display the exception in a message dialog box.

8. Write a Java program that implements a multi-thread application that has three threads. First threadgenerates a random integer every 1 second and if the value is even, the second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of the cube of the number.

9. Write a Java program for the following:
Create a doubly linked list of elements.
Delete a given element from the above list.
Display the contents of the list after deletion.

10. Write a Java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green with radio buttons. On selecting a button, an appropriate message with "Stop" or "Ready" or "Go" should appear above the buttons in the selected color. Initially, there is no message shown.

11. Write a Java program to create an abstract class named Shape that contains two integers and an empty method named print Area (). Provide three classes named Rectangle, Triangle, and Circle suchthat each one of the classes extends the class Shape. Each one of the classes contains only the methodprint Area () that prints the area of the given shape.

12. Suppose that a table named Table.txt is stored in a text file. The first line in the file is the header, andthe remaining lines correspond to rows in the table. The elements are separated by commas. Write a java program to display the table using Labels in Grid Layout.

13. Write a Java program that handles all mouse events and shows the event name at the center of thewindow when a mouse event is fired (Use Adapter classes).

14. Write a Java program that loads names and phone numbers from a text file where the data is organized as one line per record and each field in a record are separated by a tab (\t). It takes a name or phone number as input and prints the corresponding other value

from the hash table (hint:use hash tables).

15. Write a Java program that correctly implements the producer – consumer problem using theconcept of inter thread communication.

16. Write a Java program to list all the files in a directory including the files present in all itssubdirectories.

REFERENCE BOOKS:

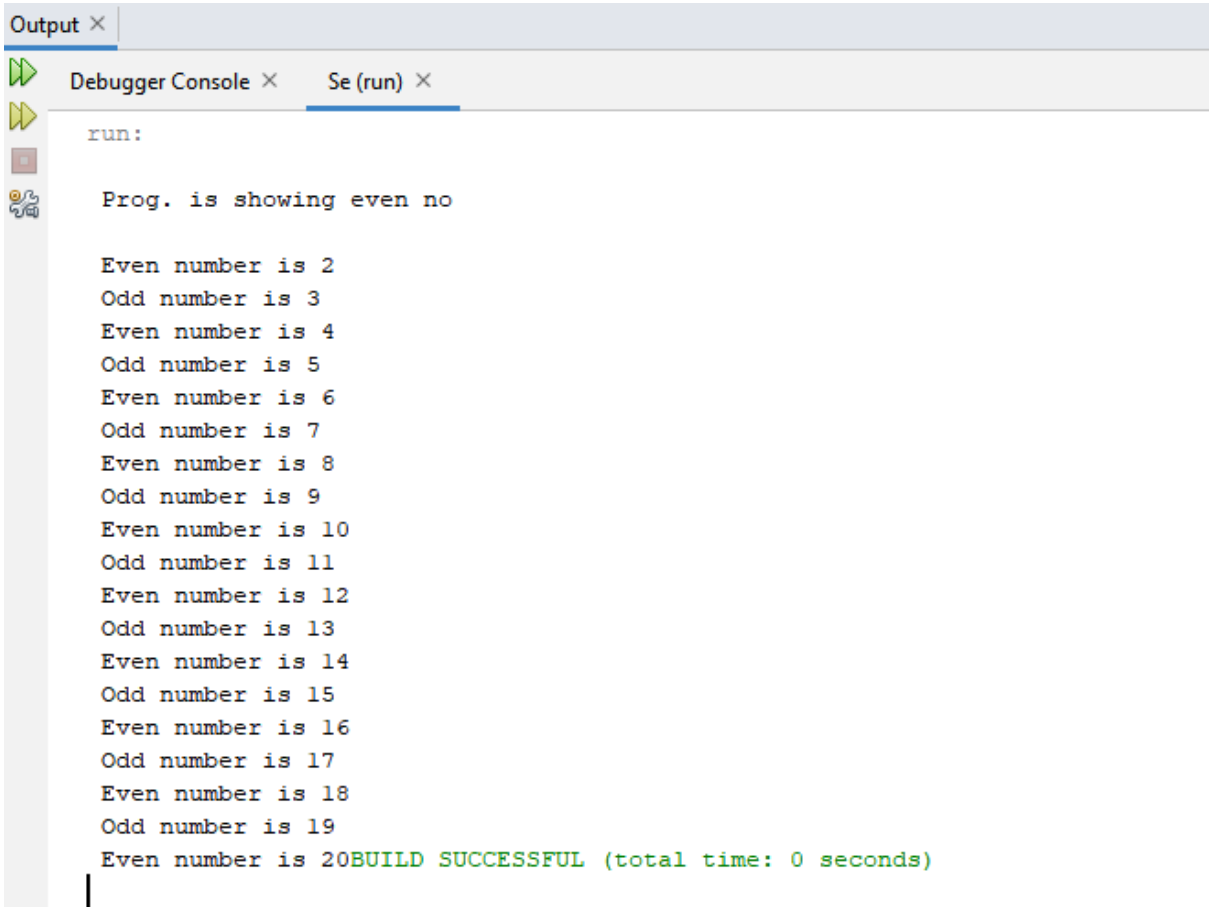
1. Java for Programmers, P. J. Deitel and H. M. Deitel, 10th Edition Pearson education.
2. Thinking in Java, Bruce Eckel, Pearson Education.
3. Java Programming, D. S. Malik and P. S. Nair, Cengage Learning.
4. Core Java, Volume 1, 9th edition, Cay S. Horstmann and G Cornell, Pearson.
5. Java the complete reference, 7th edition, Herbert schildt, TMH.

1. Use eclipse or Netbean platform and acquaint with the various menus, create a test project, add a test class and run it see how you can use auto suggestions, auto fill. Try code formatter and code refactoring like renaming variables, methods and classes. Try debug step by step with a small program of about 10 to 15 lines which contains at least one if else condition and a for loop.

Program:

```
Package  Cse;


public class Cse
{
public static void main(String[ ] args)
{
System.out.println("\n Prog. is showing even no and odd no");
for(int i=2;i<=20;i++)
{
if(i%2==0)
{
        System.out.print("\n Even number is "+i);
}
else{
System.out.print("\n Odd number is "+i);
}
}
}
}
```

Output:-



2. Write a Java program that works as a simple calculator. Use a grid layout to arrange buttons for the digits and for the +, -,*, % operations. Add a text field to display the result. Handle any possible exceptions like divide by zero.

Program:

```
import java.awt.*;

import java.awt.event.*;

public class Calculator implements ActionListener

{

int c,n;

String s1,s2,s3,s4,s5;

Frame f;

Button b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,b14,b15,b16,b17;
```

```
Panel p;
TextField    tf;
GridLayout   g;
Calculator()
{
f  = newFrame("My calculator");
p = newPanel();
f.setLayout(newFlowLayout());
b1 = newButton("0");
b1.addActionListener(this);

b2 = newButton("1");
b2.addActionListener(this);

b3 = newButton("2");
b3.addActionListener(this);

b4 = newButton("3");
b4.addActionListener(this);

b5 = newButton("4");
b5.addActionListener(this);

b6 = newButton("5");
b6.addActionListener(this);

b7 = newButton("6");
b7.addActionListener(this);

b8 = newButton("7");
b8.addActionListener(this);
```

```
b9 = newButton("8");
b9.addActionListener(this);
b10 = newButton("9");
b10.addActionListener(this);
b11 = newButton("+");
b11.addActionListener(this);
b12 = newButton("-");
b12.addActionListener(this);
b13 = newButton("*");
b13.addActionListener(this);
b14 = newButton("/");
b14.addActionListener(this);
b15 = newButton("%");
b15.addActionListener(this);
b16 = newButton("=");
b16.addActionListener(this);
b17 = new Button("C");
b17.addActionListener(this);
tf = newTextField(20);

f.add(tf);
g = newGridLayout(4,4,10,20);
p.setLayout(g);
p.add(b1);
p.add(b2);
p.add(b3);
p.add(b4);
```

```
p.add(b5);
p.add(b6);
p.add(b7);
p.add( b8);
p.add(b9);
p.add(b10);
p.add(b11);
p.add(b12);
p.add(b13);
p.add(b14);
p.add(b15);
p.add(b1 6);
p.add(b17);
f.add(p);
f.setSize(300,300);
f.setVisible(true);
}
public void actionPerformed(ActionEvent e)
{
if(e.getSource()==b1)
{
s3 = tf.getText();
s4 = "0";
s5 = s3+s4;
tf.setText(s5);
}
if(e.getSource()==b2)
{
```

```
s3 = tf.getText();
s4 = "1";
s5 = s3+s4;
tf.setText(s5);
}
if(e.getSource()==b3)
{
s3 = tf.getText();
s4 = "2";
s5 = s3+s4;
tf.setText(s5);
}if(e.getSource()==b4)
{
s3 = tf.getText();
s4 = "3";
s5 = s3+s4;
tf.setText(s5);
}
if(e.getSource()==b5)
{
        s3 = tf.getText();
s4 = "4";
s5 = s3+s4;
tf.setText(s5);
}
if(e.getSource()==b6)
{
```

```
s3 = tf.getText();
s4 = "5";
s5 = s3+s4;
tf.setText(s5);
}
if(e.getSource()==b7)
{
s3 = tf.getText();
s4 = "6";
s5 = s3+s4;
tf.setText(s5);
}
if(e.getSource()==b8)
{
s3 = tf.getText();
s4 = "7";
s5 = s3+s4;
tf.setText(s5);
}
if(e.getSource()==b9)
{
s3 = tf.getText();
s4 = "8";
s5 = s3+s4;
tf.setText(s5);
}
```

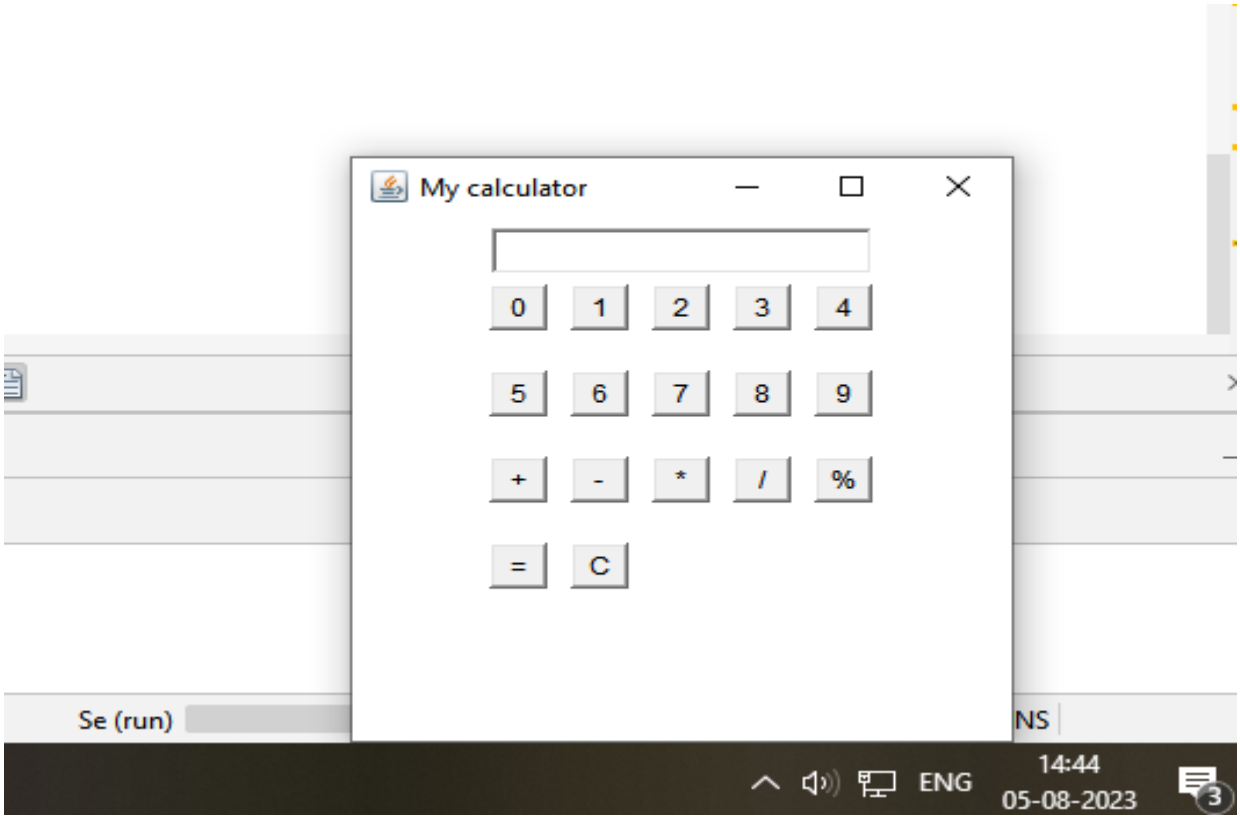
```
if(e.getSource()==b10)
{
s3 = tf.getText();
s4 = "9";
s5 = s3+s4;
tf.setText(s5);
}
if(e.getSource()==b11)
{
s1 = tf.getText();
tf.setText("");

c=1;

}
if(e.getSource()==b12)
{
        s1 = tf.getText();
tf.setText("");
c=2;
}
if(e.getSource()==b13)
{
s1 = tf.getText();
tf.setText("");
c=3;
}
if(e.getSource()==b14)
```

```
{
s1 = tf.getText();
tf.setText("");
c=4;
}
if(e.getSource()==b15)
{
s1 = tf.getText();
tf.setText("");
c=5;
}
if(e.getSource()==b16)
{
s2 = tf.getText();

if(c==1)

{
n = Integer.parseInt(s1)+Integer.parseInt(s2);
tf.setText(String.valueOf(n));
}
else if(c==2)
{
n = Integer.parseInt(s1)-Integer.parseInt(s2);
tf.setText(String.valueOf(n));
}
else if(c==3)
{
```

```
n = Integer.parseInt(s1)*Integer.parseInt(s2);
tf.setText(String.valueOf(n));
}
if(c==4)
{
try
{
int p=Integer.parseInt(s2);
if(p!=0)
{
n = Integer.parseInt(s1)/Integer.parseInt(s2);
tf.setText(String.valueOf(n));
}
elsetf.setText("infinite");
}
catch(Exception i){}
}
if(c==5)
{
n = Integer.parseInt(s1)%Integer.parseInt(s2);
tf.setText(String.valueOf(n));
}
}
if(e.getSource()==b17)
{
tf.setText("");
}
```

```
}
public static  void main(String[ ] args)
{
Calculator v = new Calculator();
}
}
```

Output:-

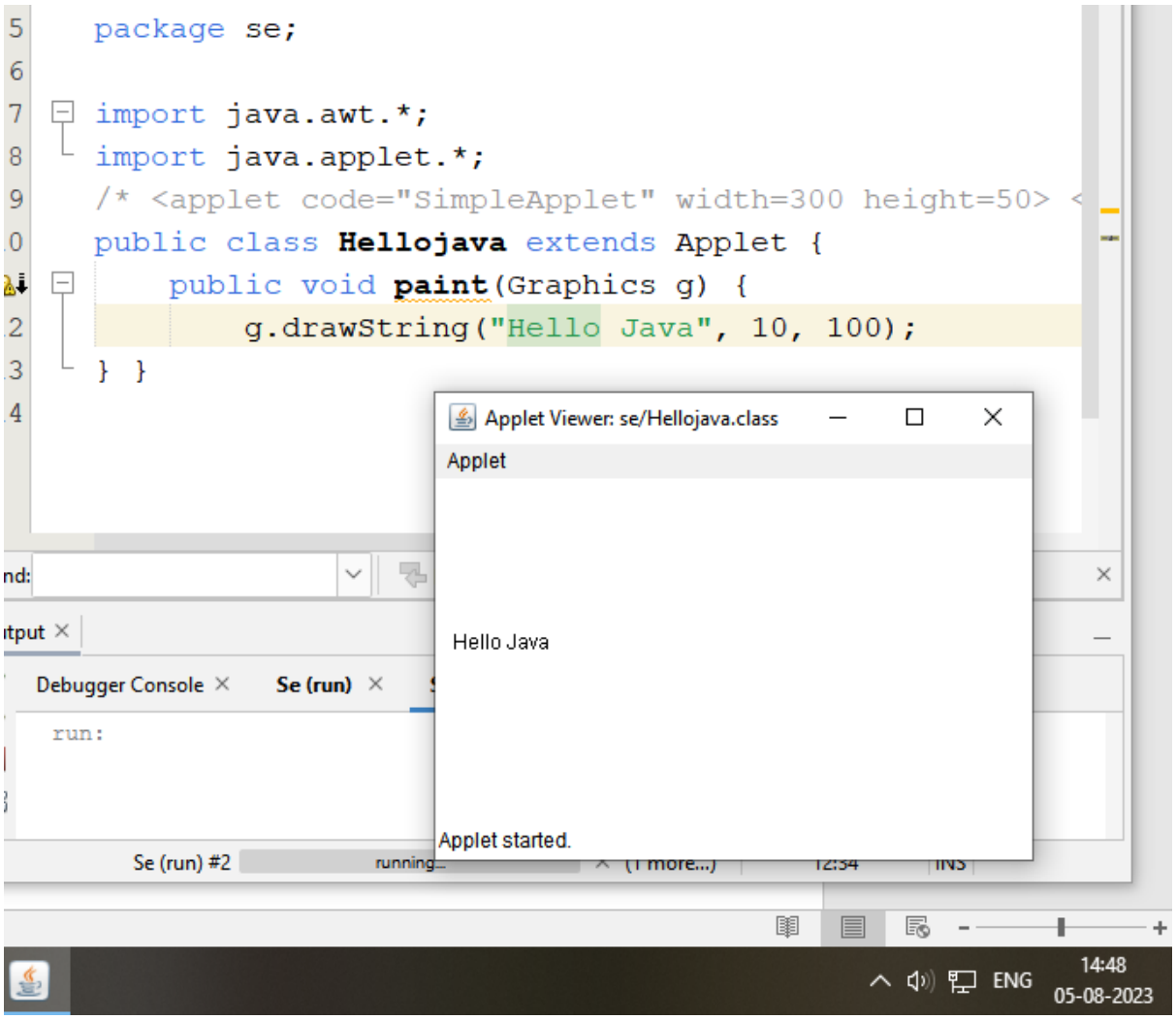


3) a) Develop an applet that displays a simple message.

Program:-

```
import java.awt.*;
import java.applet.*;
/* <applet code="SimpleApplet" width=300 height=50></applet> */
public class HelloJava extends Applet {
public void Paint(Graphics g) {
g.drawString("Hello Java", 10, 100);
}}
```

Output:

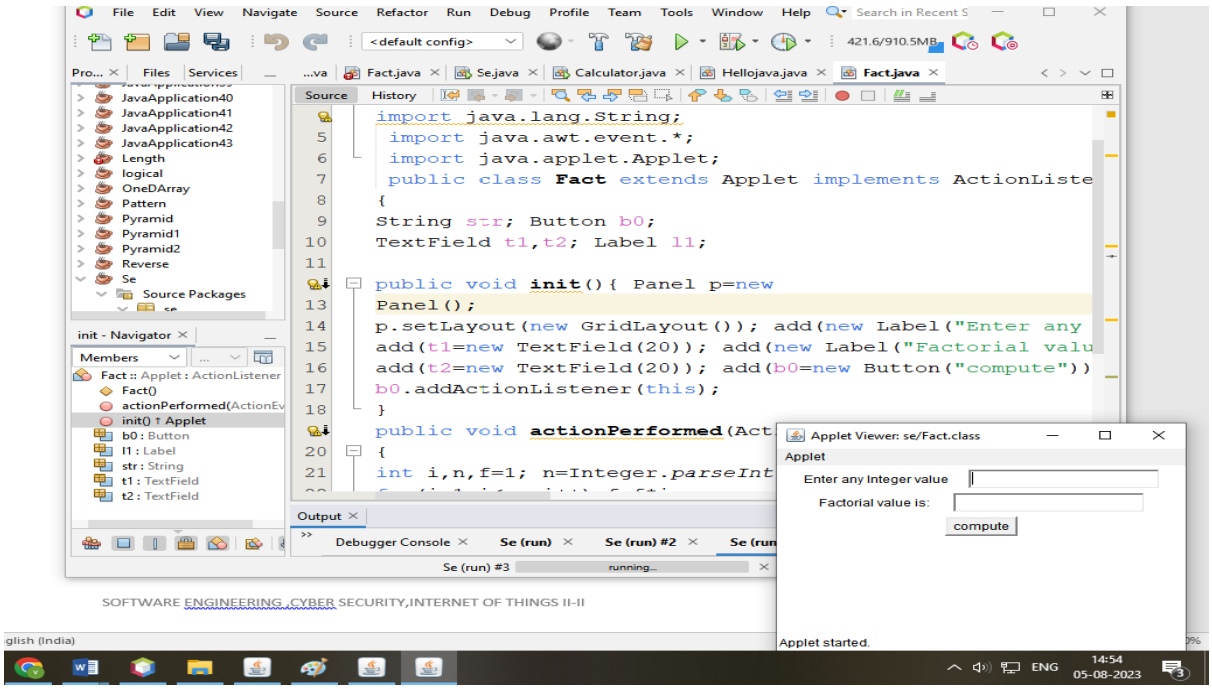

package se;
import java.awt.*;
import java.applet.*;
/* <applet code="SimpleApplet" width=300 height=50>
public class Hellojava extends Applet {
public void paint(Graphics g) {
g.drawString("Hello Java", 10, 100);
} }
Applet Viewer: se/Hellojava.class
Applet
Hello Java
Applet started.
Debugger Console
Se (run)
run:
Se (run) #2
ENG
14:48
05-08-2023

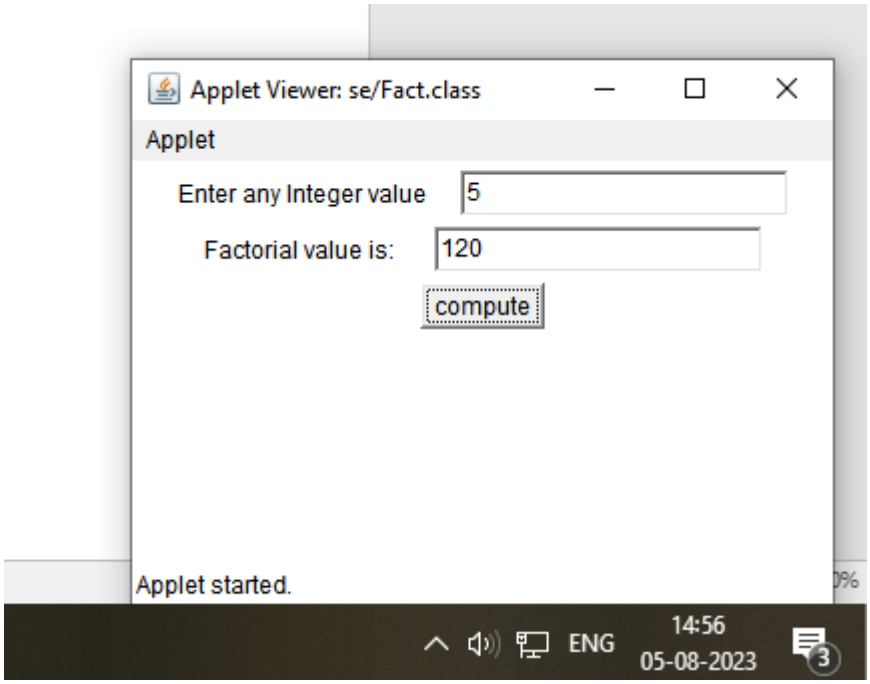
3.b) Develop an Applet that receives an integer in one text field & compute its factorial value & returns it in another text filed when the button "Compute" is clicked.

```
import java.awt.*;
import java.lang.String;
import java.awt.event.*;
import java.applet.Applet;
public class Fact extends Applet implements ActionListener
{
String str;
Button b0;
TextField t1,t2;
Label l1;
public void init(){
Panel p=new  Panel();
p.setLayout(new GridLayout());
add(new Label("Enter any Integer value"));
add(t1=new TextField(20));
add(new Label("Factorial value is: "));
add(t2=new TextField(20));
add(b0=new Button("compute"));
b0.addActionListener(this);
}
public void actionPerformed(ActionEvent e)
{
int i,n,f=1;
n=Integer.parseInt(t1.getText());
for(i=1;i<=n;i++)
```

```
f=f*i;

t2.setText(String.valueOf(f));

repaint();

 }

}
```

Output:





4. Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were Zero, the program would throw an Arithmetic Exception Display the exception in a message dialog box.

Program:-

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
public class Add1 extends Applet implements ActionListener
{
String msg;
TextField num1, num2, res;
```

```
Label l1, l2,
l3;
Button div;
public void init()
{
l1 = new Label("Number 1");
l2 = new Label("Number 2");
l3 = new Label("result");
num1 = new TextField(10);
num2 = new TextField(10);
res = new TextField(30);
div = new Button("DIV");
div.addActionListener(this);
add(l1);
add(num1);
add(l2);
add(num2);
add(l3);
add(res);
add(div);
}
public void actionPerformed(ActionEvent ae)
{
String arg = ae.getActionCommand();
if (arg.equals("DIV"))
{
```

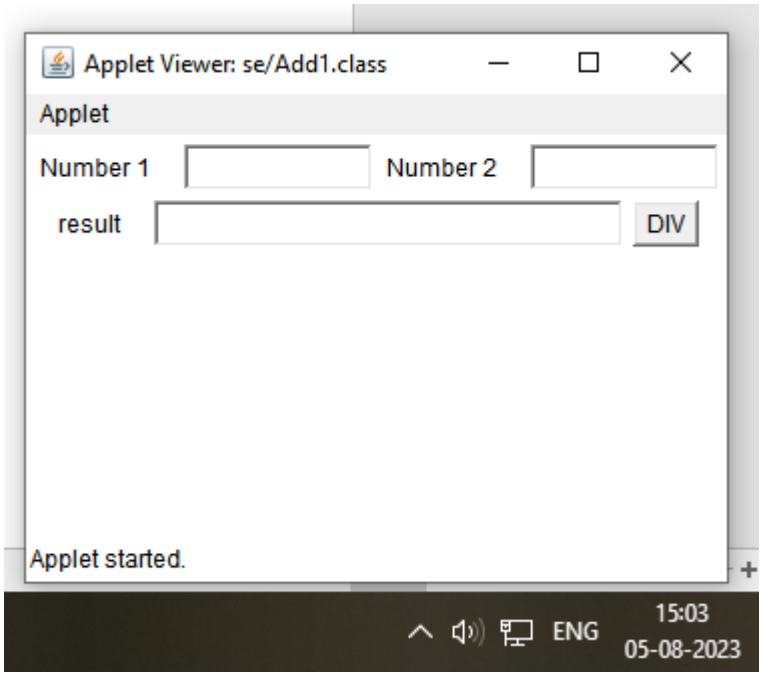
```
String s1 = num1.getText();
String s2 = num2.getText();
int num1 = Integer.parseInt(s1);
int num2 = Integer.parseInt(s2);
if (num2 == 0)
{

msg = "Arithemetic Exception ";
repaint();
}
else if ((num1 < 0) || (num2 < 0))
{
msg = "NumberFormat Exception";
repaint();
}
else
{
int num3 = num1 / num2;
msg = String.valueOf(num3);
}
res.setText(msg);
}
}
public void paint(Graphics g)
{
//g.drawString(msg, 30, 70);
}
```

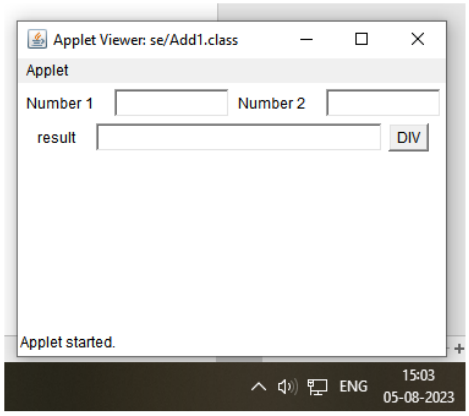
```
}
```

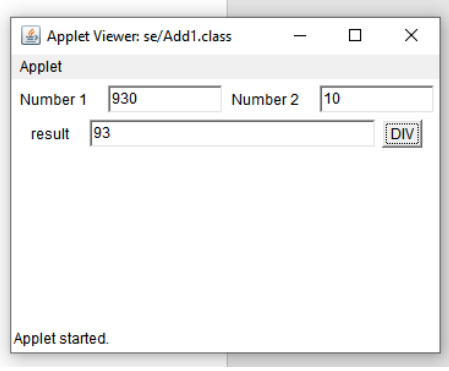
APPLET.HTML

```
<html>
<head>
</head>
<body>
/*<applet code="Add1.class"width=350 height=300>
</applet>*/
</body>
</html>
```

OUTPUT:


Applet Viewer: se/Add1.class
Applet
Number 1
Number 2
result
DIV
Applet started.
ENG
15:03
05-08-2023


Applet Viewer: se/Add1.class
Applet
Number 1
Number 2
result
DIV
Applet started.
ENG
15:03
05-08-2023


Applet Viewer: se/Add1.class
Applet
Number 1
930
Number 2
10
result
93
DIV
Applet started.

5.) Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.

```
import java.util.Random;
class RandomNumberThread extends Thread {
public void run() {
Random random = new Random();
for (int i = 0; i< 10; i++) {
int randomInteger = random.nextInt(100);
System.out.println("Random Integer generated : " + randomInteger);
if((randomInteger%2) == 0) {
SquareThreads Thread = new SquareThread(randomInteger);
sThread.start();
}
else {
CubeThread  cThread = new CubeThread(randomInteger);
cThread.start();
}
try {
Thread.sleep(1000);
}
catch (InterruptedException ex) {
System.out.println(ex);
}
}
```

```
}
}
class  SquareThread extends Thread {
int number;
SquareThread(int randomNumbern) {
number = randomNumbern;
}
public void run() {
System.out.println("Square of " + number + " = " + (number * number));
}
}
class  CubeThread extends Thread {
int number;
CubeThread(int   randomNumber) {
number = randomNumber;
}
public void run() {
System.out.println("Cube of " + number + " = " + number * number * number);
}
}
public class MultiThreadingTest {
public static void main(String args[ ]) {
RandomNumberThread  rnThread = new RandomNumberThread();
rnThread.start();
}
}
```

Output:

Or

```
package se;
import java.util.Random;
class Square extends Thread
{
int x;
Square(int n)
 {
 x = n;
 }
public void run()
 {
int sqr = x * x;
System.out.println("Square of " + x + " = " + sqr );
 }
}
class Cube extends Thread
{
int x;
Cube(int n)
 {
 x = n;
 }
public void run()
 {
int cub = x * x * x;
System.out.println("Cube of " + x + " = " + cub );
```

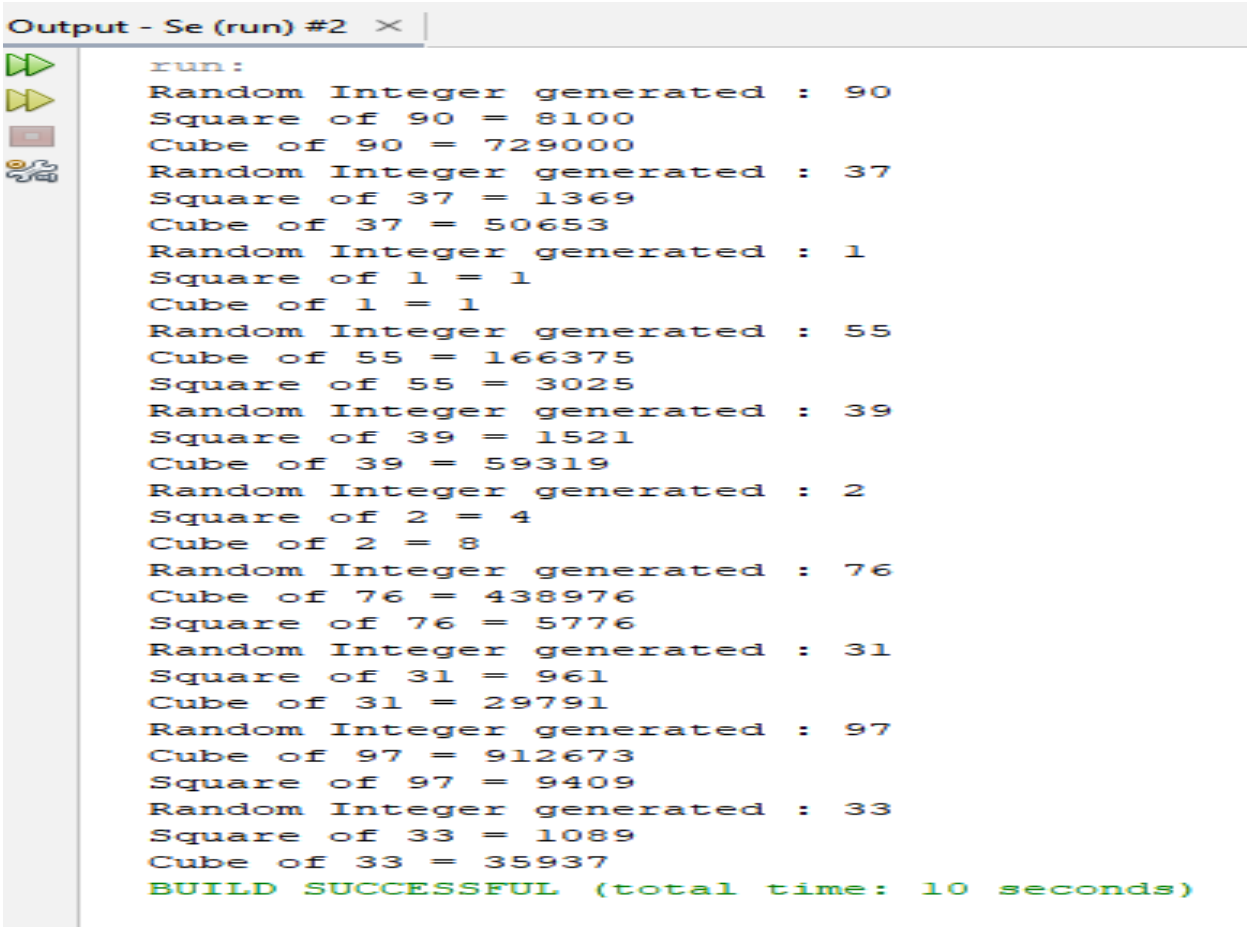
```
}
}
class Number extends Thread
{
public void run()
 {
 Random random = new Random();
for(int i =0; i<10; i++)
 {
int  randomInteger = random.nextInt(100);
System.out.println("Random Integer generated : " + randomInteger);
 Square s = new Square(randomInteger);
s.start();
 Cube c = new Cube(randomInteger);
c.start();
try {
Thread.sleep(1000);
```

/*This thread generates random number 10 times
between 1 to 100 for every 1 second. The generated
random number is then passed as argument to
Square and Cube threads.
Output varies each time a program is executed.*/

```
 } catch (InterruptedException ex) {
System.out.println(ex);
}
```

```
 }
 }
}
public class Lab3 {
public static void main(String args[ ])
 {
 Number n = new Number();
n.start();
 }
}
```

Output:



6.Write a Java program for the following: Create a doubly linked list of elements. Delete a given element from the above list. Display the contents of the list after deletion.

```
// Java program to delete a node from
// Doubly Linked List // Class for Doubly Linked List


 public class DLL {
Node head; // head of list
/* Doubly Linked list Node*/
class Node {
int data;
Node prev;
Node next; // Constructor to create a new node
// next and prev is by default initialized // as null


Node(int d) {
data = d;
 }}
public void push(int new_data) {
Node new_Node = new Node(new_data);


new_Node.next = head;
new_Node.prev = null;
if (head != null)
head.prev = new_Node;
head = new_Node;
 }
```

```
public void printlist(Node node) {
Node last = null;
while (node != null)
{
 System.out.print(node.data + " ");
last = node;
node = node.next;
}
 System.out.println();
}
void deleteNode(Node del)
{
if (head == null || del == null) {
return;
}
if (head == del) {
 head = del.next;
 }

if (del.next != null) {
del.next.prev = del.prev;
 }
if (del.prev != null) {
del.prev.next = del.next;
}
return; }
public static void main(String[ ] args)
{
```

```
DLL dll = new DLL();
dll.push(2);
dll.push(4);
dll.push(8);
dll.push(10);
System.out.print("Created DLL is: ");
dll.printlist(dll.head);
dll.deleteNode(dll.head);
System.out.print("\nList after deleting first node: ");
dll.printlist(dll.head);
dll.deleteNode(dll.head.next);
System.out.print("\nList after Deleting middle node: ");
dll.printlist(dll.head);
}
}
```

OUTPUT:

Original Linked list 10 8 4 2

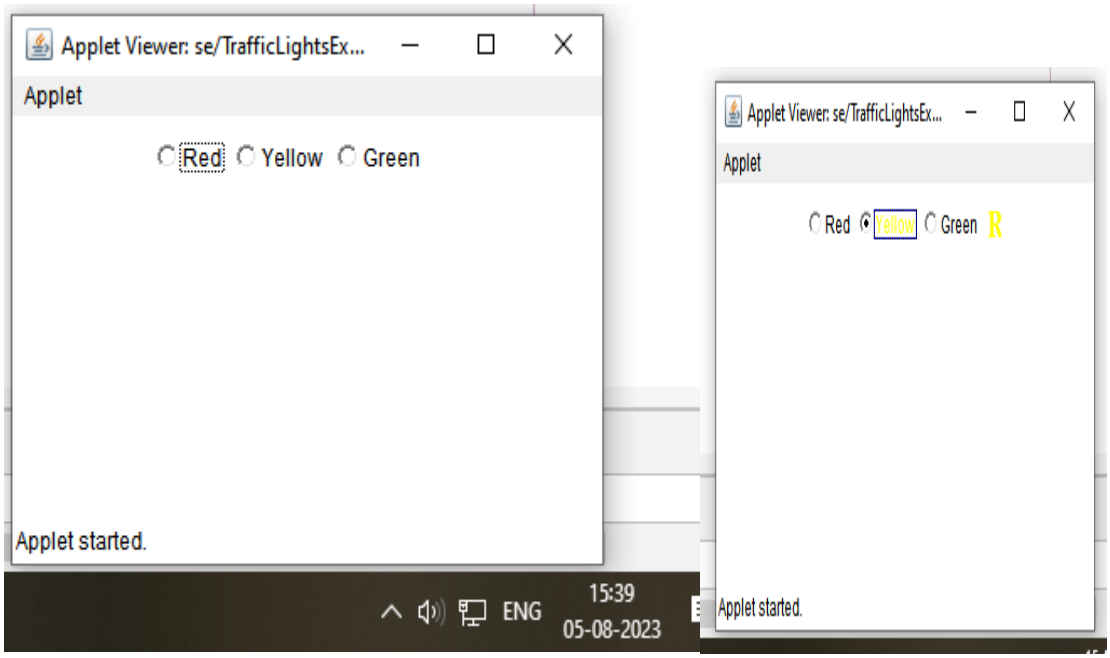
Modified Linked list 8

7. Write a Java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green with radio buttons. On selecting a button, an appropriate message with "Stop" or "Ready" or "Go" should appear above the buttons in selected color. Initially, there is no message shown.

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
/*
* <applet code = "TrafficLightsExample" width = 1000 height = 500>
* </applet>
* */

public class TrafficLightsExample extends Applet implements ItemListener{
CheckboxGroup grp = new CheckboxGroup();
Checkbox redLight, yellowLight, greenLight;
Label msg;

public void init(){

redLight = new Checkbox("Red", grp, false);
yellowLight = new Checkbox("Yellow", grp, false);
greenLight = new Checkbox("Green", grp, false);
msg = new Label("");
redLight.addItemListener(this);
yellowLight.addItemListener(this);
greenLight.addItemListener(this);

add(redLight);

add(yellowLight);
```

```
add(greenLight);

add(msg);

msg.setFont(new Font("Serif", Font.BOLD, 20));
}
public void itemStateChanged(ItemEventie) {
redLight.setForeground(Color.BLACK);
yellowLight.setForeground(Color.BLACK);
greenLight.setForeground(Color.BLACK);
if(redLight.getState() == true) {
    redLight.setForeground(Color.RED);
    msg.setForeground(Color.RED);
    msg.setText("STOP");
}
else if(yellowLight.getState() == true) {
        yellowLight.setForeground(Color.YELLOW);
        msg.setForeground(Color.YELLOW);
        msg.setText("READY");
}
else{
       greenLight.setForeground(Color.GREEN);
       msg.setForeground(Color.GREEN);
       msg.setText("GO");
}
}
}
```

Output:



8.Write a Java program to create an abstract class named Shape that contains two integers and an empty method named print Area (). Provide three classes named Rectangle, Triangle, and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the method print Area () that prints the area of the given shape.

PROGRAM:

```
import java.util.*;
 abstract class Shape {
 int length, breadth,radius;
Scanner input = new Scanner(System.in);
 abstract void printArea();
}
class Rectangle extends Shape {
 void printArea() {
```

```
System.out.println("*** Finding the Area of Rectangle ***");
System.out.print("Enter length and breadth: ");
length = input.nextInt();
breadth = input.nextInt();
System.out.println("The area of Rectangle is: " + length * breadth);
}
}
class Triangle extends Shape {
 void printArea() {
System.out.println("\n*** Finding the Area of Triangle ***");
System.out.print("Enter Base And Height: ");
length = input.nextInt();
breadth = input.nextInt();
System.out.println("The area of Triangle is: " + (length * breadth) / 2);
}
}
class  Cricle extends Shape {
void printArea() {
System.out.println("\n*** Finding the Area of Cricle ***");
System.out.print("Enter Radius: "); radius =
input.nextInt();
System.out.println("The area of Cricle is: " + 3.14f * radius * radius);
}
}
public class AbstractClassExample {
```

```
public static void main(String[] args) {
Rectangle rec = new Rectangle();
rec.printArea();
Triangle tri = new Triangle();
tri.printArea();
Cricle cri = new Cricle();
cri.printArea();
}
}
```

Output:

```
*** Finding the Area of Rectangle ***
Enter length and breadth: 1
2
The area of Rectangle is: 2

*** Finding the Area of Triangle ***
Enter Base And Height: 1 2
The area of Triangle is: 1

*** Finding the Area of Cricle ***
Enter Radius: 3
The area of Cricle is: 28.26
BUILD SUCCESSFUL (total time: 49 seconds)
```

9.Suppose that a table named Table.txt is stored in a text file. The first line in the file is the header, and the remaining lines correspond to rows in the table. The elements are separated by commas. Write a java program to display the table using Labels in Grid Layout.

program:

```
import java.io.*;
import java.util.*;
import java.awt.*;
import javax.swing.*;
class A extends JFrame {
public A() {
setSize(400, 400);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
GridLayout g = new GridLayout(0, 3);
setLayout(g);
try {
FileInputStream fin = new FileInputStream("C:\\Users\\User\\eclipseworkspace\\LabManual\\src\\HashTab.txt");
Scanner sc = new Scanner(fin).useDelimiter(",");
String[] arrayList;
String a;
while (sc.hasNextLine()) {
a = sc.nextLine();
arrayList = a.split(",");
for (String i : arrayList) {
add(new JLabel(i));
}
}
```

```
} catch (Exception ex) {
}
setDefaultLookAndFeelDecorated(true);
pack();
setVisible(true);
}
}
public class TableTest {
public static void main(String[] args) {
A a = new A();
}
}
```

Output:

10.Write a Java program that handles all mouse events and shows the event name at the center of the window when a mouse event is fired (Use Adapter classes).

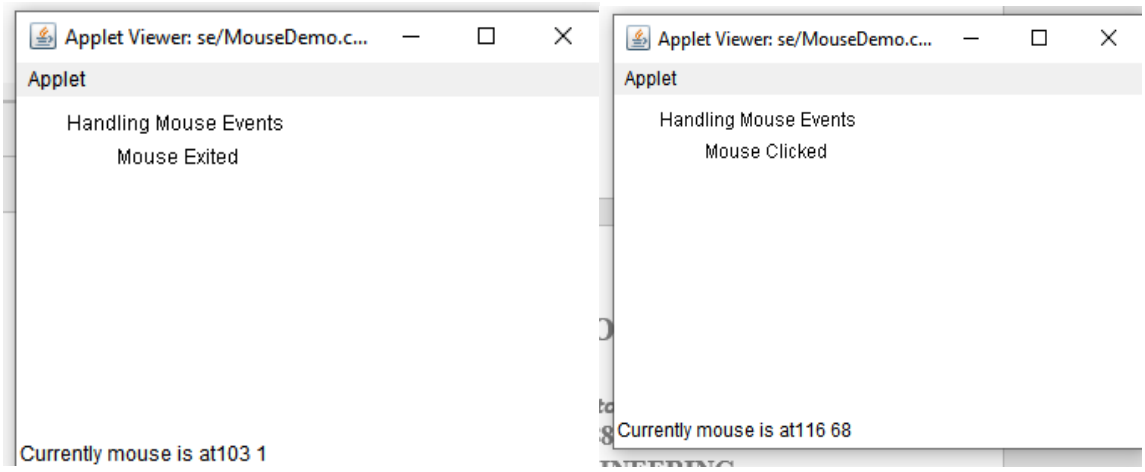
Program:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
/*<applet code="MouseDemo" width=300 height=300>
</applet>*/

public class MouseDemo extends Applet implements MouseListener,
MouseMotionListener {
int mx = 0;
int my = 0;
String msg = "";
public void init() {
addMouseListener(this);
addMouseMotionListener(this);
}
public void mouseClicked(MouseEvent me)
 {
mx = 20;
my = 40;
msg = "Mouse Clicked";
repaint();
}
```

```
public void mousePressed(MouseEvent me) {
mx = 30;
my = 60;
msg = "Mouse Pressed";
repaint();
}
public void mouseReleased(MouseEvent me) {
mx = 30;
my = 60;
 msg = "Mouse Released";
repaint();
}
public void mouseEntered(MouseEvent me) {
mx = 40;
my = 80;
 msg = "Mouse Entered";
repaint();
}
public void mouseExited(MouseEvent me) {
mx = 40;
my = 80;
msg = "Mouse Exited";
repaint();
}
public void mouseDragged(MouseEvent me) {
mx = me.getX();
my = me.getY();
```

```
showStatus("Currently mouse dragged" + mx + " " + my);
repaint();
}
public void mouseMoved(MouseEvent me) {
mx = me.getX();
my = me.getY();
showStatus("Currently mouse is at" + mx + " " + my);
repaint();
}
public void paint(Graphics g) {
g.drawString("Handling Mouse Events", 30, 20);
g.drawString(msg, 60, 40);
}
}
```

Output:



11.Write a java program that loads names and phone numbers from a text file where the data is organized as one line per record and each field in a record are separated by a tab (\t).it takes a name or phone number as input and prints the corresponding other value from the hash table(hint: use hash tables)

Program:

```
importjava.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.Set;
public class HashTab {
public static void main(String[] args) {
HashTab prog11 = new HashTab();
Hashtable<String, String>hashData = prog11.readFromFile("HashTab.txt");
System.out.println("File data into Hashtable:\n" + hashData);
prog11.printTheData(hashData,
"raja"); prog11.printTheData(hashData, "123");
prog11.printTheData(hashData, "--- ");
}
private void printTheData(Hashtable<String, String>hashData, String input)
{ String output =null;
if (hashData != null) {
Set<String> keys = hashData.keySet();
if (keys.contains(input)) {
output = hashData.get(input);
```

```
} else {
Iterator<String> iterator = keys.iterator(); while (iterator.hasNext())
{
String key = iterator.next();
String value = hashData.get(key);
if (value.equals(input)) {
output = key; break;
}
}
}
}
System.out.println("Input given:" + input);
if (output != null) {
System.out.println("Data found in HashTable:" + output);
} else {
}
}
privateHashtable<String, String>readFromFile(String fileName) {
Hashtable<String, String>hashData = new Hashtable<String, String>();
try {
File f = new File("D:\\java\\" + fileName);
BufferedReaderbr = new BufferedReader(new FileReader(f));
String line = null; while ((line = br.readLine()) != null) {
String[] details = line.split("\t");
hashData.put(details[0], details[1]);
}
} catch (FileNotFoundException e) {
e.printStackTrace();
```

```
} catch (IOException e) {
e.printStackTrace();
}
returnhashData;
}
}
```

Output:

12.Write a Java program that correctly implements the producer – consumer problem using the concept of interthread communication.

Program:

```
classItemQueue {
int item;
booleanvalueSet = false;
synchronized intgetItem()
{
while (!valueSet)
try {
 wait();
} catch (InterruptedException e) {
System.out.println("InterruptedException caught");
}
System.out.println("Consummed:" + item); valueSet
= false;
try {
Thread.sleep(1000);
} catch (InterruptedException e) {
System.out.println("InterruptedException caught");
}
notify();
return item;
}
synchronized void putItem(int item) {
while (valueSet)
```

```
try { wait();
} catch
(InterruptedE
xception e) {
System.out.println("InterruptedException caught");
}
this.item = item;
valueSet = true;
System.out.println("Produced: " + item);
try {
Thread.sleep(1000);
} catch (InterruptedException e) {
System.out.println("InterruptedException caught");
}
notify();
}
}
class Producer implements Runnable{
ItemQueueitemQueue;
Producer(ItemQueueitemQueue){
this.itemQueue = itemQueue; new
Thread(this, "Producer").start();
}
public void run() {
int i = 0; while(true) {
itemQueue.putItem(i++);
}
}
}
class Consumer implements Runnable{
```

```
ItemQueue  itemQueue;
Consumer(ItemQueueitemQueue){
this.itemQueue  =  itemQueue;  new
Thread(this, "Consumer").start();
}
public void run() { while(true)
{ itemQueue.getItem();
}
}
}
classProducerConsumer{
public static void main(String args[]) {
ItemQueueitemQueue
= new ItemQueue(); new Producer(itemQueue); new
Consumer(itemQueue);
}
}
```

Output:

```
run:
Produced: 0
Consummed:0
Produced: 1
Consummed:1
Produced: 2
Consummed:2
Produced: 3
Consummed:3
Produced: 4
Consummed:4
Produced: 5
Consummed:5
Produced: 6
Consummed:6
Produced: 7
Consummed:7
Produced: 8
Consummed:8
Produced: 9
Consummed:9
Produced: 10
Consummed:10
Produced: 11
Consummed:11
Produced: 12
Consummed:12
Produced: 13
Consummed:13
Produced: 14
Consummed:14
Produced: 15
Consummed:15
Produced: 16
Consummed:16
Produced: 17
Consummed:17
```

13.Write a Java program to list all the files in a directory including the files present in all its subdirectories.

Program:

```
importjava.util.Scanner;
 import java.io.*;
 public class ListingFiles {
public static void main(String[] args) {
String path = null;
Scanner read = new Scanner(System.in);
System.out.print("Enter the root directory name: "); path =
read.next() + ":\\"; File f_ref = new File(path);
if (!f_ref.exists()) {
printLine();
System.out.println("Root directory does not exists!"); printLine();
}
else {
String ch = "y";
while (ch.equalsIgnoreCase("y")) {
printFiles(path);
System.out.print("Do you want to open any sub-directory(Y/N): ");
ch = read.next().toLowerCase(); if (ch.equalsIgnoreCase("y")) {
System.out.print("Enter the sub-directory name: "); path = path +
"\\\\" + read.next(); File f_ref_2 = new File(path);
if (!f_ref_2.exists()) {
printLine();
System.out.println("The sub-directory does not exists!");
printLine();
```

```
intlastIndex = path.lastIndexOf("\\");

path = path.substring(0, lastIndex);
}
}
}
}
System.out.println("***** Program Closed *****");
}
public static void printFiles(String path) {
System.out.println("Current Location: " + path);
File f_ref = new File(path);
File[] filesList = f_ref.listFiles();
for (File file : filesList) {
if (file.isFile())
System.out.println("- " + file.getName());
else
System.out.println("> " + file.getName());
}
}
public static void printLine() {
System.out.println("- -");
}
}
```

Output:

```
Enter the root directory name: secsiot
- -
Root directory does not exists!
- -
***** Program Closed *****
BUILD SUCCESSFUL (total time: 35 seconds)
```

14. Write a Java program that implements Quick sort algorithm for sorting a list of names 41 in ascending Order.

Program:

```
public class QuickSortOnStrings { String names[];

int length;

public static void main(String[] args) {

QuickSortOnStringsobj = new

QuickSortOnStrings();

String stringsList[] = {""cse", "aiml", "ds", "se", "cs", "iot", "hello"};
obj.sort(stringsList);
for (String i : stringsList) {
System.out.print(i);

System.out.print(" ");

}

}

void sort(String array[]) {

if (array == null || array.length == 0) {

return;

}

this.names = array;

this.length = array.length;

quickSort(0, length - 1);
```

```
}
void   quickSort(intlowerIndex, inthigherIndex) {
 int i = lowerIndex; int j = higherIndex;
String pivot = this.names[lowerIndex + (higherIndex - lowerIndex) / 2];
while (i<= j) {
while (this.names[i].compareToIgnoreCase(pivot) < 0) {

 i++;
}
while (this.names[j].compareToIgnoreCase(pivot) > 0) {
j--;
} if (i<= j)
{
exchangeNames(i, j);
i++;
j--;
}
}
if (lowerIndex< j) {

quickSort(lowerIndex, j);
}
if (i<higherIndex) { quickSort(i,
higherIndex);
}
}
void   exchangeNames(int i, int j)
{
```

```
String temp = this.names[i];


this.names[i] = this.names[j];
this.names[j] = temp;
}
}
```

Output:

```
aiml cs cse ds hello iot se BUILD SUCCESSFUL (total time: 0 seconds)
```

15.Write a Java program that implements Bubble sort algorithm for sorting in descending order and also shows the number of interchanges occurred for the given set of integers.

Program:

```
importjava.util.Scanner;
public class BubbleSort {
 public static void main(String[] args) {
Scanner read = new
Scanner(System.in);
int size, count = 0;
//Reading size of the list
System.out.print("Enter the list size: ");
size = read.nextInt(); //Creating list
with elements int list[] = new int[size];
System.out.println("Enter any " + size + " integer numbers: ");
 for(inti = 0; i< size; i++)

list[i] = read.nextInt();
```

```
// Bubble sort logic int temp=0;
for(inti=0;i<size-1;i++) {
for(int j=0;j<size-i1;j++) {
 if(list[j]<list[j+1])
{
temp=list[j];
list[j]=list[j+1];
list[j+1]=temp;
count++;
}
}
}
// Displaying sorted list
System.out.println("List of sorted elements: ");
for(int x:list) {
System.out.print(x + " ");
}
System.out.println("\nTotal number of Interchanges is " + count);
}
}
```

Output:

```
Enter the list size: 10
Enter any 10 integer numbers:
1
1
2
3
2
4
3
8
7
9
```