**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

**JAVA PROGRAMMING LAB MANUAL -** B20CS14 **(R22)**

**B. TECH - III SEMESTER**

****

**VAAGDEVI COLLEGE OF ENGINEERING**

**(UGC AUTONOMOUS)**

**VAAGDEVI COLLEGE OF ENGINEERING**

**(AUTONOMOUS)**

**(**B20CS14**) JAVA PROGRAMMING LAB**

**B. TECH- III Semester L /T/ P/ C**

**0 / 0//3 /1.5**

**Course Objectives:**

* To write programs using abstract class.
* 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 class.
* Able to write multithreaded programs.
* Able to write GUI programs using swing controls in Java.

**Syllabus Content**

**Week 1:**

Use Eclipse or Net bean 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..

**Week 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 divided by zero.

**Week 3:**

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 factorial Value and returns it in another text field, when the button named “Compute” is clicked.

**Week 4:** Write a Java 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 Num 2 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 Number Format Exception. If Num2 were Zero, the program would throw an Arithmetic Exception. Display the exception in a message dialog box.

**Week 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.

**Week 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.

**Week 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.

**Week 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.

**Week 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.

**Week 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).

**Week 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)

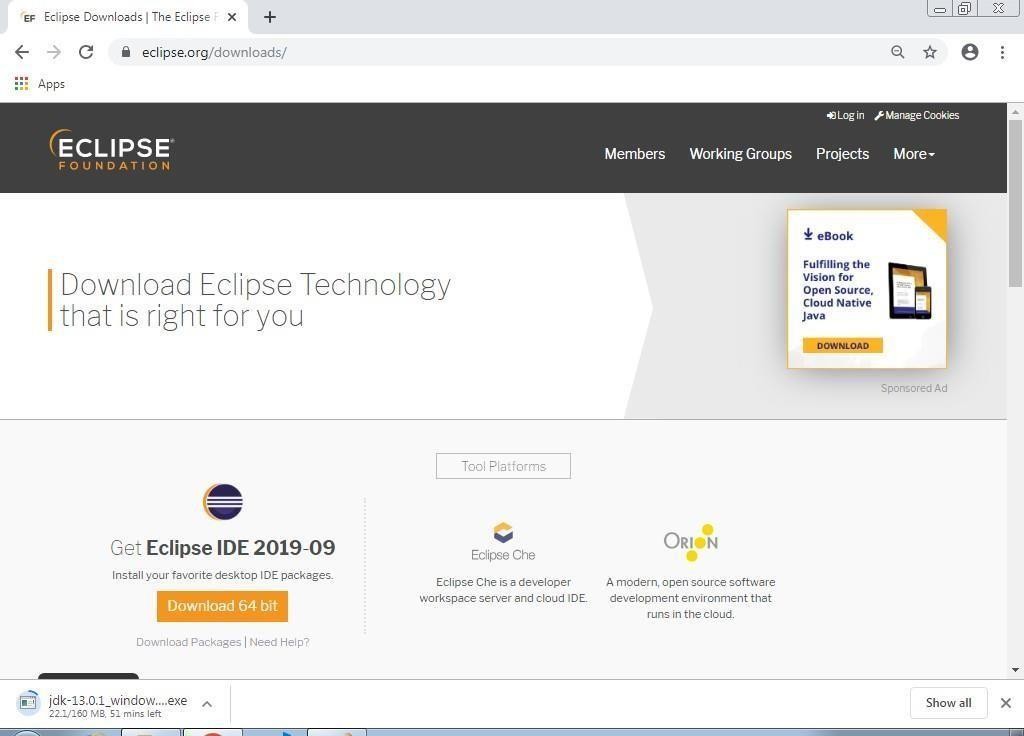
**Week – 12:** Write a Java program that correctly implements the producer – consumer problem using the concept of nterthread communication.

**Week – 13:** Write a Java program to list all the files in a directory including the files present in all its subdirectories

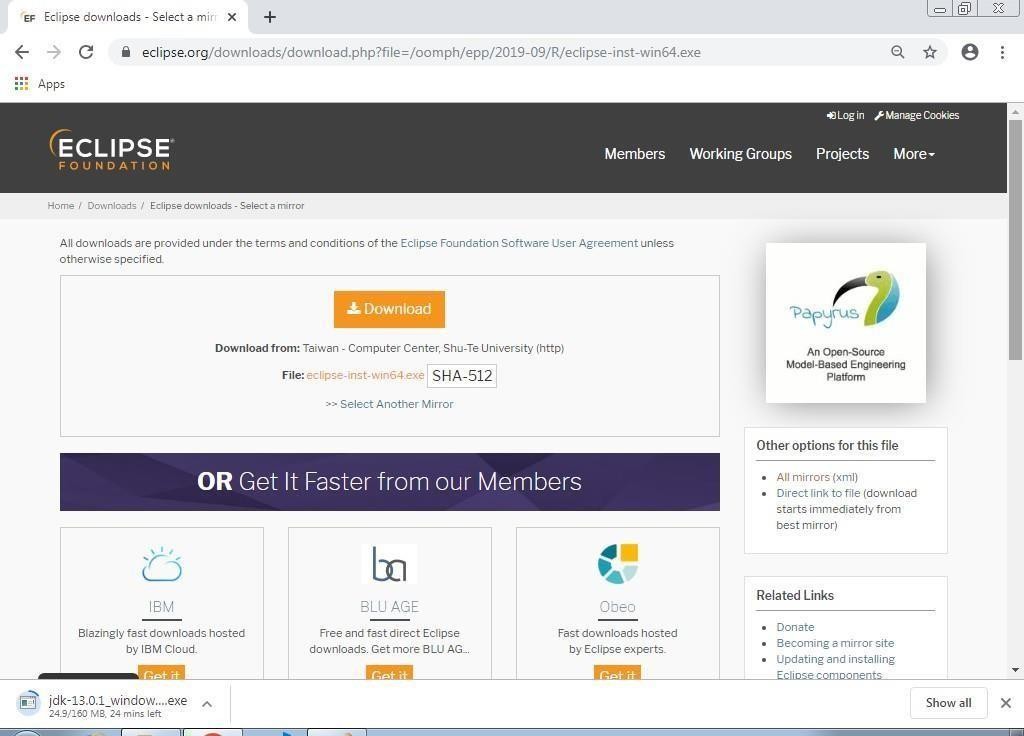
**Week 1:**

Use Eclipse or Net bean 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..

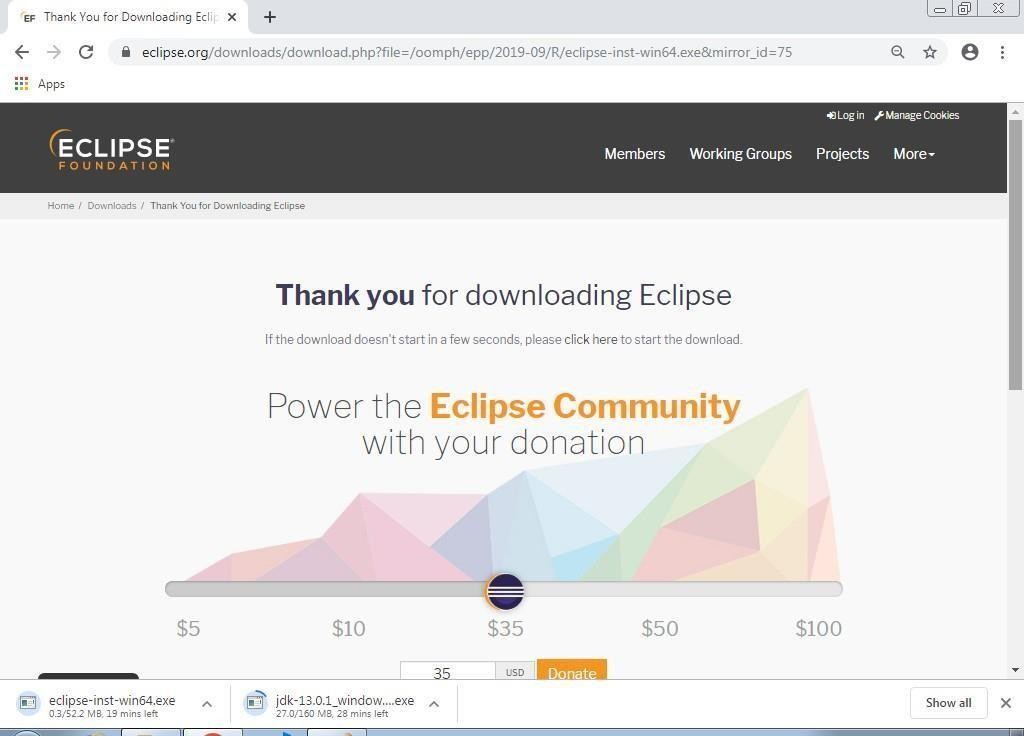
* + **Step 1 -** Install JDK in the computer.
  + **Step 2 -** Set the path in the Environment Variables from Advanced Setting of computer
  + **Step 3 -** Download Eclipse from Eclipse website
  + **Step 4 -** Install the Eclipse (follow the screen to install eclipse)



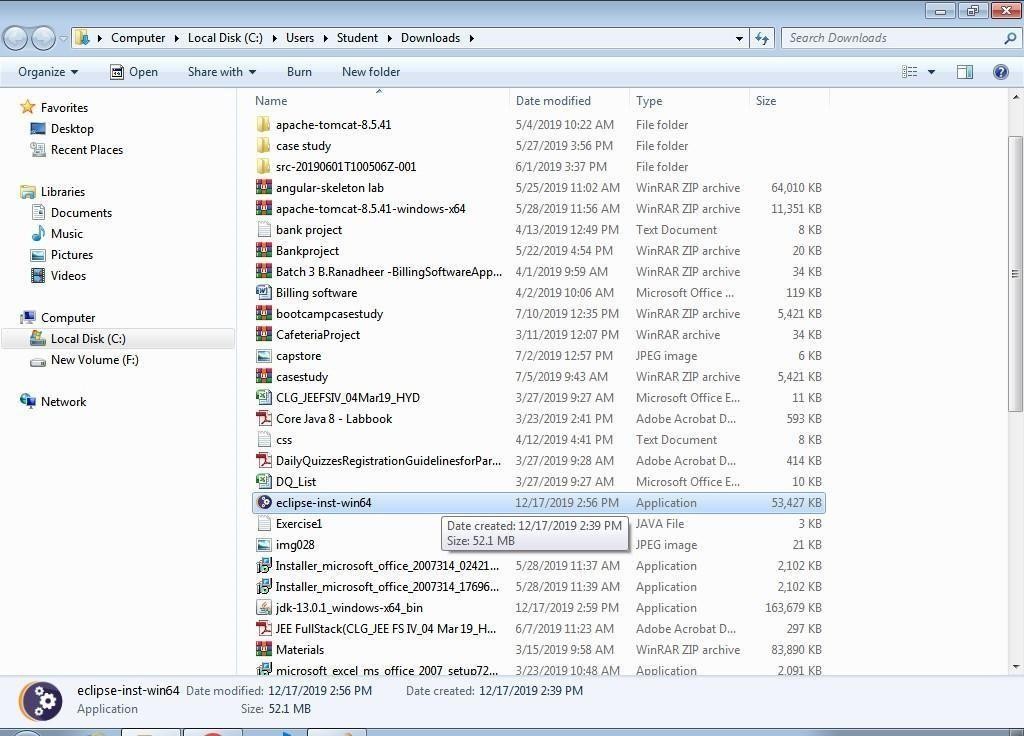
Select the suitable version based on your OS.



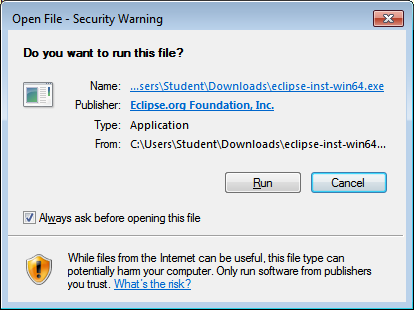
Then download get starts.



Double click on the Eclipse Application.



Click on Run in the Security Warning box.



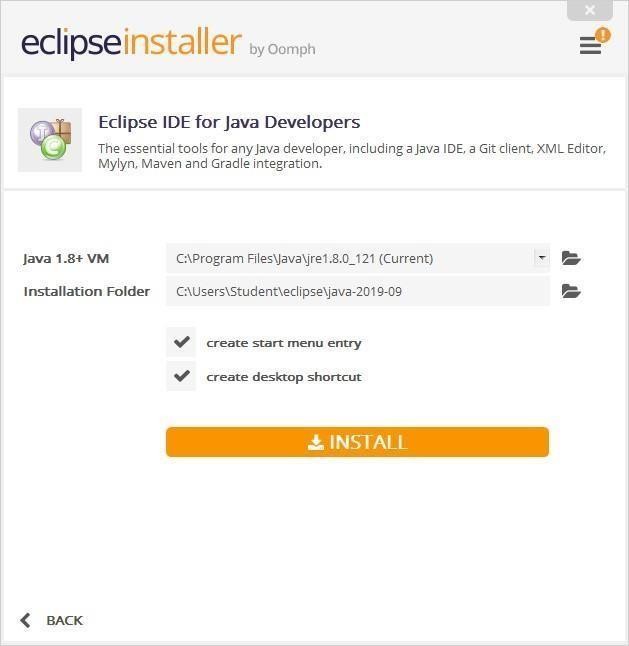
Then, the installation process begins.



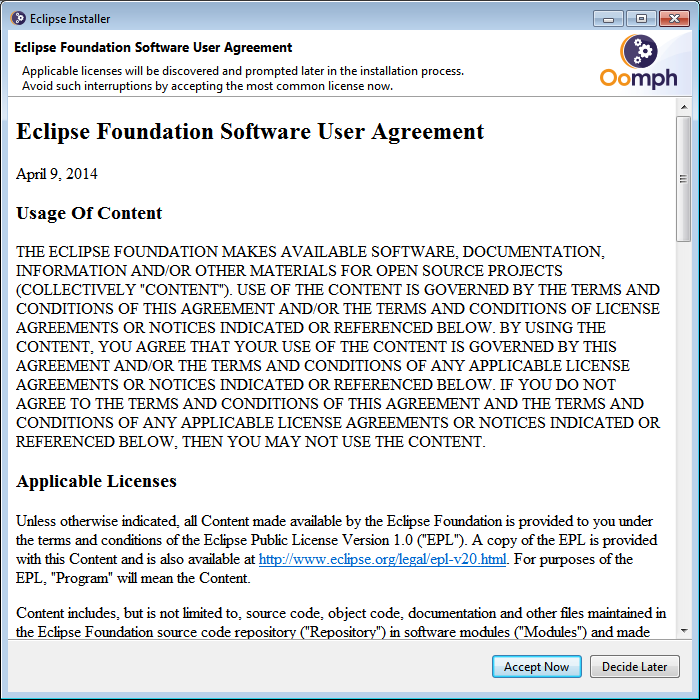
Click on Eclipse IDE for Java Developers.



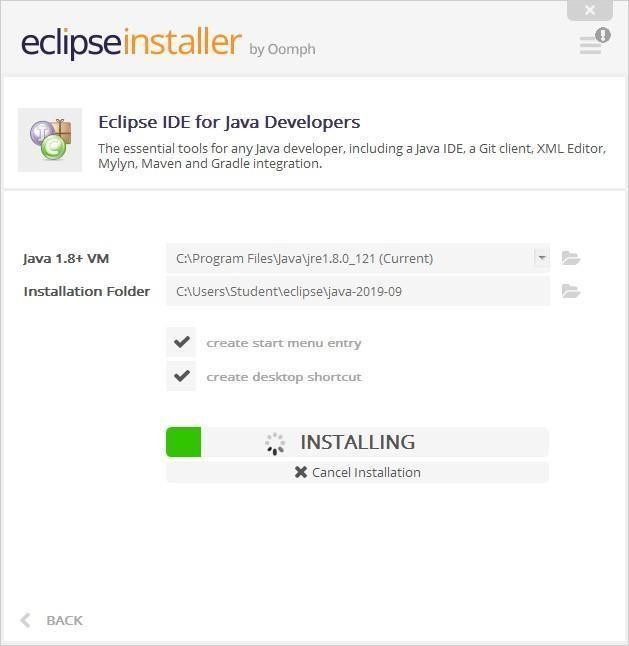
Click on Install button.



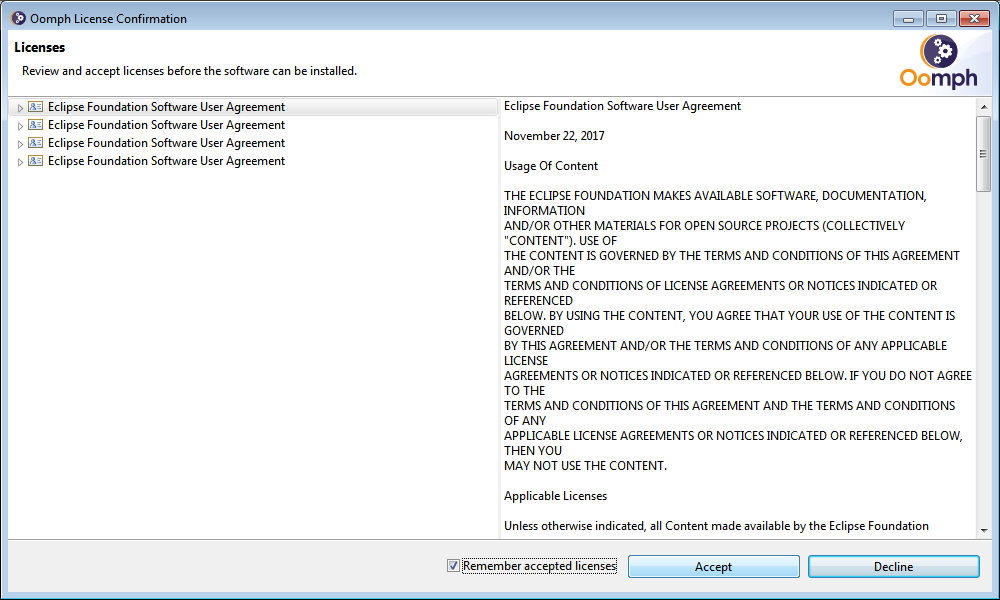
Click on Accept Now.



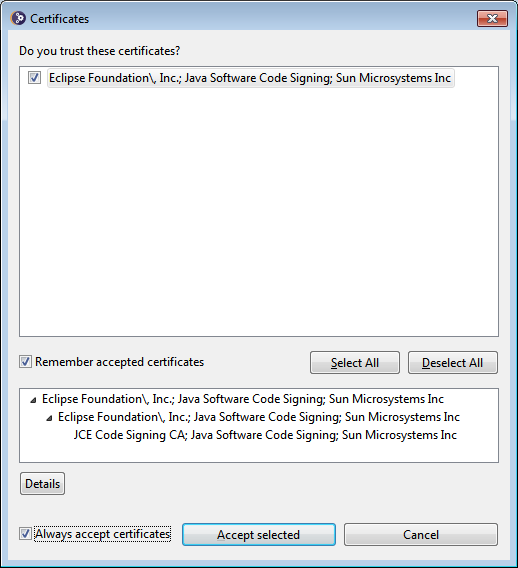
Then the Eclipse installation begins.



Click on Accept



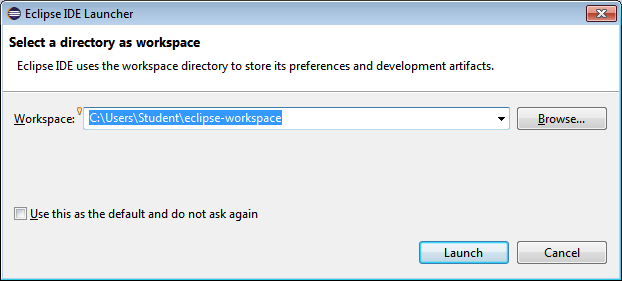
Click on Select All and Accept Selected.



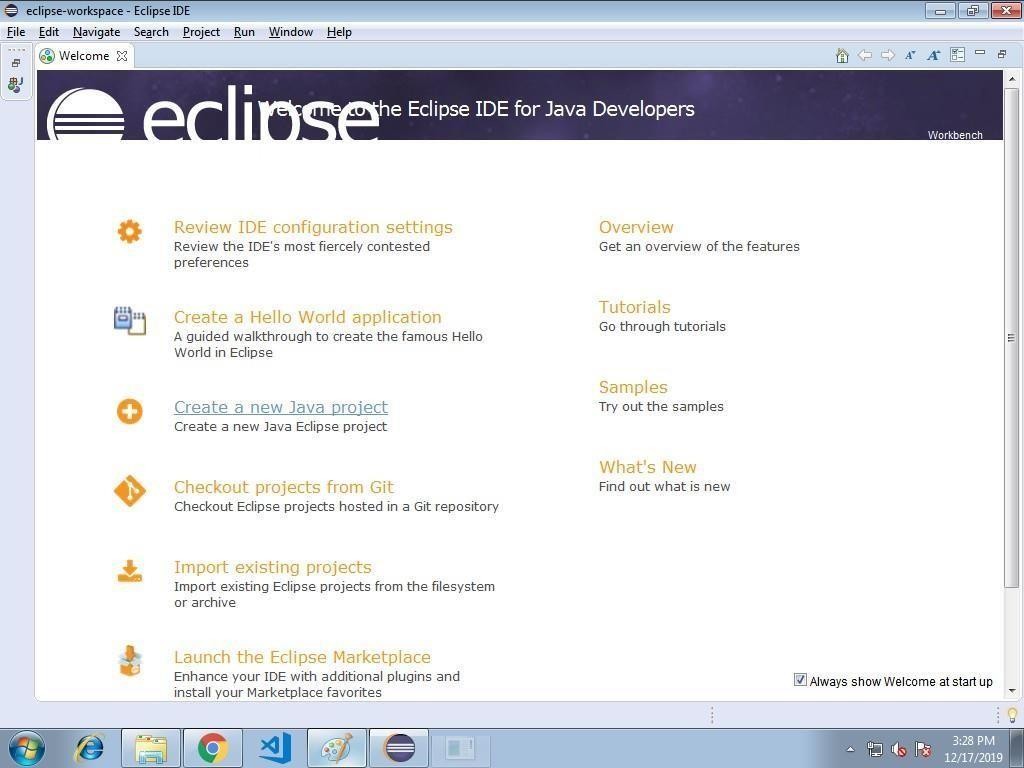
After completing, click on Launch to start the Eclipse IDE.

CREATING PROJECT AND CLASSES IN ECLIPSE IDE

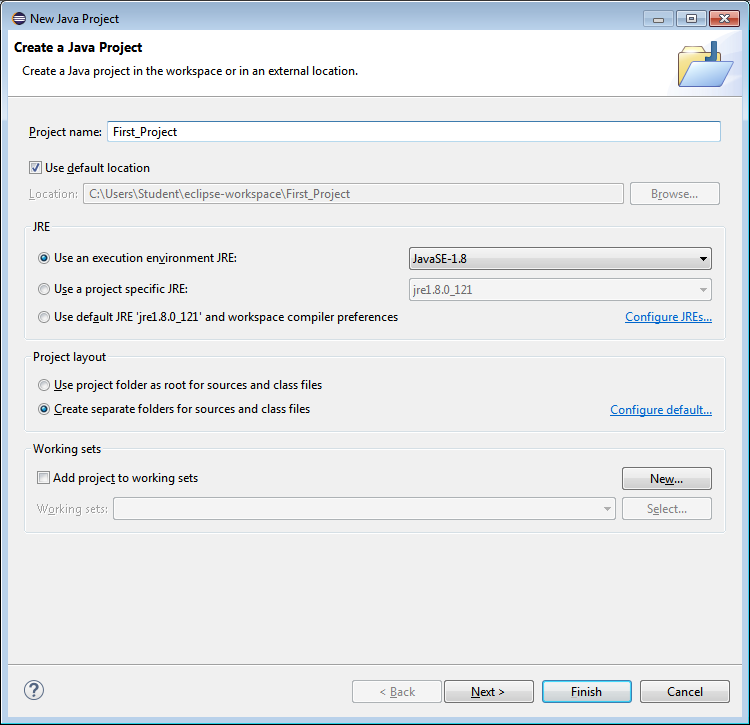
Browse the Workspace for storing the java project and click on Launch.



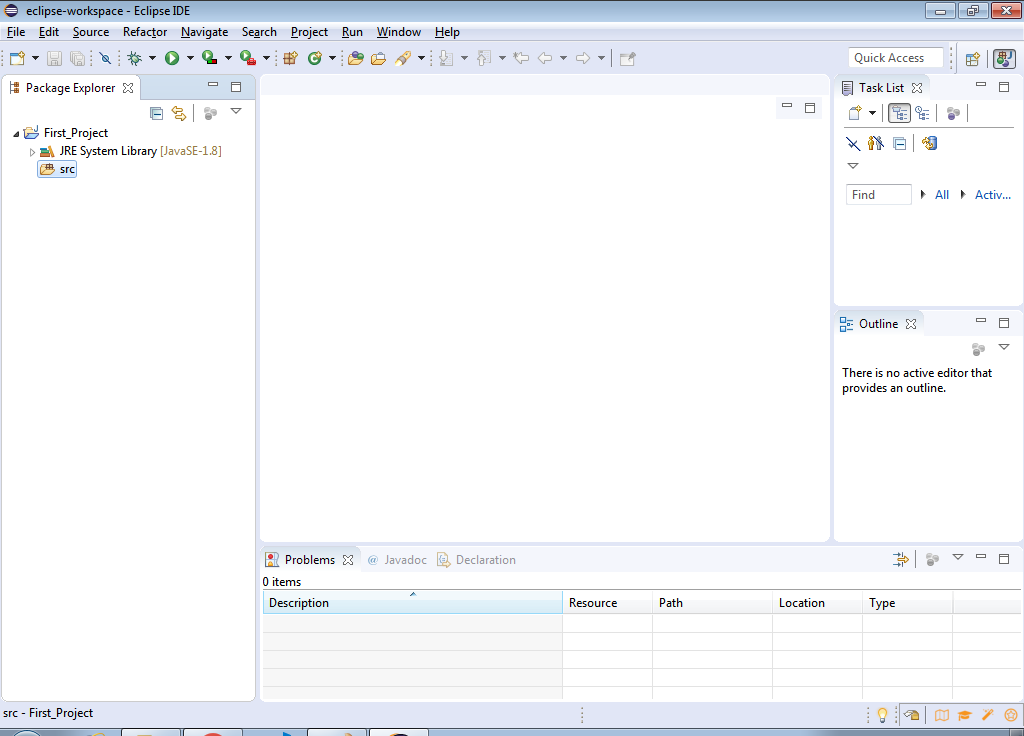
Select "Create a new Java project".

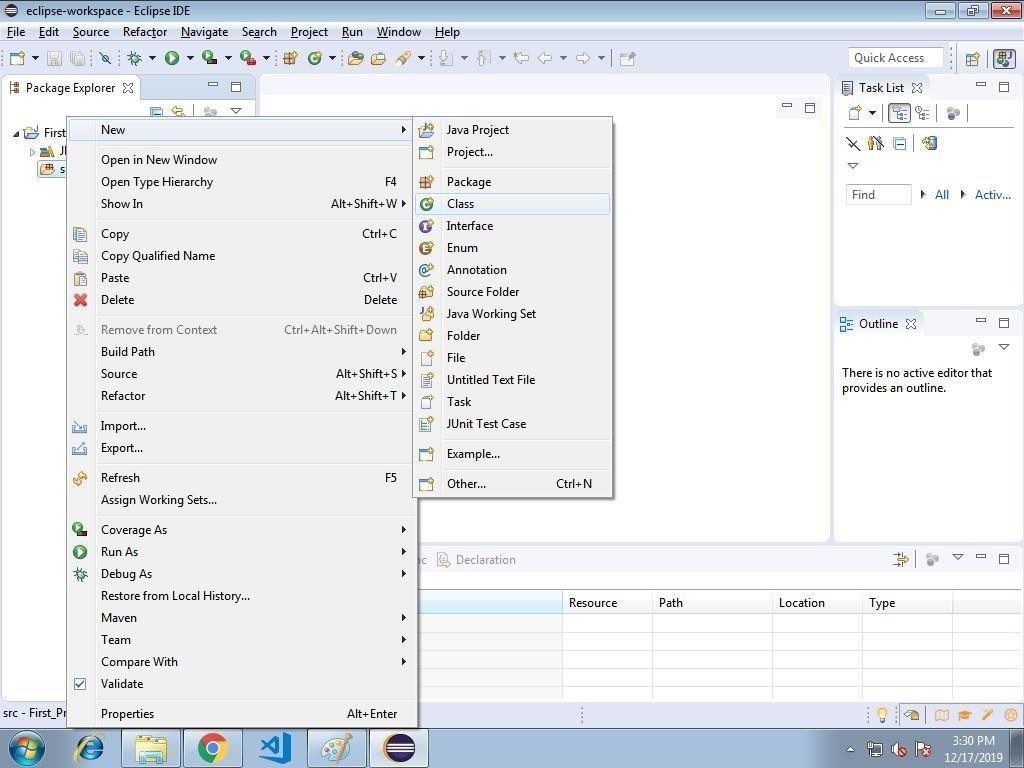


Type the project name and click on Finish.

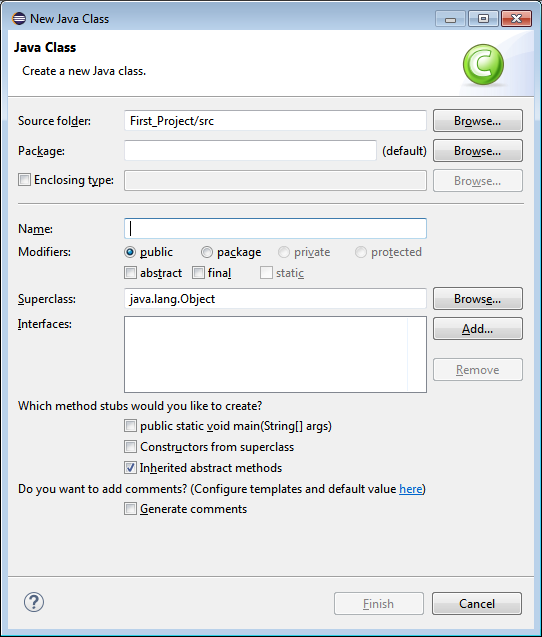


Now, create the class in src directory from Package Explorer window.

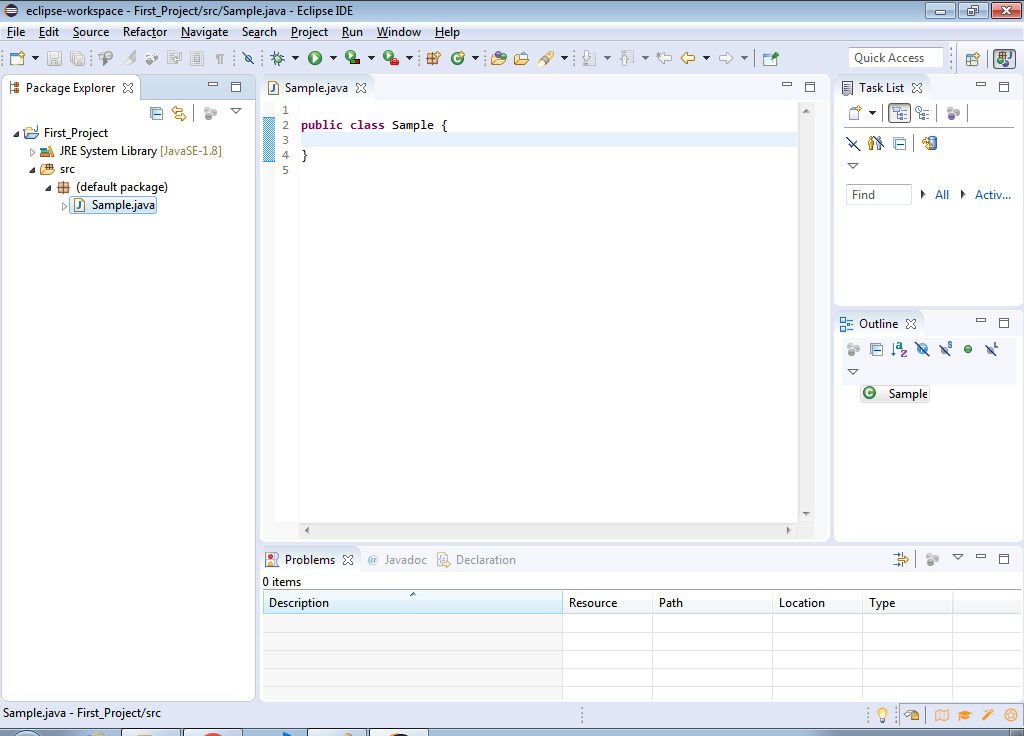




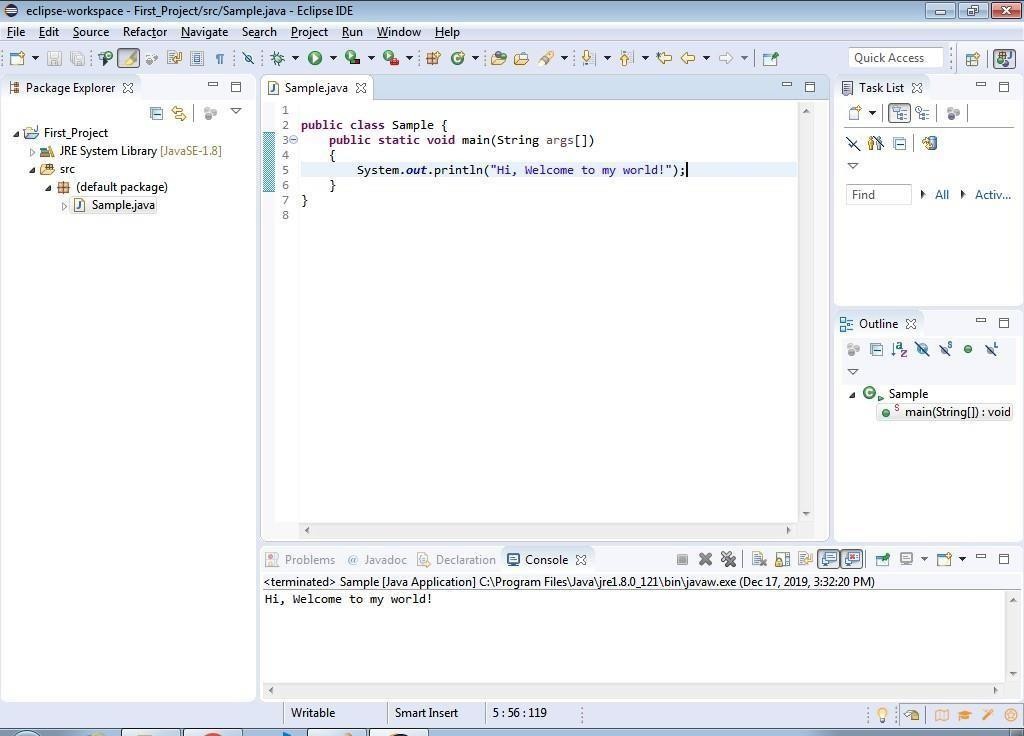
Type the class name and click on Finish.



Type the java code.



Click on Play button to run or execute the java code.



**Program:**

public class Prog1

{

public static void main(String[] args)

{

System.out.println("\n Prog. is showing even no");

for(int i=2;i<=20;i++)

{

if(i%2==0)

{

System.out.print("\n "+i);

}

}

}

}

**Compile:**

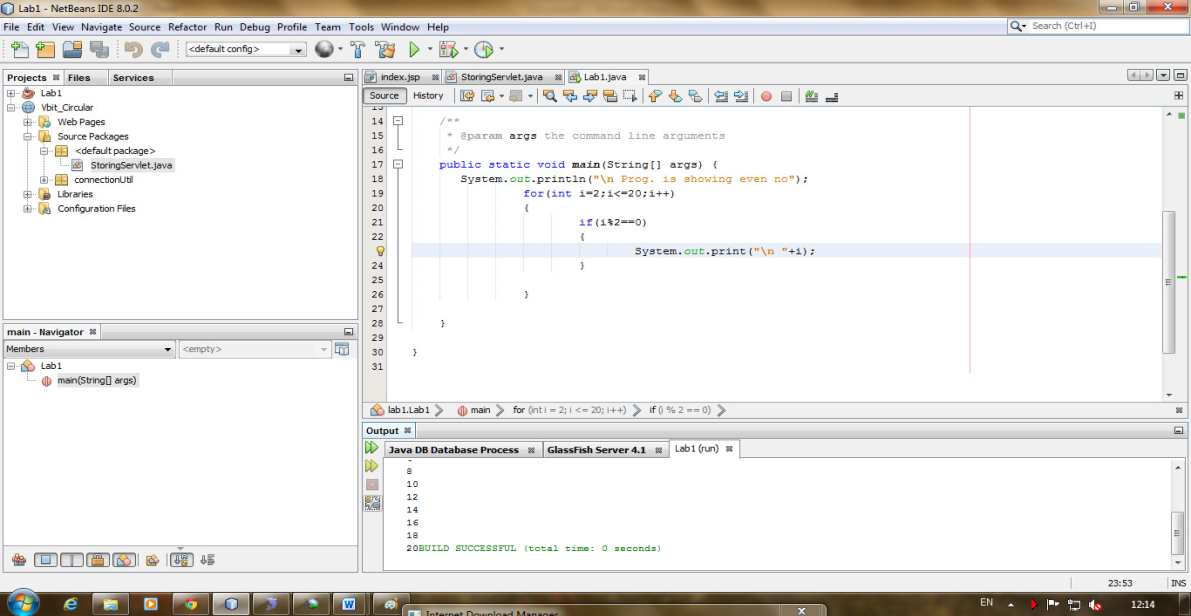
D:>javac Prog1.java

**Run:**

D:>java Prog1

**Output:**

**In Netbeans IDE:-**

****

**Week 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 divided by zero.

**Program:**

**import** java.awt.\*; **import** java.awt.event.\*; **import** java.applet.\*;

/\*

\* <applet code="Calculator" width=500 height=500></applet>

\* \*/

**public class** Calculator **extends** Applet **implements** ActionListener

{

String msg=" "; **int** v1,v2,result; TextField t1;

Button b[]=**new** Button[10];

Button add,sub,mul,div,clear,mod,EQ;

**char** OP;

**public void** init()

{

Color k=**new** Color(10,89,90); setBackground(k);

t1=**new** TextField(50);

GridLayout gl=**new** GridLayout(6,3); setLayout(gl);

**for**(**int** i=0;i<10;i++)

{

b[i]=**new** Button(""+i);

}

add=**new** Button("+"); sub=**new** Button("-"); mul=**new** Button("\*"); div=**new** Button("/"); mod=**new** Button("%"); clear=**new** Button("Clear"); EQ=**new** Button("="); t1.addActionListener(**this**)

;add(t1);

**for**(**int** i=0;i<10;i++)

{

add(b[i]);

}

add(add);

add(sub);

add(mul);

add(div);

add(mod);

add(clear); add(EQ);

**for**(**int** i=0;i<10;i++)

{

b[i].addActionListener(**this**);

}

add.addActionListener(**this**);

sub.addActionListener(**this**); mul.addActionListener(**this**); div.addActionListener(**this**);

mod.addActionListener(**this**); clear.addActionListener(**this**); EQ.addActionListener(**this**);

}

**public void** actionPerformed(ActionEvent ae)

{

String str=ae.getActionCommand();

**char** ch=str.charAt(0);

**if** ( Character.*isDigit*(ch)) t1.setText(t1.getText()+str);

##### else

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

{

v1=Integer.*parseInt*(t1.getText()); OP='+';

t1.setText("");

}

**else if**(str.equals("-"))

{

v1=Integer.*parseInt*(t1.getText()); OP='-'; t1.setText("");

}

**else if**(str.equals("\*"))

{

v1=Integer.*parseInt*(t1.getText()); OP='\*';

t1.setText("");

}

**else if**(str.equals("/"))

{

v1=Integer.*parseInt*(t1.getText()); OP='/';

t1.setText("");

}

**else if**(str.equals("%")){ v1=Integer.*parseInt*(t1.getText()); OP='%';

t1.setText("");

}

**if**(str.equals("=")){

v2=Integer.*parseInt*(t1.getText());

**if**(OP=='+')

result=v1+v2;

**else if**(OP=='-')

result=v1-v2;

**else if**(OP=='\*')

result=v1\*v2;

**else if**(OP=='/')

result=v1/v2;

**else if**(OP=='%')

result=v1%v2; t1.setText(""+result);

}

**if**(str.equals("Clear"))

{

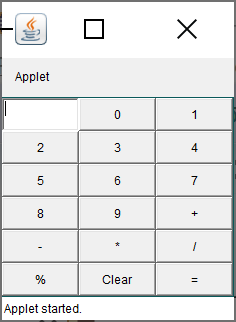
t1.setText("");

}

}

}

**Output:-**

****

**Week 3:**

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 factorial Value and returns it in another text field, when the button named “Compute” is clicked.

**a) Program:-**

// Import the packages to access the classes and methods in awt and applet classes.

**import** java.awt.\*;

**import** java.applet.\*;

/\* <applet code="Applet1" width=200 height=300></applet>\*/

**public class** Applet1 **extends** Applet

{

// Paint method to display the message.

**public void** paint(Graphics g)

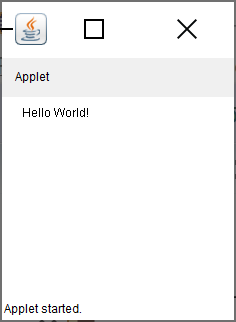
{

g.drawString("Hello World!",20,20);

}

}

**Output:-**

****

**b)Program:-**

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** java.applet.Applet;

/\*<applet code="Fact.class" height=300 width=300></applet>\*/

**public class** Factorial **extends** Applet **implements** ActionListener{ Label l1,l2;

TextField t1,t2; Button b1;

**public void** init(){

l1=**new** Label("Enter any integer value: "); add(l1);

t1=**new** TextField(5); add(t1);

b1=**new** Button("Calculate"); add(b1); b1.addActionListener(**this**);

l2=**new** Label("Factorial of given integer number is "); add(l2);

t2=**new** TextField(10); add(t2);

}

**public void** actionPerformed(ActionEvent e){

**if**(e.getSource()==b1){

**int** fact=fact(Integer.*parseInt*(t1.getText())); t2.setText(String.*valueOf*(fact));

}

}

**int** fact(**int** f) {

**int** s=0; **if**(f==0)

**return** 1;

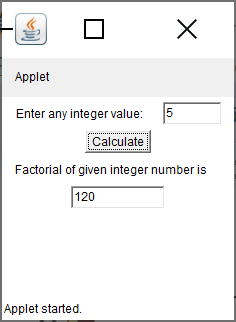
##### else

}

}

**return** f\*fact(f-1);

**Output:**



**Week 4:**

Write a Java 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 Num 2 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 Number Format Exception. If Num2 were Zero, the program would throw an Arithmetic Exception. Display the exception in a message dialog box..

**Source code:**

# ALGORITHMS:

1. create an applet from extending Applet class.
2. Draw a line using drawLine()method. 3.Draw rectangle using drawRec() method. 4.Draw an ovel using drawOvel()method.
3. Include all these methods in the paint()method.

#### Source code:

**import** java.awt.\*; **import** java.awt.event.\*; **import** java.applet.\*;

/\*<applet code="DivisionExample"width=230 height=250></applet>\*/

**public class** DivisionExample **extends** Applet **implements** ActionListener { String msg;

TextField num1, num2, res; Label l1, l2, l3;

Button div;

**public void** init()

{

l1 = **new** Label("Dividend");

l2 = **new** Label("Divisor");

l3 = **new** Label("Result");

num1 = **new** TextField(10);

num2 = **new** TextField(10);

res = **new** TextField(10);

div = **new** Button("Click"); 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();

**int** num1 = 0, num2 = 0;

**if** (arg.equals("Click"))

{

**if** (**this**.num1.getText().isEmpty()|**this**.num2.getText().isEmpty())

{

msg="Enter the valid numbers!"; repaint();

} **else** {

##### try {

num1 = Integer.*parseInt*(**this**.num1.getText()); num2 = Integer.*parseInt*(**this**.num2.getText());

**int** num3 = num1 / num2;

res.setText(String.*valueOf*(num3));

msg = "Operation Succesfull!!!"; repaint();

}

**catch** (NumberFormatException ex)

{

System.***out***.println(ex); res.setText("");

msg = "NumberFormatException-Non-numeric"; repaint();

}

**catch** (ArithmeticException e)

{

System.***out***.println("Can't be divided by Zero" + e); res.setText("");

msg = "Can't be divided by Zero"; repaint();

}

}

}

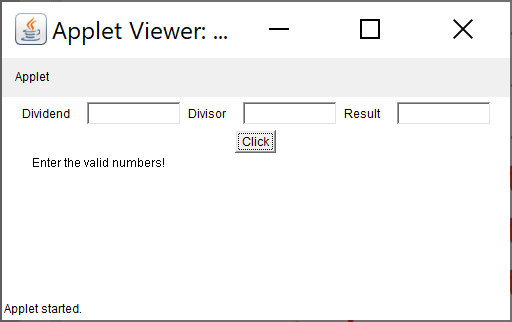
}

**public void** paint(Graphics g) { g.drawString(msg, 30, 70);

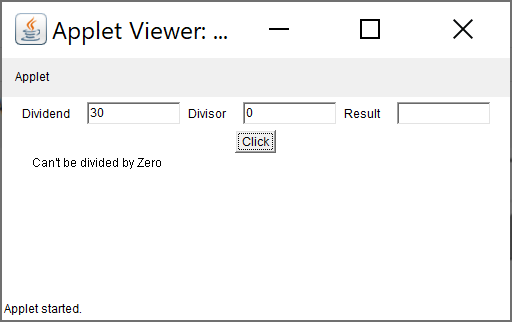
}

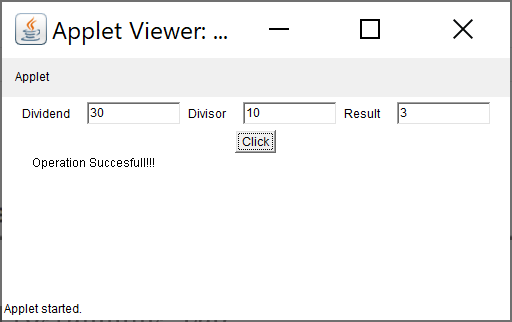
}

**Output:**



****

****

****

**Week 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.

**ALGORITHMS:**

## 1Create a thread class by implementing the Runnable interface.

2.Start the thread in the constructor. 3.implement the run() method.

## create multiple threads in the main()

## Display the threads

### **Source code:**

**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)

{

SquareThread sThread = **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** randomNubern)

{

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[]) {

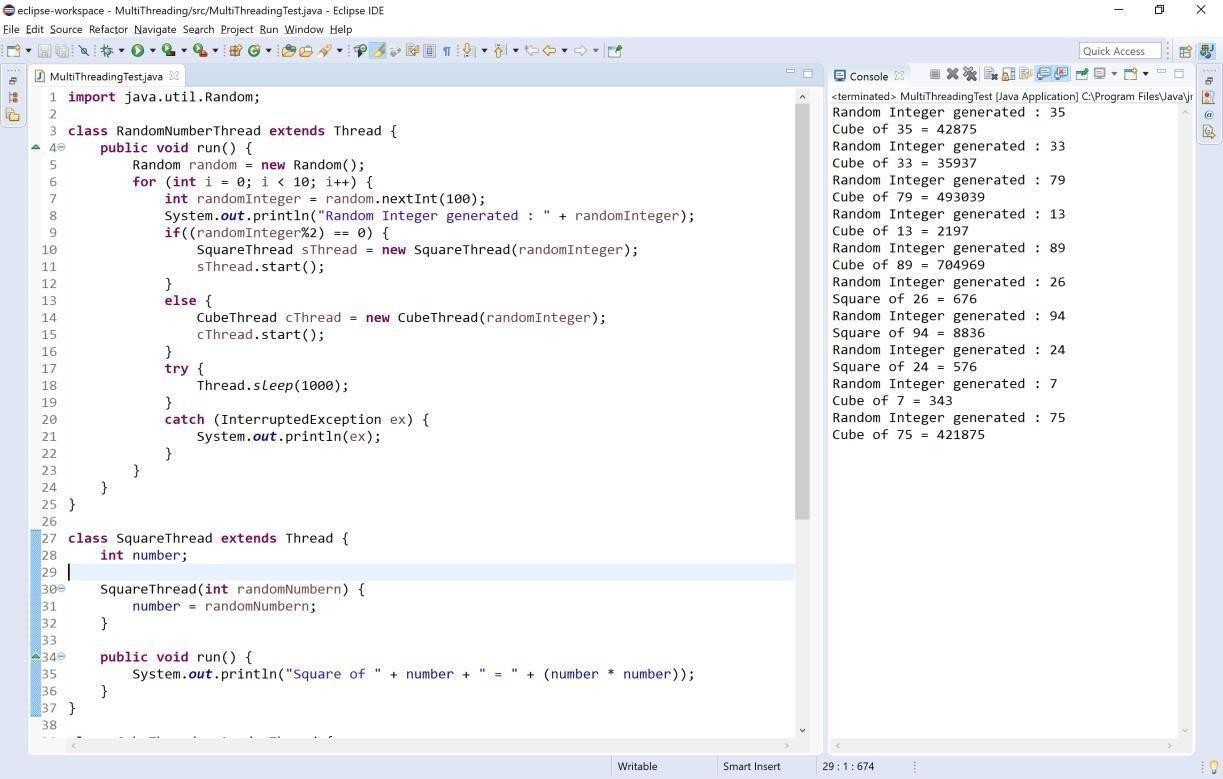
RandomNumberThread rnThread = **new** RandomNumberThread(); rnThread.start();

}

}

}

**output:**



**Week 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.

**Soure code:**

**public class** DoubleLinkedList

{

**class** Node

{

**int** data;

Node previous; Node next;

**public** Node(**int** data)

{

**this**.data = data;

}

}

Node head, tail = **null**;

**public void** addNode(**int** data)

{

Node newNode = **new** Node(data);

**if** (head==**null**)

{

head = tail = newNode; head.previous = **null**;

tail.next = **null**;

}

e**lse**

{

tail.next = newNode; newNode.previous=tail;

tail = newNode;

tail.next = **null**;

}

}

**public void** display()

{

Node current = head;

**if**(head == **null**)

{

System.***out***.println("List is empty");

*return;*

}

System.***out***.println("Nodes of doubly linked list: ");

**while**(current!=**null**)

{

System.***out***.print(current.data + " "); current = current.next;

}

}

**public static void** main(String[] args)

{

DoubleLinkedList dList = **new** DoubleLinkedList(); dList.addNode(1);

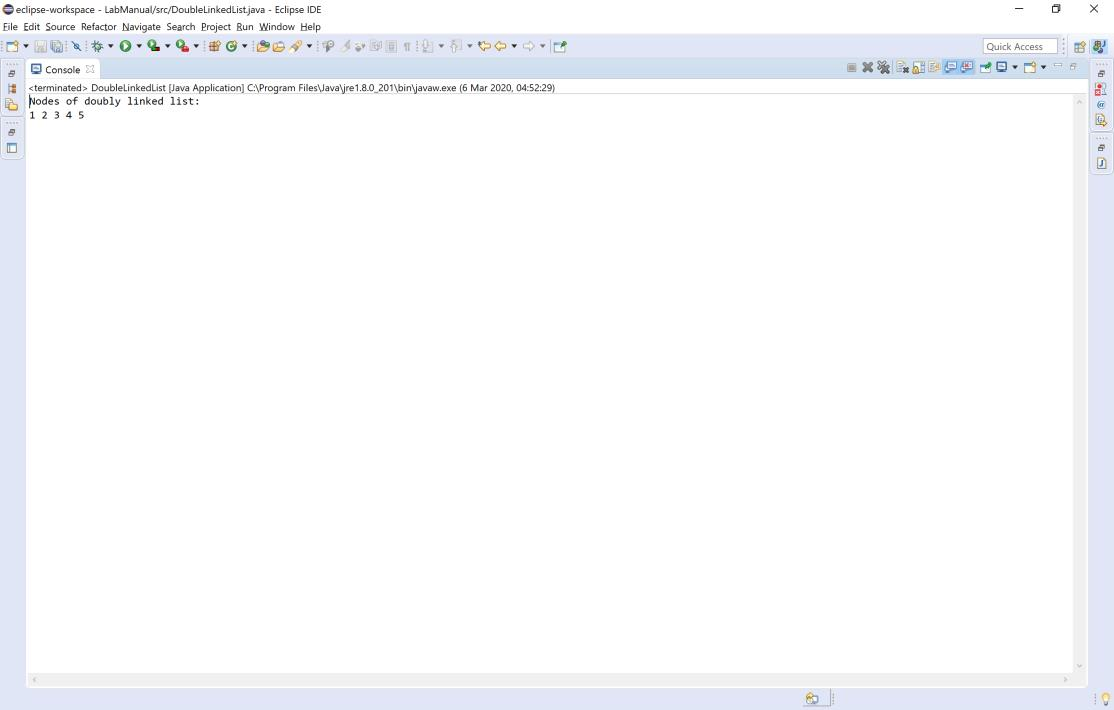
dList.addNode(2); dList.addNode(3); dList.addNode(4); dList.addNode(5);

dList.display();

}

}

**output:**



**Week 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.

**Source code:**

**import** java.applet.Applet;

**import** java.awt.\*;

**import** java.awt.event.\*;

//<applet code = "TrafficLights" width = 1000 height = 500></applet>

**public class** TrafficLights **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(ItemEvent ie)

{

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**)

{

}

##### else

{

yellowLight.setForeground(Color.***YELLOW***); msg.setForeground(Color.***YELLOW***); msg.setText("READY");

greenLight.setForeground(Color.***GREEN***); msg.setForeground(Color.***GREEN***); msg.setText("GO");

}

}

}

**Output**:



**Week 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.

**Source code:**

**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** AbstractClass

{

**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:**

C:\Users\sriindu\Desktop\sasi>javac AbstractClass.java C:\Users\sriindu\Desktop\sasi>java AbstractClass

\*\*\* Finding the Area of Rectangle \*\*\* Enter length and breadth: 2 3

The area of Rectangle is: 6

\*\*\* Finding the Area of Triangle \*\*\* Enter Base And Height: 3 3

The area of Triangle is: 4

\*\*\* Finding the Area of Cricle \*\*\* Enter Radius: 3

The area of Cricle is: 28.26 C:\Users\sriindu\Desktop\sasi>

**Week 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.

**Source code:**

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\\eclipse-workspace\\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)

{

System.out.println ("Exception raised" +e.toString());

}

setDefaultLookAndFeelDecorated(true);

pack();

setVisible(true);

}

}

public class TableTest

{

public static void main(String[] args)

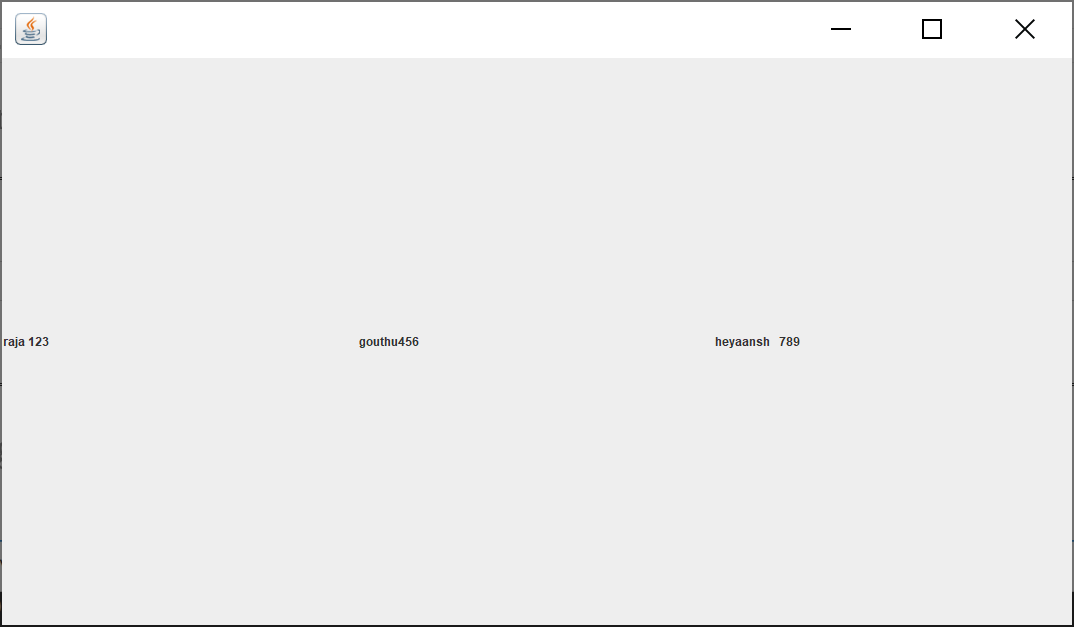
{

A a = new A();

}

}

**Output**

****

**Week 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).

**Source code:**

**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();

}

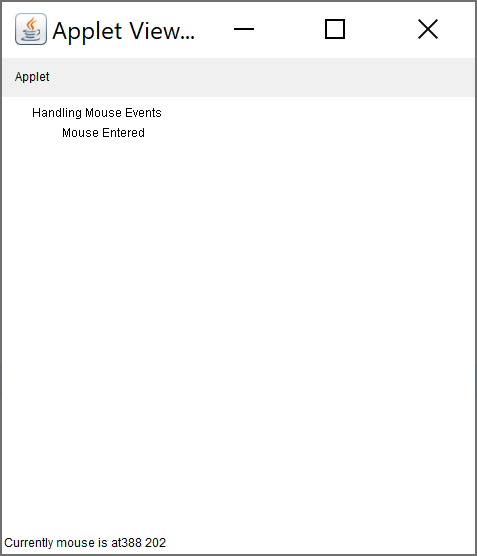
**public void** paint(Graphics g) { g.drawString("Handling Mouse Events", 30, 20);

g.drawString(msg, 60, 40);

}

}

**Output:**

****

**Week 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)**

**Source code:**

**import** java.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))

{

}

##### else

{

output = hashData.get(input);

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** {

System.***out***.println("Data not found in HashTable");

}

}

**private** Hashtable<String, String> readFromFile(String fileName)

{

Hashtable<String, String> hashData = **new** Hashtable<String,String>();

##### try

{

File f = **new** File("C:\\Users\\sriindu\\Desktop\\sasi\\" + fileName); BufferedReader br = **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();

}

**return** hashData;

}

}

**HashTab.txt**

raja 123

rani 456

anu 789

### Output:

C:\Users\sriindu\Desktop\sasi>javac HashTab.java C:\Users\sriindu\Desktop\sasi>java HashTab

File data into Hashtable:

{anu=789, rani=456, raja=123} Input given : raja

Data found in HashTable : 123 Input given : 123

Data found in HashTable : raja Input given:

Data not found in HashTable

**Week – 12**

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

**Source Code:**

**class** ItemQueue {

**int** item;

**boolean** valueSet= **false**; **synchronized int** getItem()

{

**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** (InterruptedException 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

{

ItemQueue itemQueue; Producer(ItemQueue itemQueue)

{

**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(ItemQueue itemQueue)

{

**this**.itemQueue = itemQueue;

**new** Thread(**this**, "Consumer").start();

}

**public void** run()

{

##### while(true)

{

itemQueue.getItem();

}

}

}

**class** ProducerConsumer

{

**public static void** main(String args[])

{

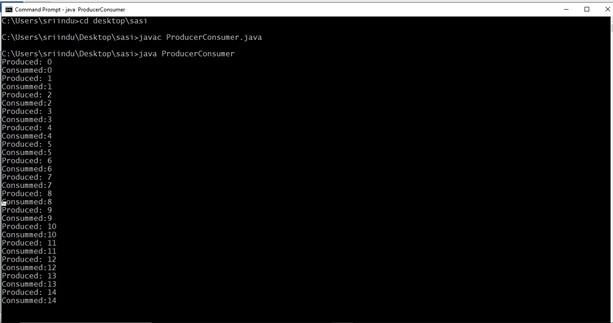
ItemQueue itemQueue = **new** ItemQueue();

**new** Producer(itemQueue);

**new** Consumer(itemQueue);

}

}

**Output:**

**Week – 13**

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

**Source Code:**

import java.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();

int lastIndex = 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());

}

}

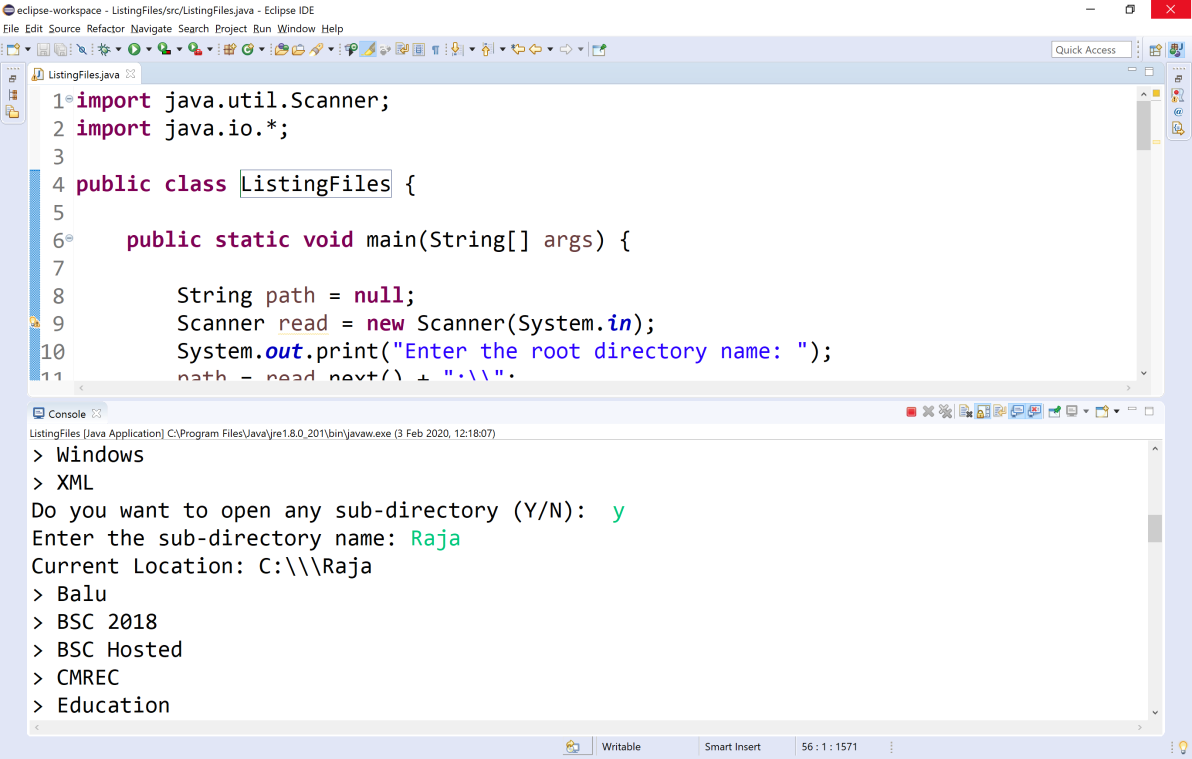
public static void printLine() {

System.out.println("----------------------------------------");

}

}

**output:**

****