

# Lab Manual

# Java Programming

Computer Science & Engineering | Information Technology  
(II- B. Tech. – II- Semester)



## Regulation R18

Visit – [www.btechsmartclass.com](http://www.btechsmartclass.com) to get more like

- ✓ Study Materials
- ✓ Presentations
- ✓ Video Lectures and many more

## 1. Syllabus

### CS408PC: JAVA PROGRAMMING LAB

**B.Tech. II Year II Sem.**

**L T P C**  
**0 0 2 1**

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.
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.
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.
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.
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.
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.
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.
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.
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.
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).
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).
12. Write a Java program that correctly implements the producer – consumer problem using the concept of interthread communication.
13. Write a Java program to list all the files in a directory including the files present in all its subdirectories.
14. Write a Java program that implements Quick sort algorithm for sorting a list of names in ascending Order.
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 OUTCOMES (PO's)**

<b>PO No.</b>	<b>Program Outcomes (PO's)</b>
PO1	An ability to apply knowledge of computing, mathematics, science and engineering fundamentals appropriate to the discipline.
PO2	An ability to analyze a problem, and identify and formulate the computing requirements appropriate to its solution.
PO3	An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.
PO4	An ability to design and conduct experiments, as well as to analyze and interpret data.
PO5	An ability to use current techniques, skills, and modern tools necessary for computing practice.
PO6	An ability to analyze the local and global impact of computing on individuals, organizations, and society.
PO7	Knowledge of contemporary issues.
PO8	An understanding of professional, ethical, legal, security and social issues and responsibilities.
PO9	An ability to function effectively individually and on teams, including diverse and multidisciplinary, to accomplish a common goal.
PO10	An ability to communicate effectively with a range of audiences.
PO11	An understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects.
PO12	Recognition of the need for and an ability to engage in continuing professional development.

### 3.Lesson/Course Plan

Week No.	Name of the Program	No. of Hours required	Text Books	Mode of Assessment
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.	1	1	Viva&Execution
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.	1	T1	Viva&Execution
3	<p><b>a)</b> Develop an applet in Java that displays a simple message.</p> <p><b>b)</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.</p>	1	T1	Viva&Execution
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.	1	T1	Viva&Execution
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.	1	T1	Viva&Execution
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.	1	T1	Viva&Execution
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.	1	T1	Viva&Execution
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	1	T1	Viva&Execution

	shape.			
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.	1	T1	Viva&Execution
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).	1	T1	Viva&Execution
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).	1	T1	Viva&Execution
12	Write a Java program that correctly implements the producer – consumer problem using the concept of interthread communication.	1	T1	Viva&Execution
13	Write a Java program to list all the files in a directory including the files present in all its subdirectories.	1	T1	Viva&Execution
14	Write a Java program that implements Quick sort algorithm for sorting a list of names in ascending Order	1	T1	Viva&Execution
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.	1	T1	Viva&Execution
16	LEAD Experiments	1	T1	Viva&Execution
	<b>Total no of HOURS required to complete syllabus</b>	<b>16</b>		

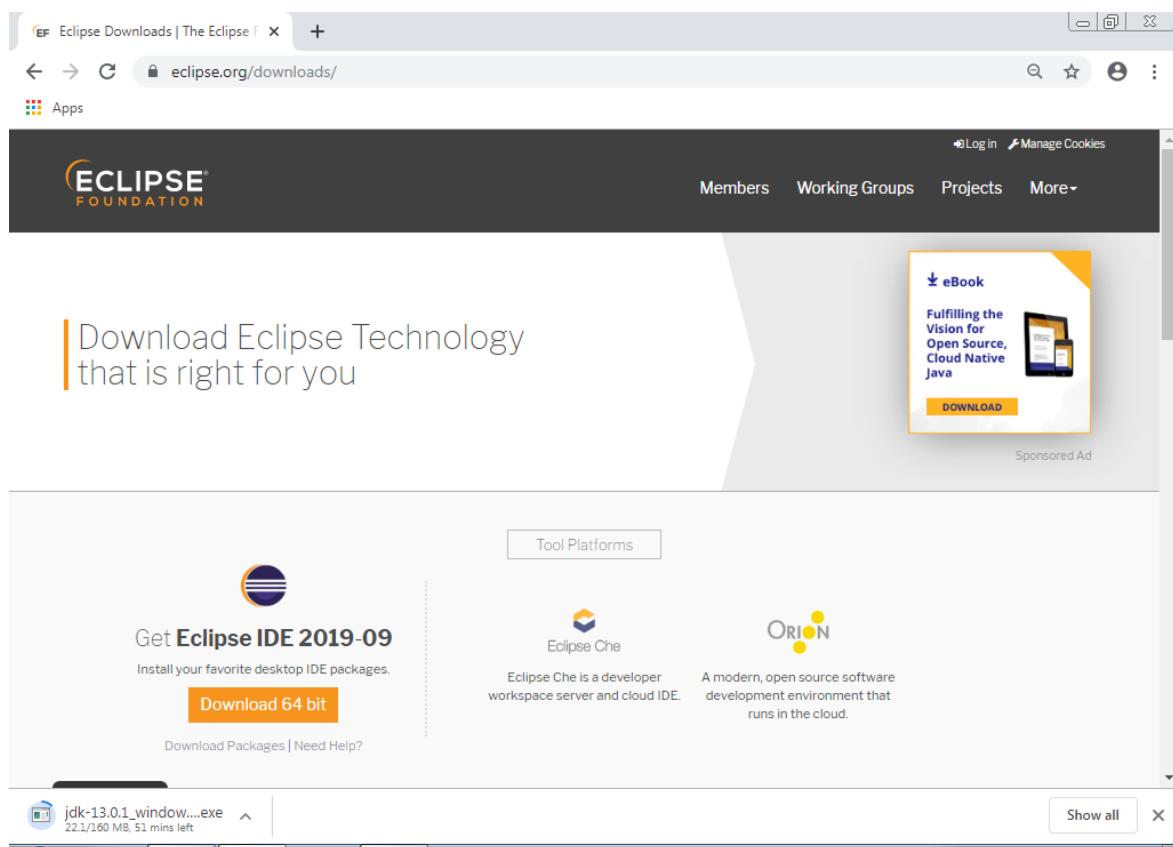
## PROGRAMS

### Week 1.

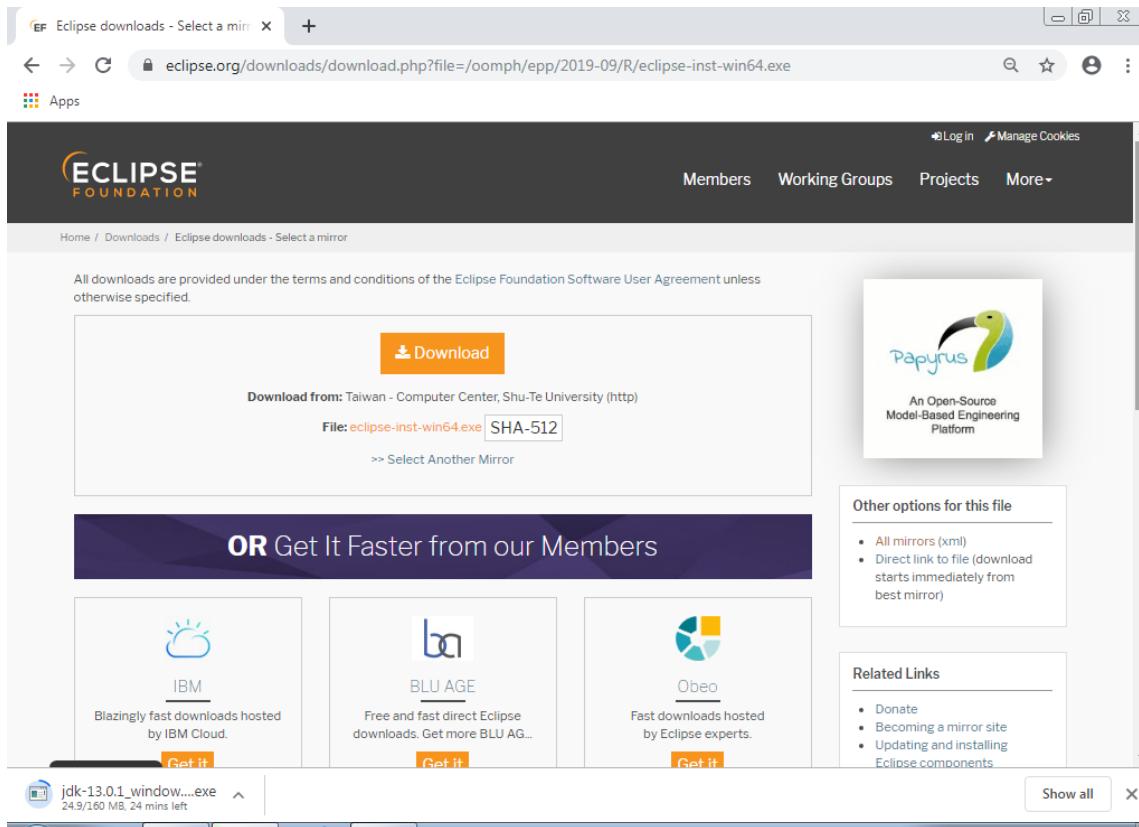
**Aim:** Use Eclipse or Net bean platform and acquaint with the various menus. Create a test project, add a test class, and run it. See how you can use auto suggestions, auto fill. Try code formatter and code refactoring like renaming variables, methods, and classes. Try debug step by step with a small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

### Solution:

- **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)

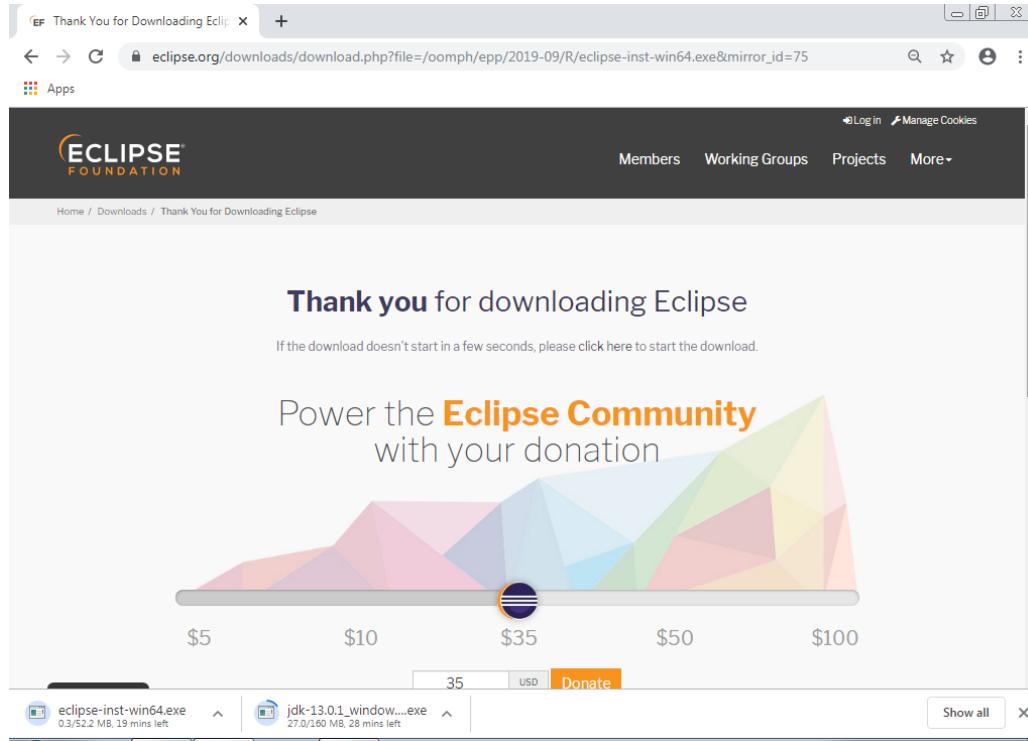


Select the suitable version based on your OS.



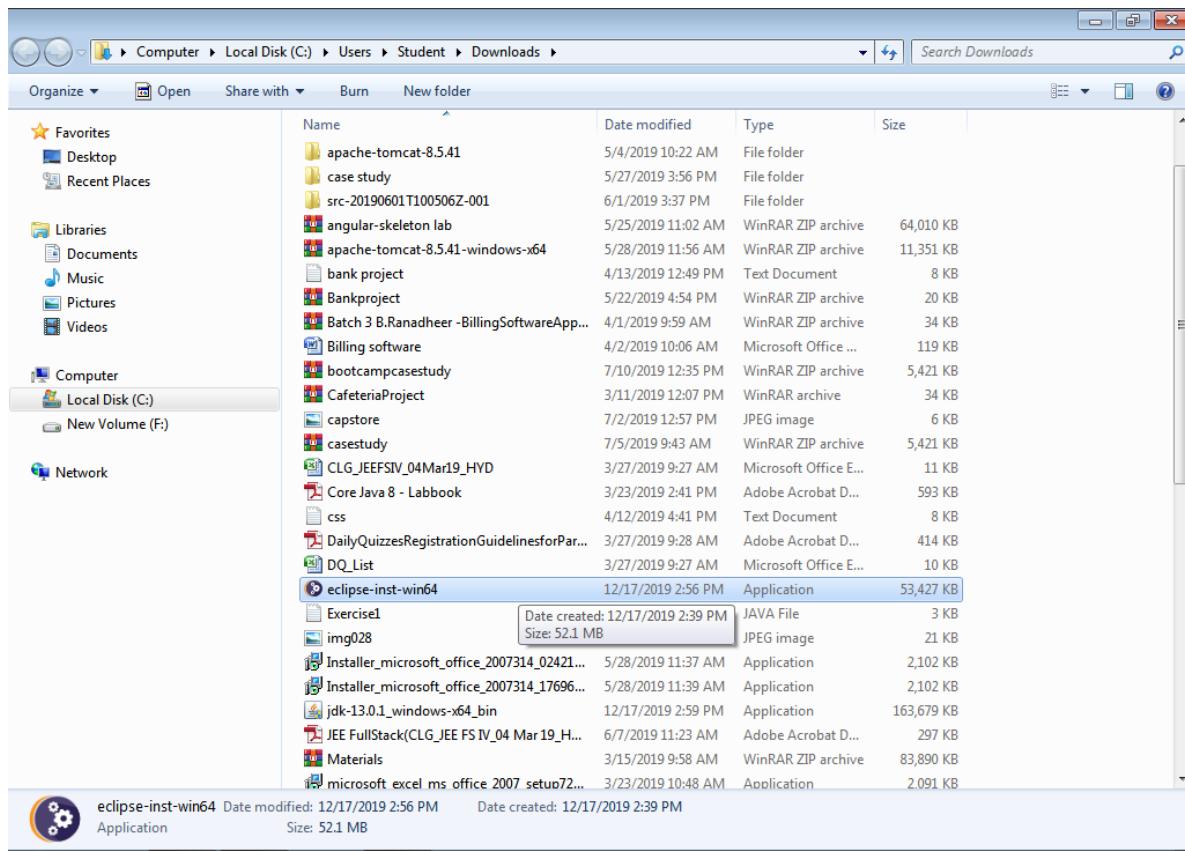
The screenshot shows the Eclipse Foundation's Downloads page. At the top, there is a message: "All downloads are provided under the terms and conditions of the Eclipse Foundation Software User Agreement unless otherwise specified." Below this, there is a large orange "Download" button. To its right, it says "Download from: Taiwan - Computer Center, Shu-Te University (http)". Underneath, it shows the file name "File: eclipse-inst-win64.exe" and a SHA-512 hash. There is also a link to "Select Another Mirror". To the right of the main download area, there is a sidebar for "Papyrus" which is described as "An Open-Source Model-Based Engineering Platform". Below the main area, there is a section titled "OR Get It Faster from our Members" featuring links to IBM, BLU AGE, and Obeo, each with their own download buttons. On the far left, there is a progress bar for a download of "jdk-13.0.1\_windows-x64.exe" at 24.9/160 MB, 24 mins left. On the far right, there are sections for "Other options for this file" and "Related Links".

Then download get starts.

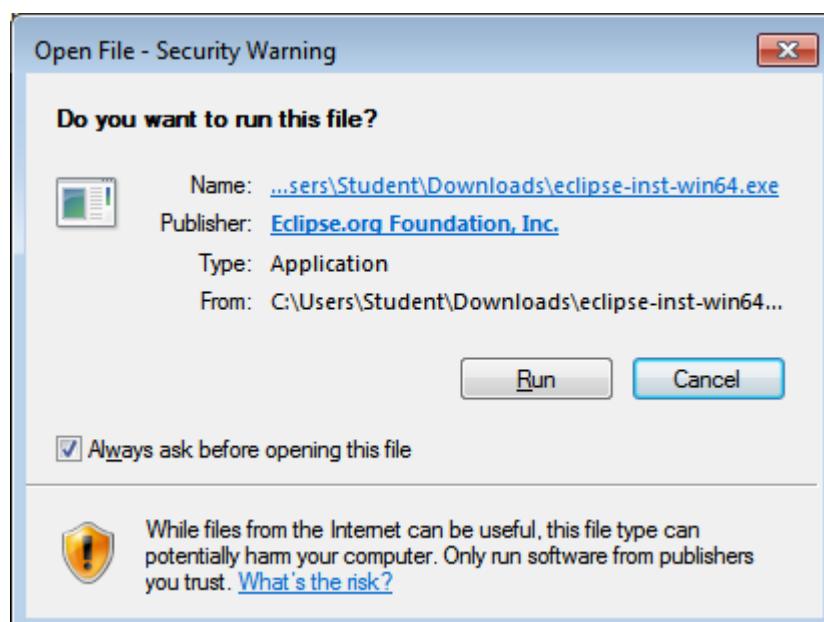


The screenshot shows the Eclipse Foundation's Downloads page after a download has completed. The main message is "Thank you for downloading Eclipse". Below this, it says "If the download doesn't start in a few seconds, please click here to start the download." A large graphic features the text "Power the **Eclipse Community** with your donation" above a stylized polygonal shape. Below the shape is a horizontal slider with a circular dial set to \$35 USD. At the bottom of the slider, there is a "Donate" button. On the far left, there is a progress bar for a download of "eclipse-inst-win64.exe" at 0.3/52.2 MB, 19 mins left. On the far right, there is a progress bar for a download of "jdk-13.0.1\_windows-x64.exe" at 27.0/160 MB, 28 mins left. The "Show all" button is visible on both the left and right progress bars.

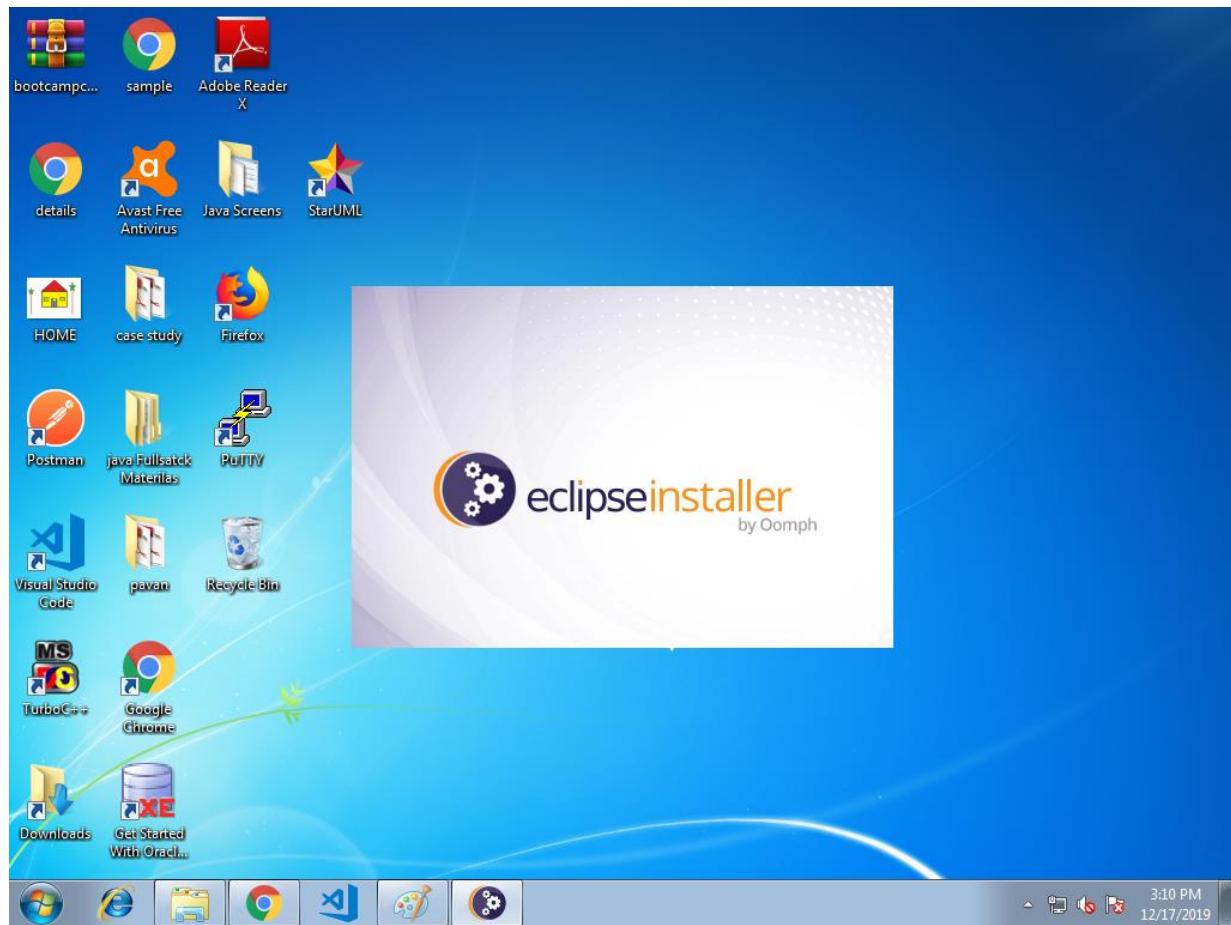
Double click on the Eclipse Application.



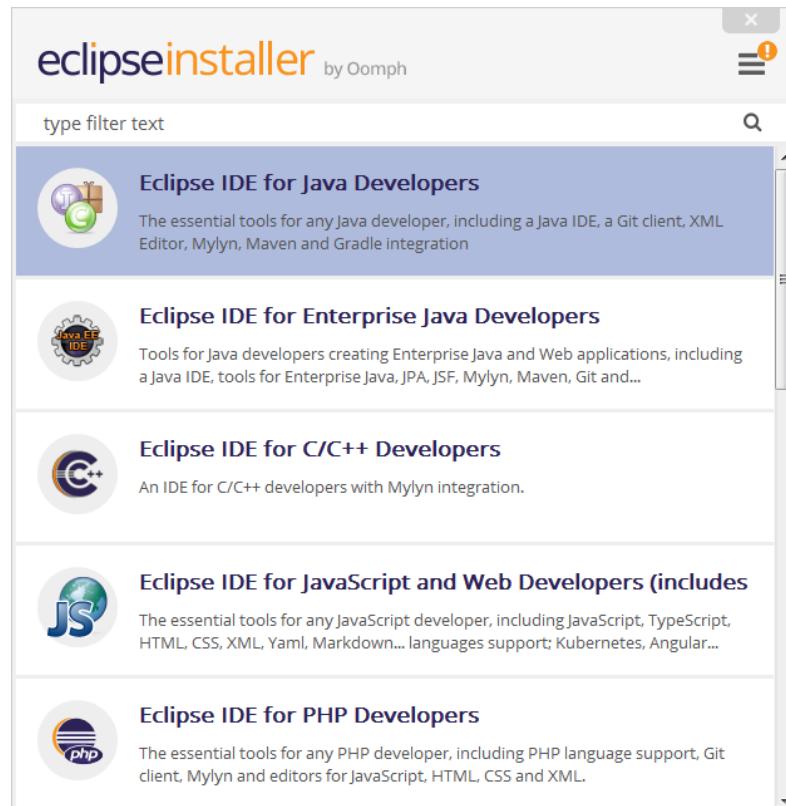
Click on Run in the Security Warning box.



Then, the installation process begins.



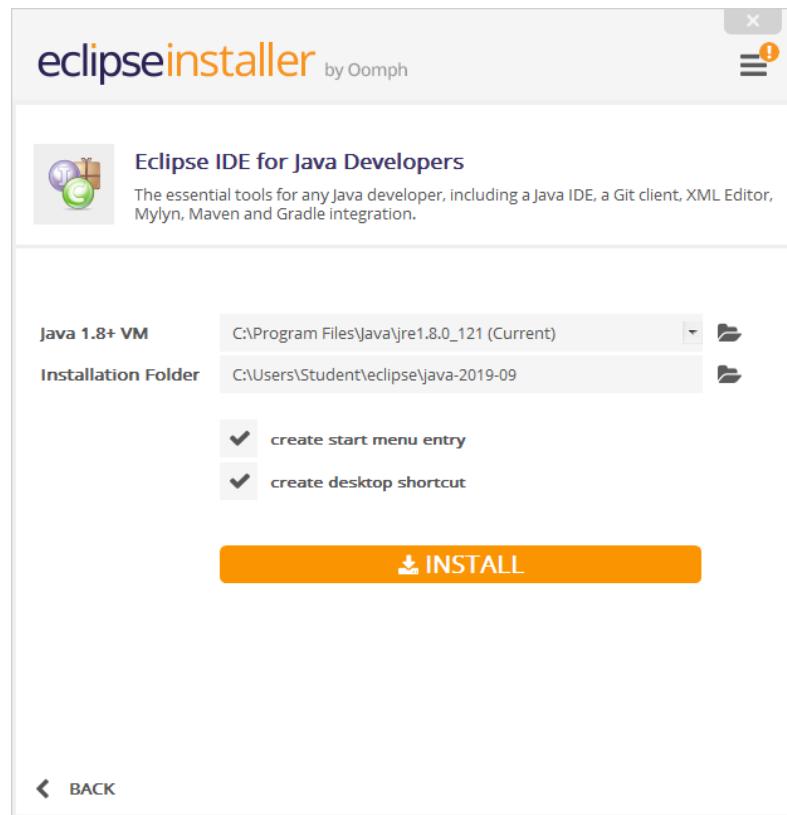
Click on Eclipse IDE for Java Developers.


 A screenshot of the Eclipse Installer application window. The title bar reads "eclipseinstaller by Oomph". Below it is a search bar with the placeholder "type filter text" and a magnifying glass icon. The main content area displays five sections, each representing a different Eclipse IDE edition:
 

- Eclipse IDE for Java Developers**: Described as "The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration".
- Eclipse IDE for Enterprise Java Developers**: Described as "Tools for Java developers creating Enterprise Java and Web applications, including a Java IDE, tools for Enterprise Java, JPA, JSF, Mylyn, Maven, Git and...".
- Eclipse IDE for C/C++ Developers**: Described as "An IDE for C/C++ developers with Mylyn integration."
- Eclipse IDE for JavaScript and Web Developers (includes)**: Described as "The essential tools for any JavaScript developer, including JavaScript, TypeScript, HTML, CSS, XML, Yaml, Markdown... languages support; Kubernetes, Angular...".
- Eclipse IDE for PHP Developers**: Described as "The essential tools for any PHP developer, including PHP language support, Git client, Mylyn and editors for JavaScript, HTML, CSS and XML."

 The "Eclipse IDE for Java Developers" section is currently highlighted with a blue background, indicating it is the selected edition.

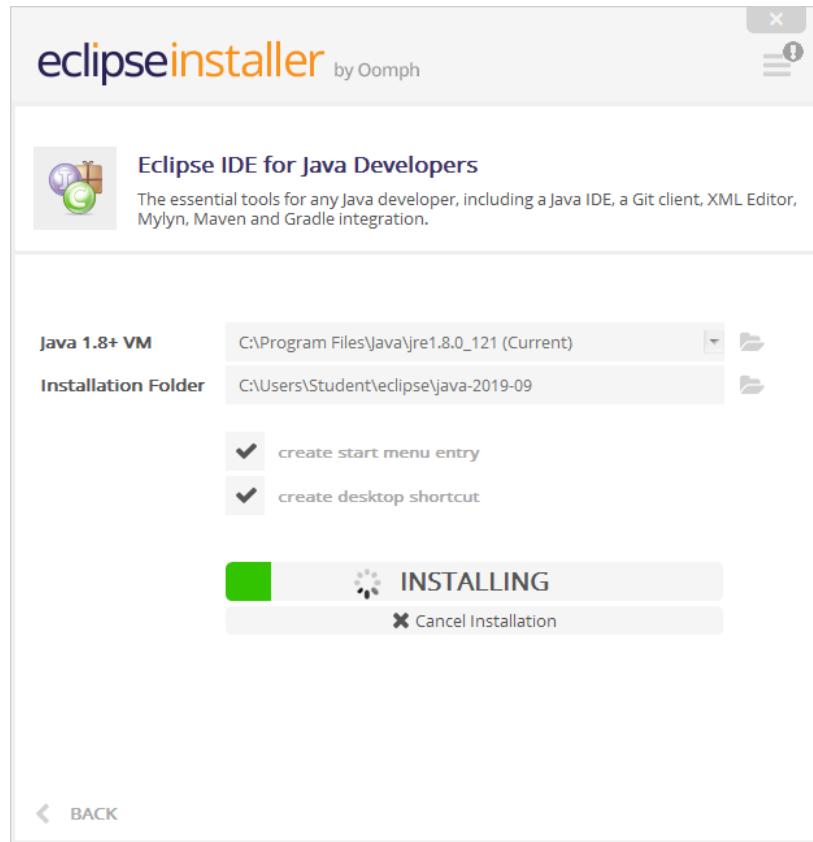
Click on Install button.



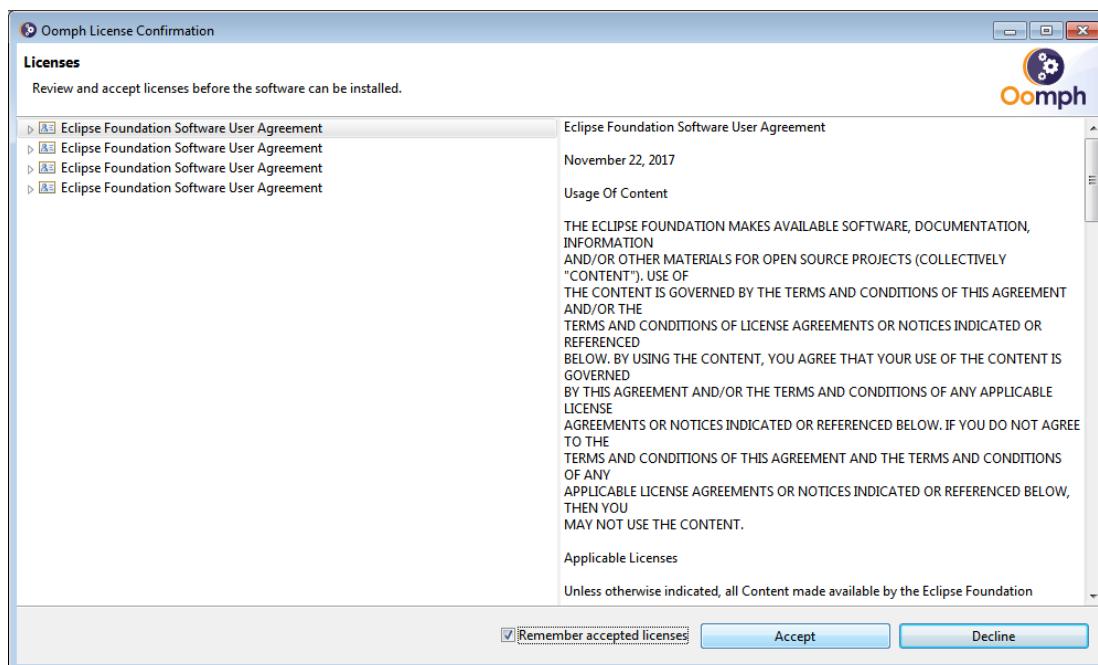
Click on Accept Now.

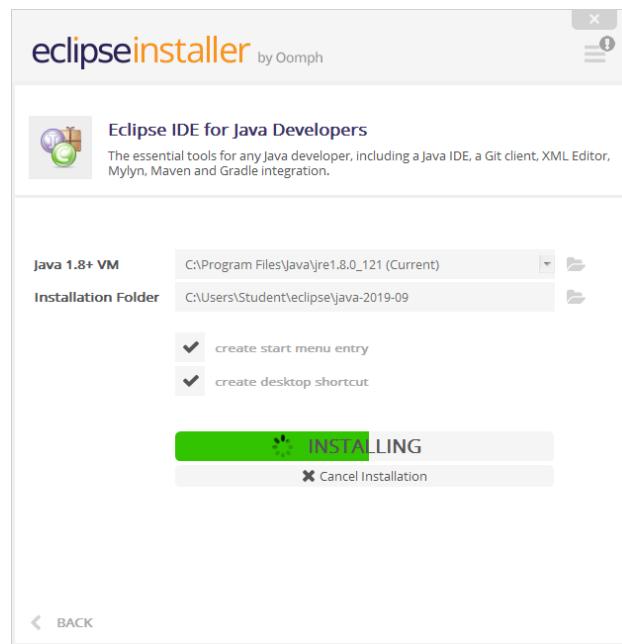


Then the Eclipse installation begins.

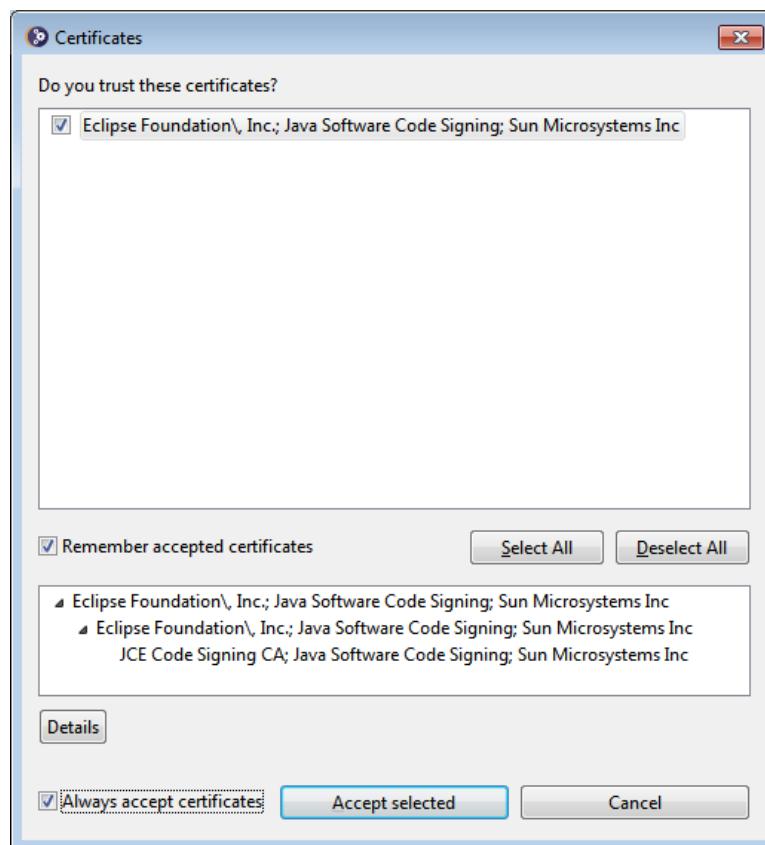


Click on Accept

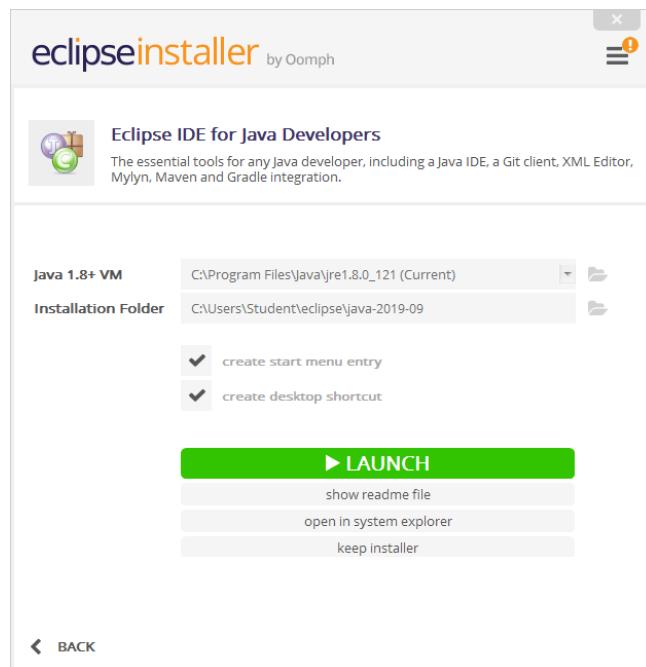




Click on Select All and Accept Selected.

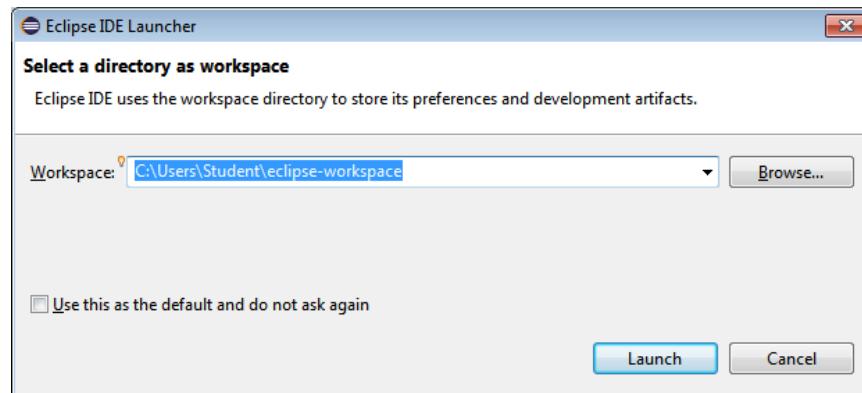


After completing, click on Launch to start the Eclipse IDE.

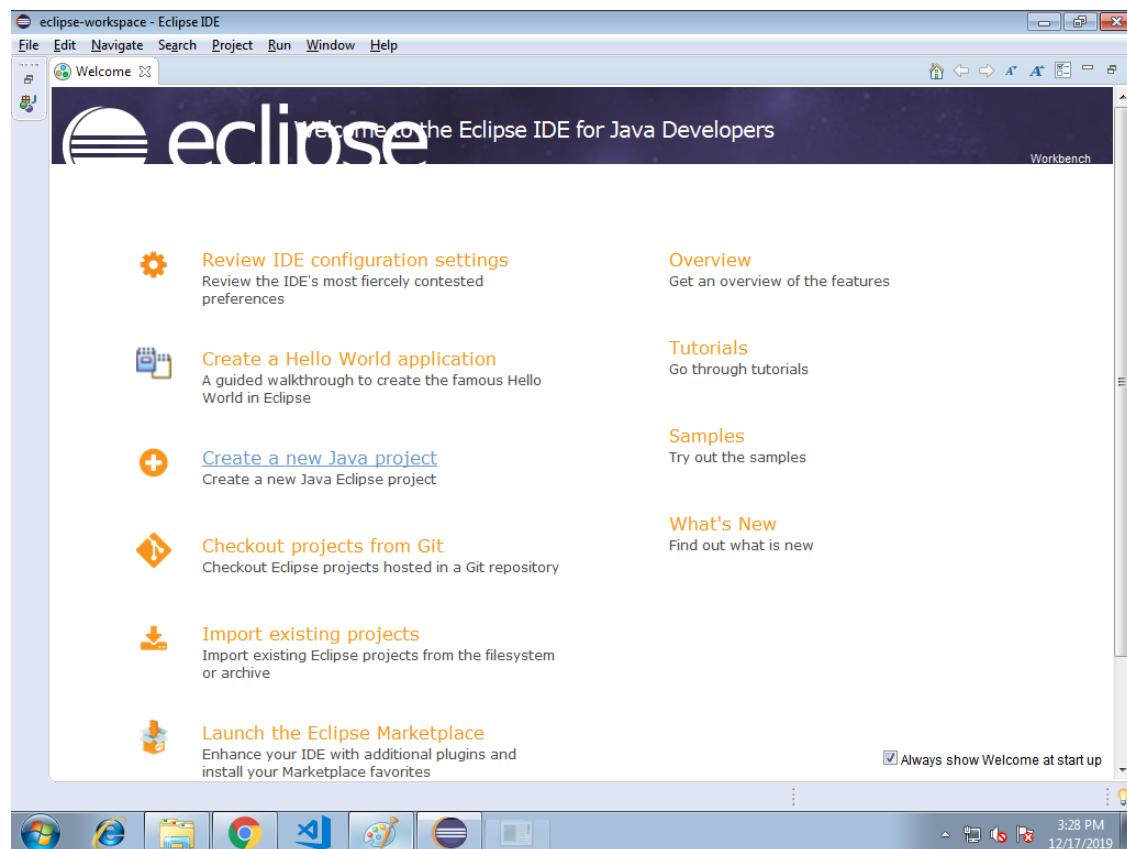


## CREATING PROJECT AND CLASSES IN ECLIPSE IDE

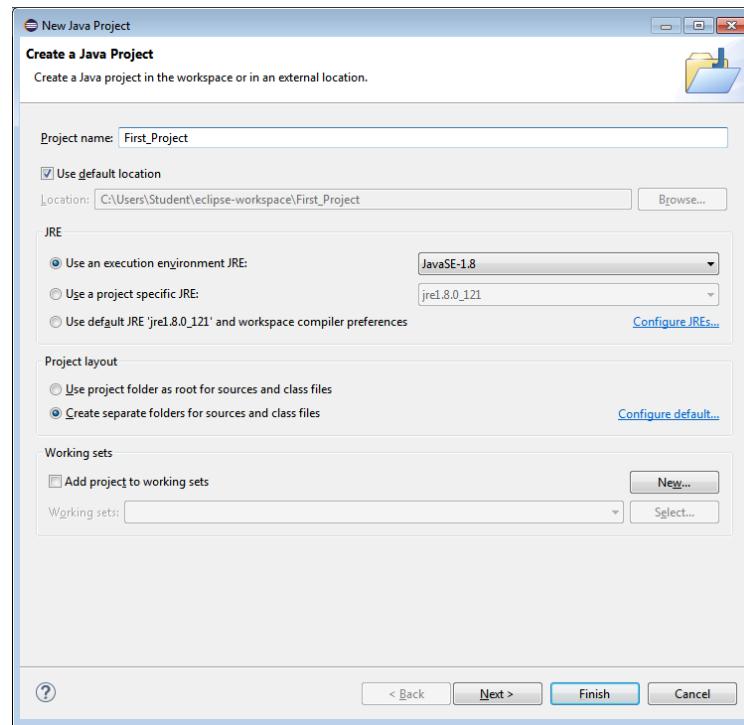
Browse the Workspace for storing the java project and click on Launch.



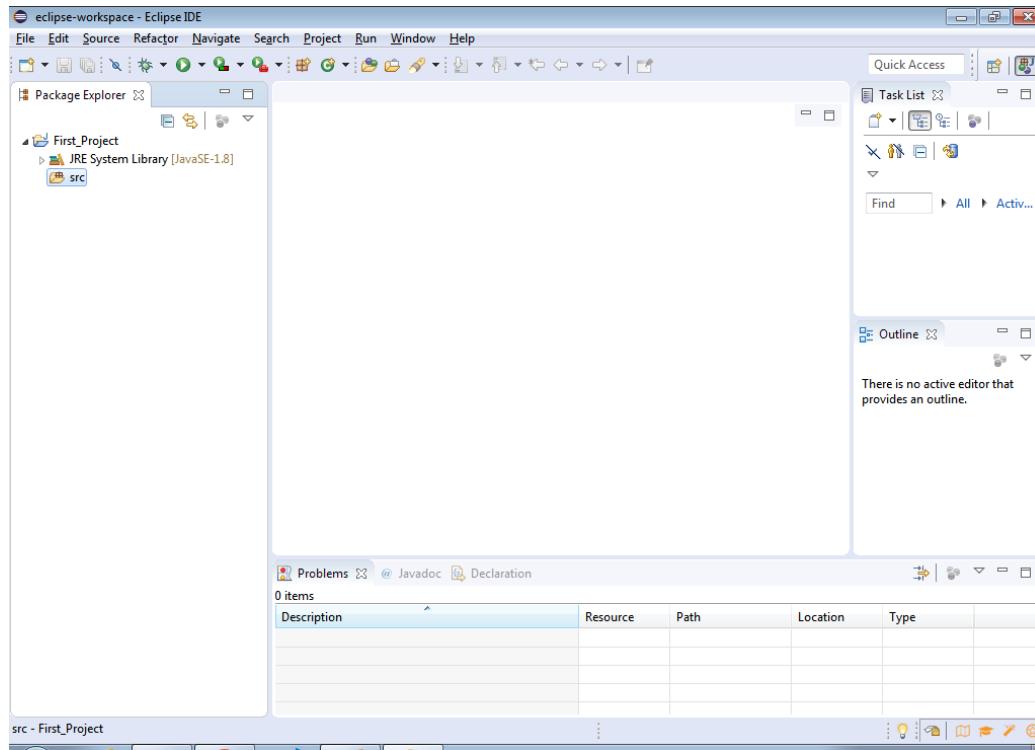
Select "Create a new Java project".

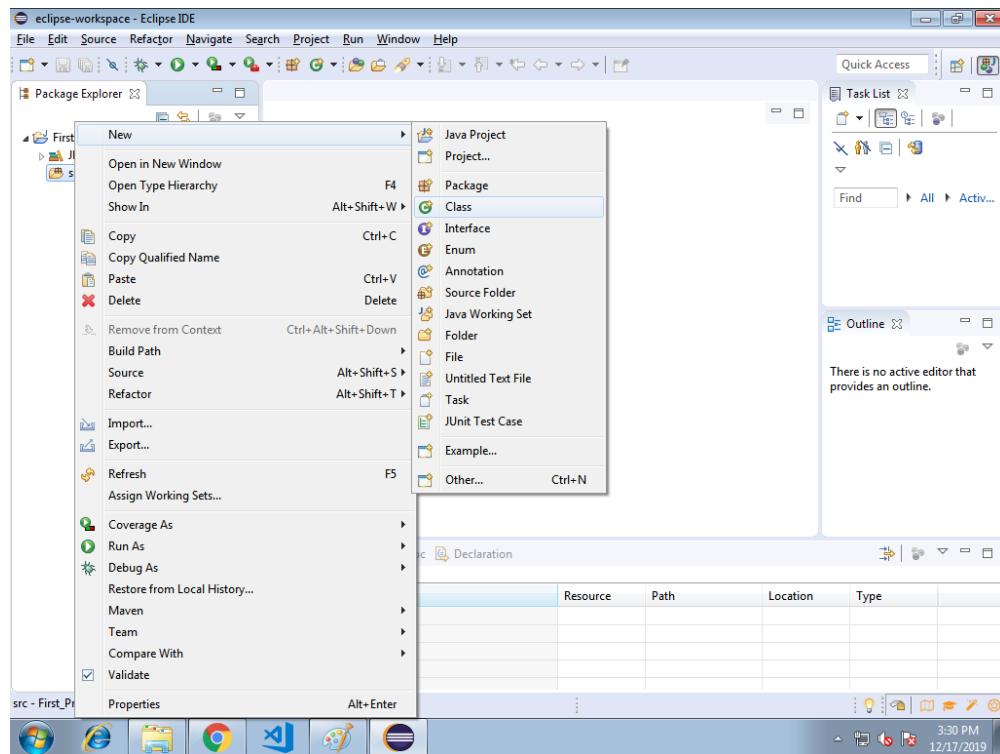


Type the project name and click on Finish.

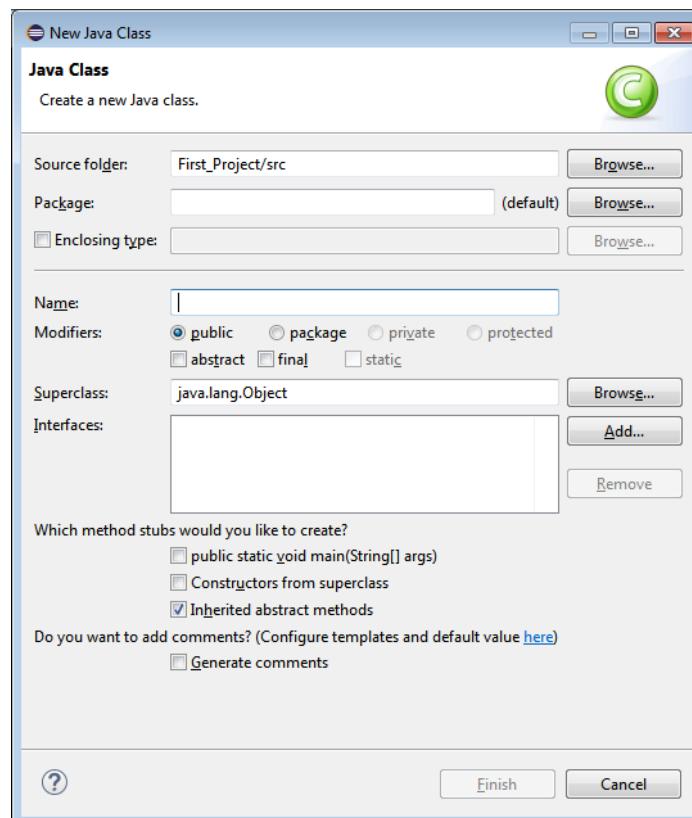


Now, create the class in src directory from Package Explorer window.

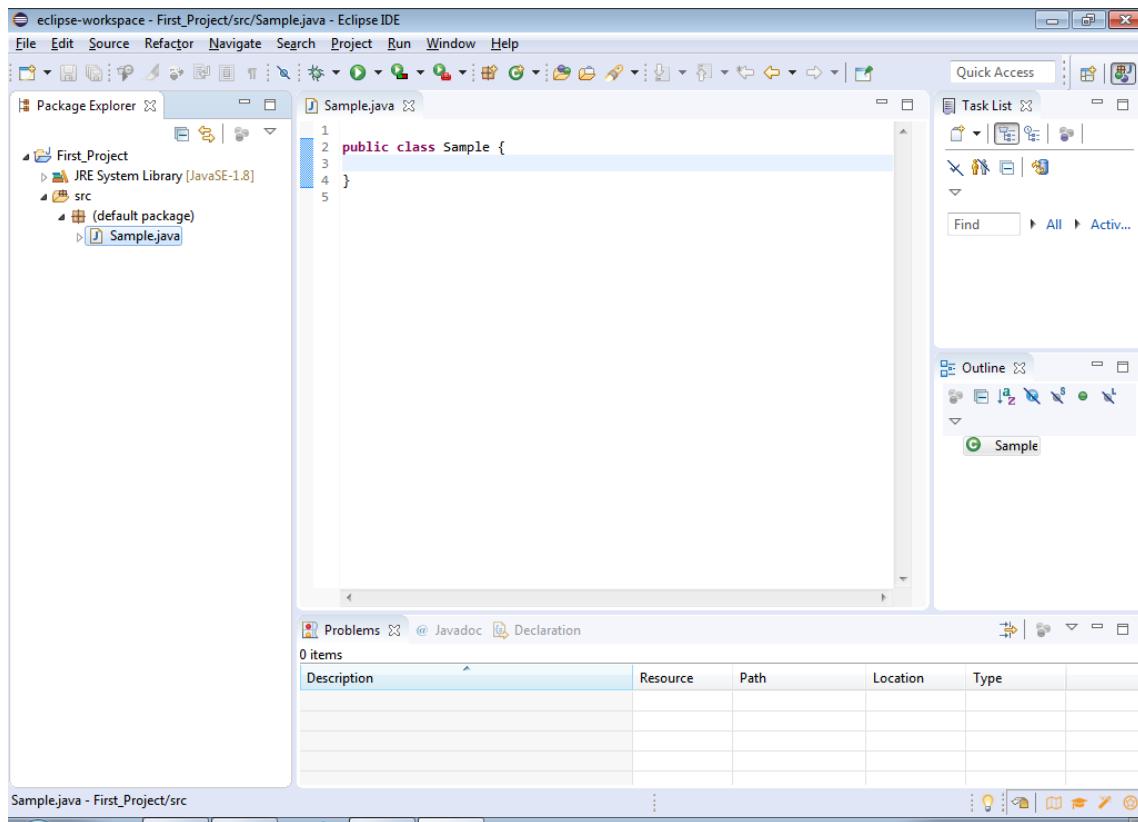




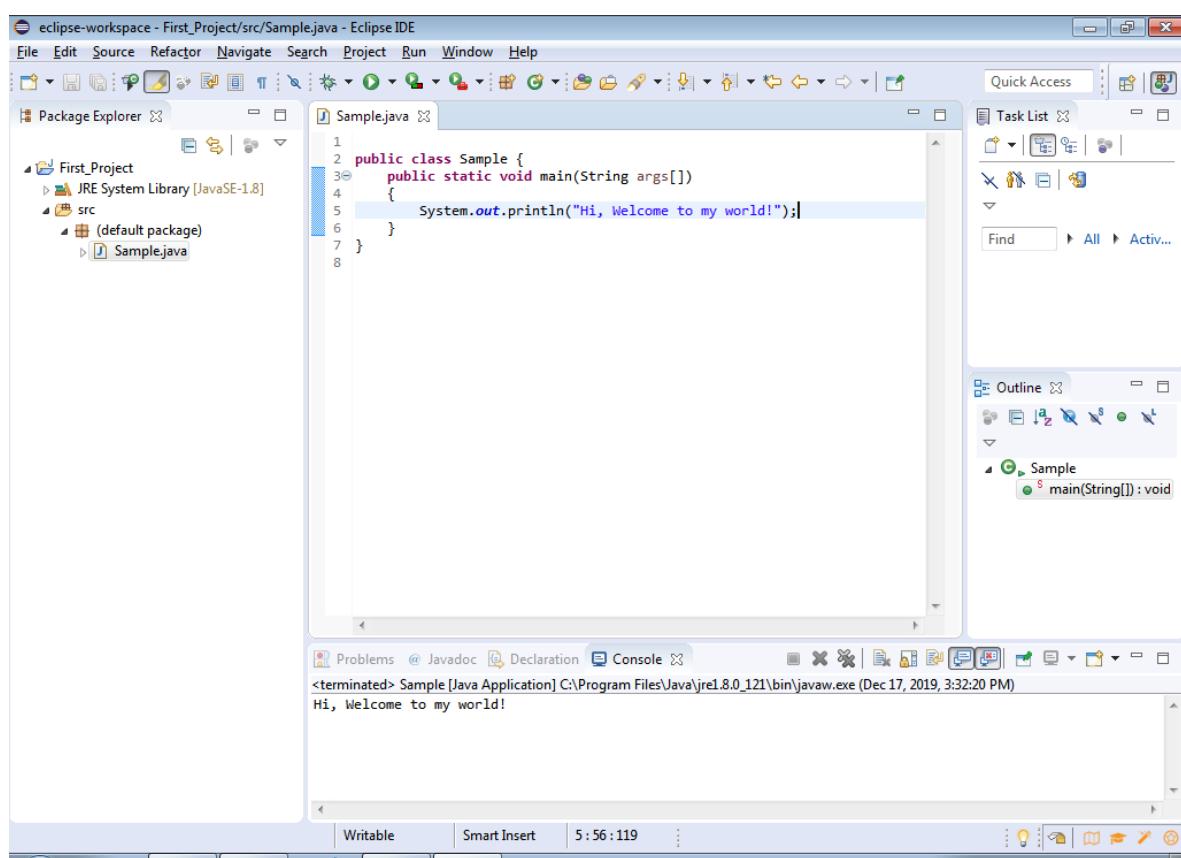
Type the class name and click on Finish.



Type the java code.



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



## 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.

### Source Code:

```

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);
    }
}

```

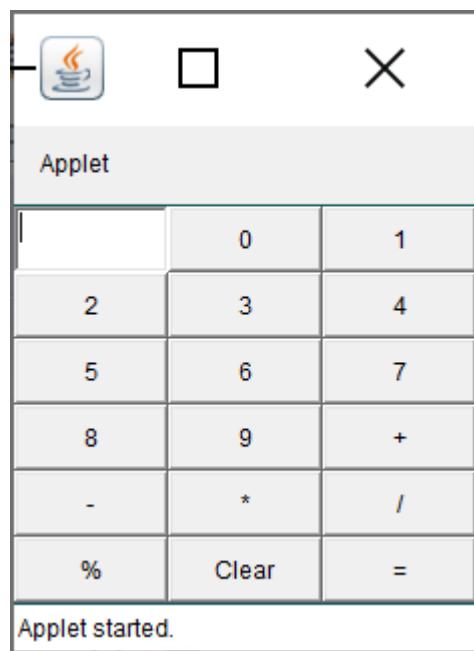
```

        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:**

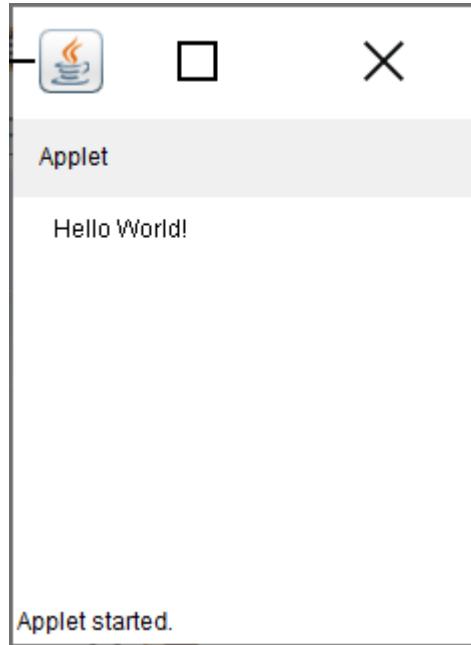
- a) Develop an applet in Java that displays a simple message.  
b) Develop an applet in Java that receives an integer in one text field, and computes its factorial Value and returns it in another text field, when the button named “Compute” is clicked.

**Source code for question a:**

```
// 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 AppletExample extends Applet
{
    // Paint method to display the message.
    public void paint(Graphics g)
    {
        g.drawString("Hello World!",20,20);
    }
}
```

**Output:**

### Source code for question b:

```

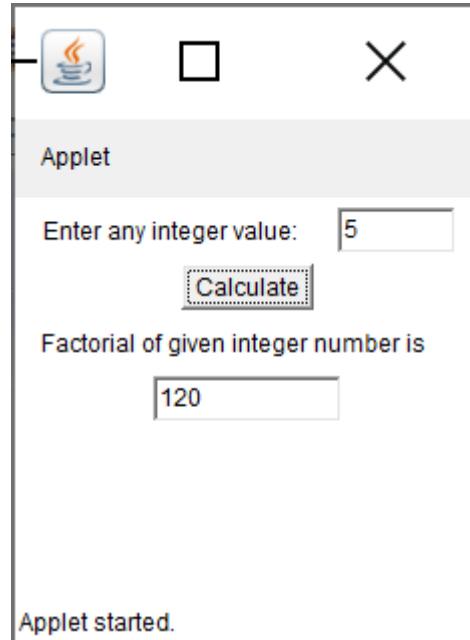
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;

/*<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 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 Number Format Exception. If Num2 were Zero, the program would throw an Arithmetic Exception. Display the exception in a message dialog box.

### 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());

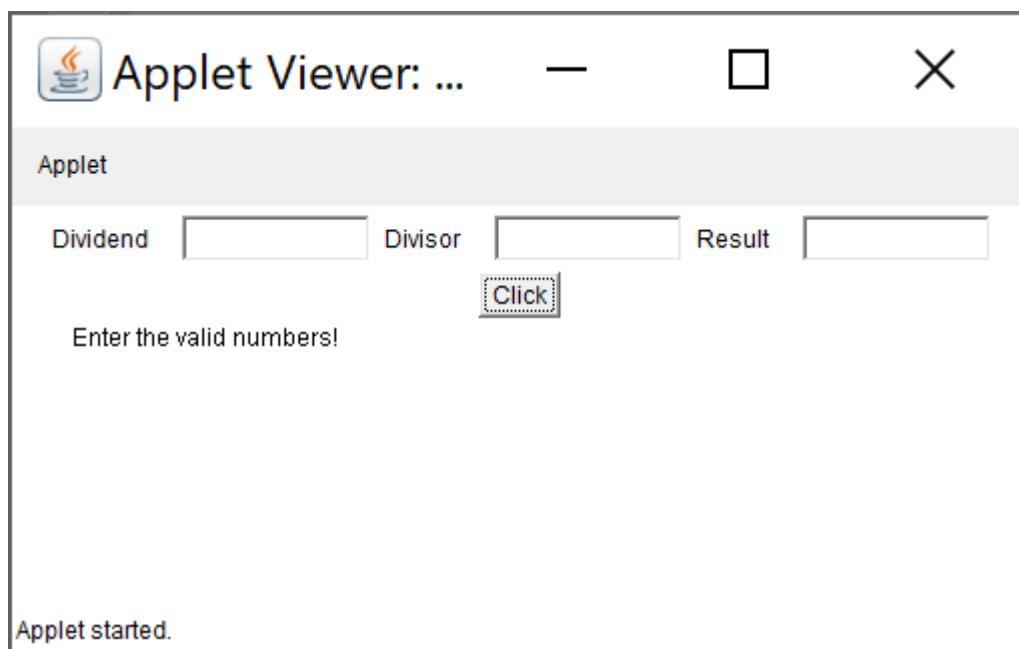
                    int num3 = num1 / num2;

                    res.setText(String.valueOf(num3));
                    msg = "Operation Succesfull!!!";
                    repaint();
                } catch (NumberFormatException ex) {
                    System.out.println(ex);
                    res.setText("");
                    msg = "NumberFormatException - Non-numeric";
                }
            }
        }
    }
}

```

```
        repaint();
    } catch (ArithmaticException 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:**

 Applet Viewer: ... — □ ×

Applet

Dividend  Divisor  Result

Can't be divided by Zero

Applet started.

 Applet Viewer: ... — □ ×

Applet

Dividend  Divisor  Result

Operation Succesfull!!!

Applet started.

**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.

**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 randomNumber) {
        number = randomNumber;
    }

    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:**

The screenshot shows the Eclipse IDE interface with the following details:

- File:** MultiThreadingTest.java
- Content:** Java code for multi-threading. It generates random integers, calculates their squares, and prints both the integer and its square.
- Output Console:** Displays the generated random integers and their squares.

```
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 randomNumber) {
            number = randomNumber;
        }
        public void run() {
            System.out.println("Square of " + number + " = " + (number * number));
        }
    }
}
```

Console Output:

```
<terminated> MultiThreadingTest [Java Application] C:\Program Files\Java\...
Random Integer generated : 35
Cube of 35 = 42875
Random Integer generated : 33
Cube of 33 = 35937
Random Integer generated : 79
Cube of 79 = 493039
Random Integer generated : 13
Cube of 13 = 2197
Random Integer generated : 89
Cube of 89 = 704969
Random Integer generated : 26
Square of 26 = 676
Random Integer generated : 94
Square of 94 = 8836
Random Integer generated : 24
Square of 24 = 576
Random Integer generated : 7
Cube of 7 = 343
Random Integer generated : 75
Cube of 75 = 421875
```

**Week6:**

Write a C++ to illustrate the concepts of console I/O operations.

**Source 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;
        } else {
            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) {

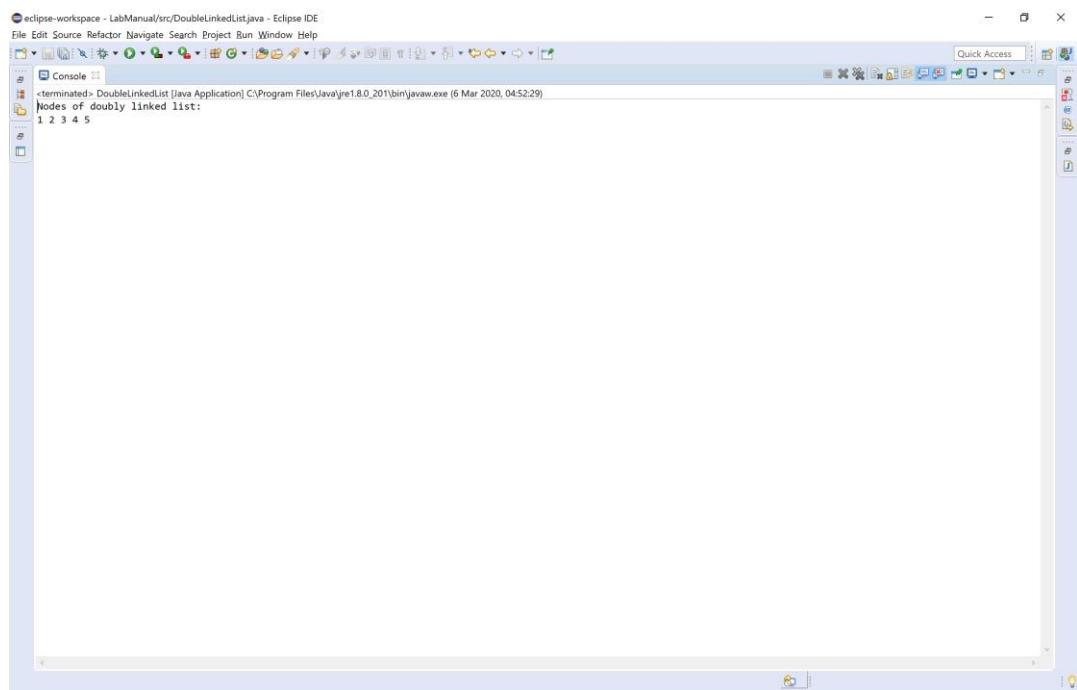
            System.out.print(current.data + " ");
            current = current.next;
        }
    }

    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:**

The screenshot shows the Eclipse IDE interface with a single open project named "eclipse-workspace - LabManual/src/DoubleLinkedList.java". The Java application has been run, and its output is displayed in the "Console" tab. The output text reads: "<terminated> DoubleLinkedList [Java Application] C:\Program Files\Java\jre1.8.0\_201\bin\javaw.exe (6 Mar 2020, 04:52:29)" followed by "Nodes of doubly linked list:" and then the numbers "1 2 3 4 5" on a new line.

```
<terminated> DoubleLinkedList [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (6 Mar 2020, 04:52:29)
Nodes of doubly linked list:
1 2 3 4 5
```

## 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 = "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(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) {
            yellowLight.setForeground(Color.YELLOW);
            msg.setForeground(Color.YELLOW);
            msg.setText("READY");
        }
        else{
            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 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:**

The screenshot shows the Eclipse IDE interface with the following details:

- Left Panel (Code Editor):** Displays the Java code for `AbstractClassExample.java`. The code defines an abstract class `Shape` with an abstract method `printArea()`. It also contains three concrete subclasses: `Rectangle`, `Triangle`, and `Cricle`, each overriding the `printArea()` method. The `AbstractClassExample` class has a `main` method that creates a `Rectangle` object and calls its `printArea` method.
- Right Panel (Console):** Shows the execution output of the program. It starts with a header: "\*\*\* Finding the Area of Rectangle \*\*\*". It prompts the user to "Enter length and breadth: 2 3" and then displays the result "The area of Rectangle is: 6". This is followed by another header: "\*\*\* Finding the Area of Triangle \*\*\*". It prompts the user to "Enter Base And Height: 4 5" and then displays the result "The area of Triangle is: 10". Finally, it shows a third header: "\*\*\* Finding the Area of Cricle \*\*\*". It prompts the user to "Enter Radius: 5" and then displays the result "The area of Cricle is: 78.5".

## 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:

```

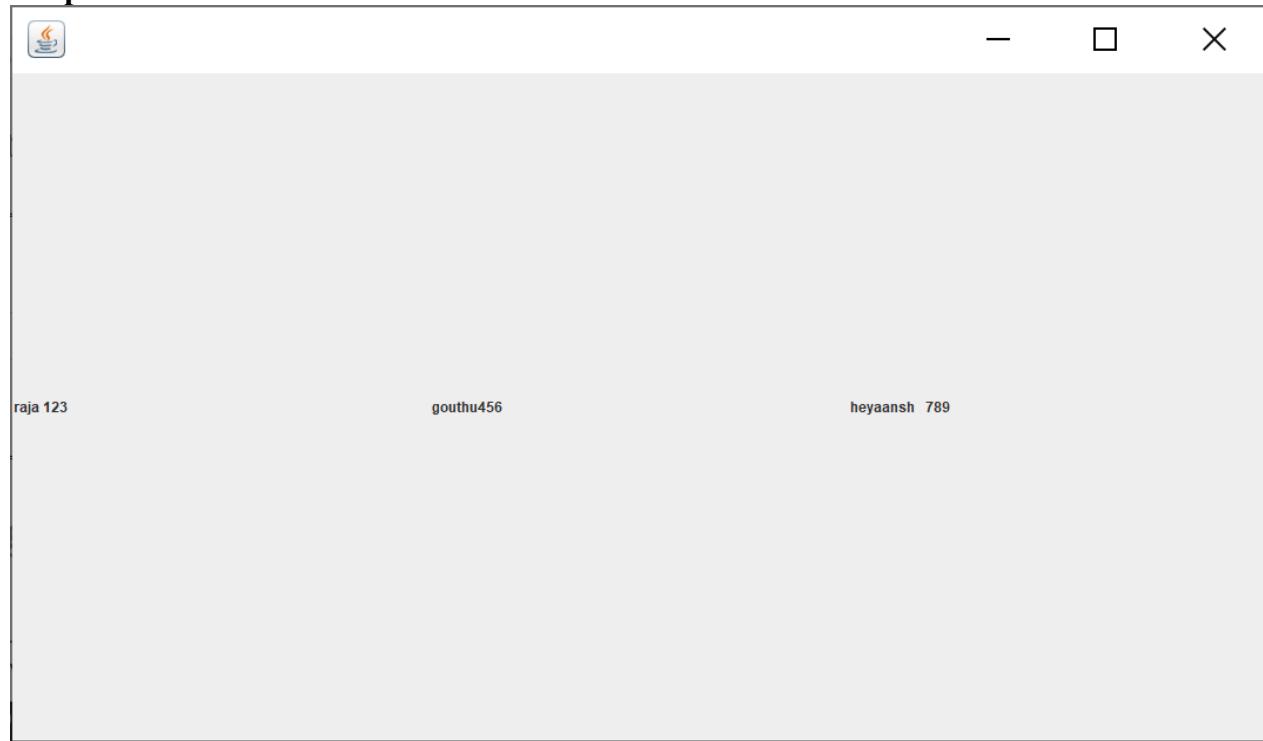
import java.io.*;
import java.util.*;
import java.awt.*;
import javax.swing.*;

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\\\\HashTable.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:**



A screenshot of a terminal window with a light gray background. In the top-left corner is a small icon of a computer monitor with a flame on it. In the top-right corner are three standard window control buttons: a minus sign for minimize, a square for maximize/minimize, and an X for close. At the bottom of the window, there are three entries, each consisting of a name followed by a space and a string of numbers: "raja 123", "gouthu456", and "heyaansh 789".

## 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();
    }

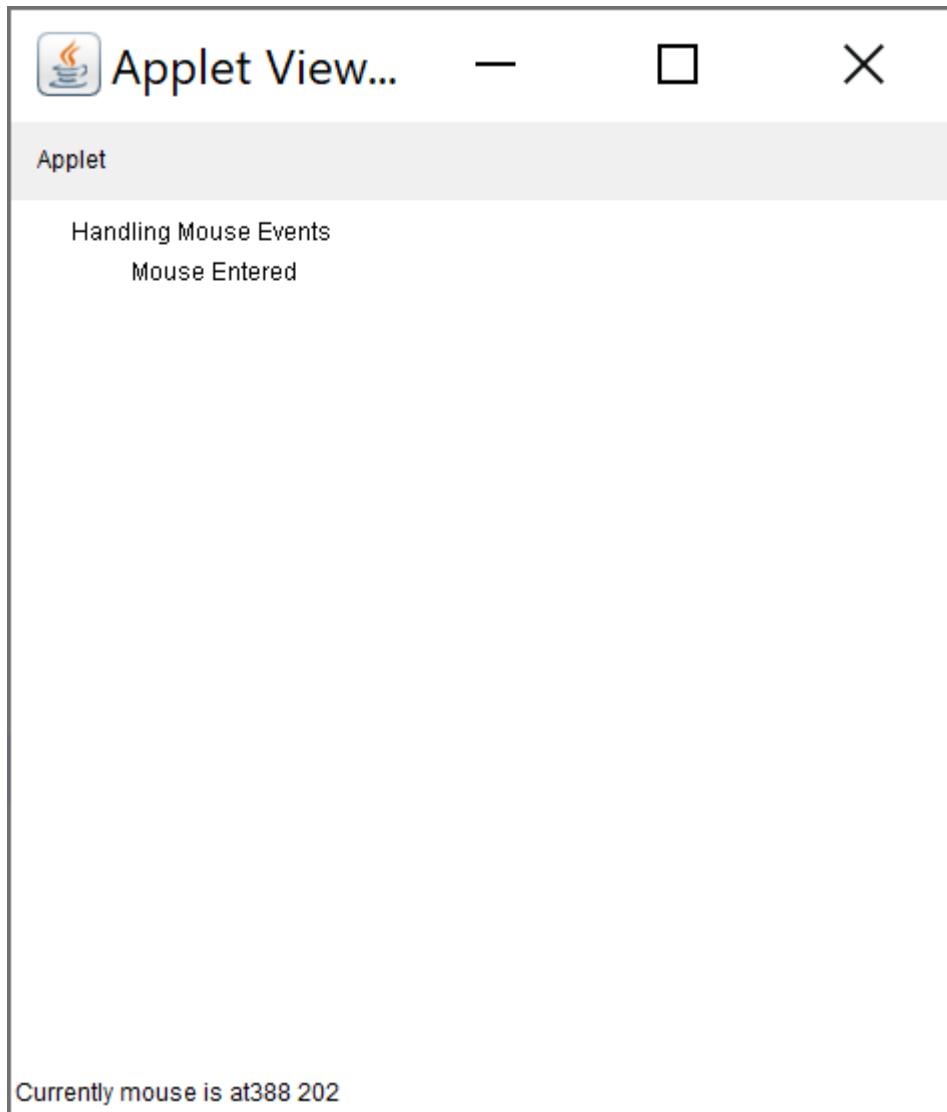
    public void mouseDragged(MouseEvent me) {
        mx = me.getX();
        my = me.getY();
        showStatus("Currently mouse dragged" + mx + " " + my);
    }
}

```

```
        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("HashTable.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 {
            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("D:\\java\\" + 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;
}
}
```

**Output:**

```
C:\WINDOWS\system32\cmd.exe
Data not found in HashTable
D:\java>javac HashTab.java
D:\java>java HashTab
File data into HashTable:
Data given:vbit
Data found in HashTable:vbit
Input given:123
Data found in HashTable:123
Input given:123
Data found in HashTable:123
D:\java>
```

## Week – 12

Write a Java program that correctly implements the producer – consumer problem using the concept of interthread 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++);
        }
    }
}

```

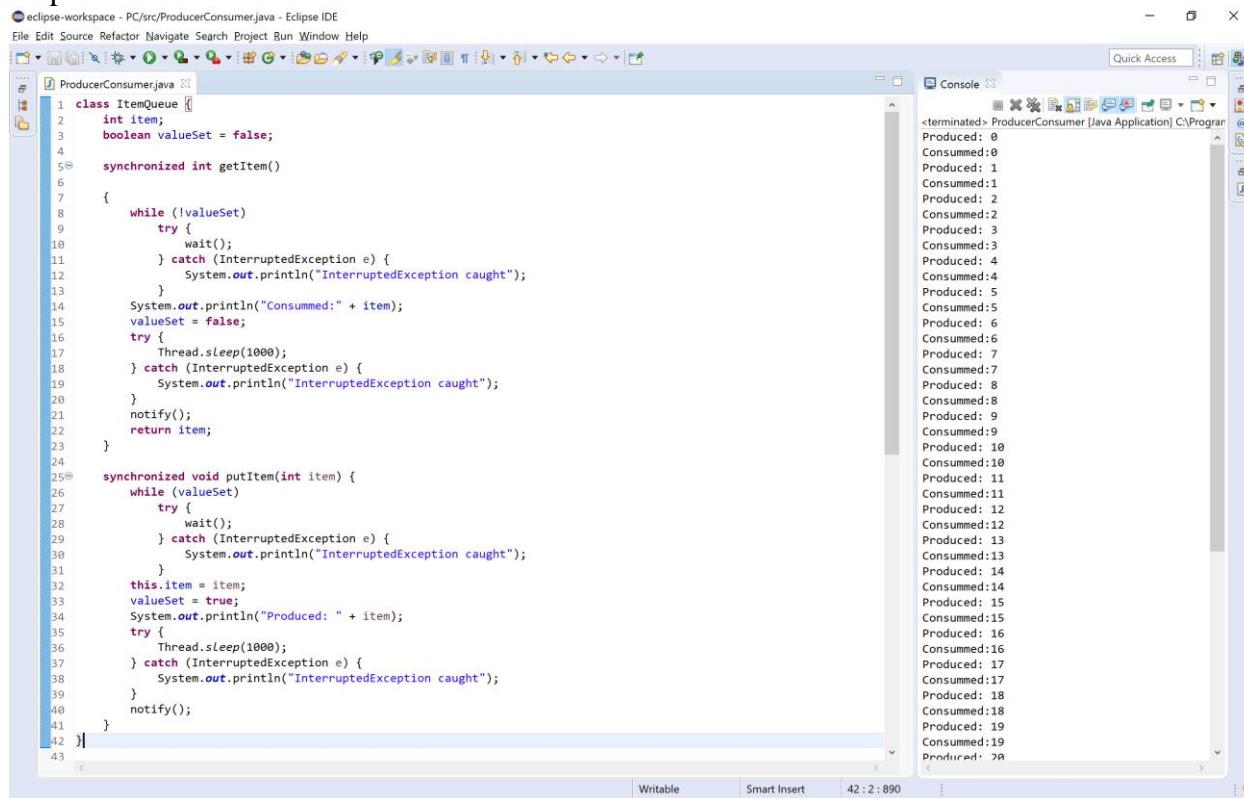
```

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:



The screenshot shows the Eclipse IDE interface with the ProducerConsumer.java file open in the editor. The code defines an ItemQueue class with synchronized methods for getting and putting items, and a ProducerConsumer class with a main method creating instances of ItemQueue, Producer, and Consumer.

The Console view displays the program's output, showing a sequence of produced and consumed items. The output is as follows:

```

<terminated> ProducerConsumer [Java Application] C:\Program
Produced: 0
Consumed:0
Produced: 1
Consumed:1
Produced: 2
Consumed:2
Produced: 3
Consumed:3
Produced: 4
Consumed:4
Produced: 5
Consumed:5
Produced: 6
Consumed:6
Produced: 7
Consumed:7
Produced: 8
Consumed:8
Produced: 9
Consumed:9
Produced: 10
Consumed:10
Produced: 11
Consumed:11
Produced: 12
Consumed:12
Produced: 13
Consumed:13
Produced: 14
Consumed:14
Produced: 15
Consumed:15
Produced: 16
Consumed:16
Produced: 17
Consumed:17
Produced: 18
Consumed:18
Produced: 19
Consumed:19
Produced: 20

```

## 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:**

The screenshot shows the Eclipse IDE interface with a Java project named "ListingFiles". The code in ListingFiles.java is as follows:

```
1 import java.util.Scanner;
2 import java.io.*;
3
4 public class ListingFiles {
5
6     public static void main(String[] args) {
7
8         String path = null;
9         Scanner read = new Scanner(System.in);
10        System.out.print("Enter the root directory name: ");
11        path = read.nextLine() + "\\\".
```

The console output shows the program running and prompting for input:

```
ListingFiles [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (3 Feb 2020, 12:18:07)
> Windows
> XML
Do you want to open any sub-directory (Y/N): y
Enter the sub-directory name: Raja
Current Location: C:\\\\Raja
> Balu
> BSC 2018
> BSC Hosted
> CMREC
> Education
```

**Week 14**

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

**Source Code:**

```

public class QuickSortOnStrings {

    String names[];
    int length;

    public static void main(String[] args) {
        QuickSortOnStrings obj = new QuickSortOnStrings();
        String stringsList[] = {"raja", "gouthu", "rani", "gouthami", "honey",
        "heyaansh", "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(int lowerIndex, int higherIndex) {
        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:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the file `QuickSortOnStrings.java`.
- Code Editor:** Displays the Java code for `QuickSortOnStrings`. The code includes a main method that initializes an array of strings with values "raja", "gouthu", "rani", "gouthami", "honey", "heyaansh", and "hello". It then calls the `sort` method on an object of `QuickSortOnStrings`. Inside the `sort` method, it prints each string followed by a space.
- Console:** Shows the terminal output of the application. The output reads:  
<terminated> QuickSortOnStrings [Java Application] C:\Program Files\Java\jre1.8.0\_201\bin\javaw.exe (21 Jan 2020, 15:03:13)  
gouthami gouthu hello heyaansh honey raja rani |

## Week 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.

**Source Code:**

```
import java.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(int i = 0; i < size; i++)
            list[i] = read.nextInt();

        // Bubble sort logic
        int temp=0;
        for(int i=0;i<size-1;i++) {
            for(int j=0;j<size-i-1;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:**

The screenshot shows the Eclipse IDE interface with a Java file named BubbleSort.java open in the editor. The code implements a bubble sort algorithm to sort an array of integers entered by the user. The console window shows the execution of the program, including the input of five integers (5, 4, 3, 2, 1), the sorted output (5, 4, 3, 2, 1), and the total number of interchanges (5).

```
1 import java.util.Scanner;
2 public class BubbleSort {
3
4     public static void main(String[] args) {
5         Scanner read = new Scanner(System.in);
6         int size, count = 0;
7         //Reading size of the list
8         System.out.print("Enter the list size: ");
9         size = read.nextInt();
10
11        //Creating list with elements
12        int list[] = new int[size];
13        System.out.println("Enter any " + size + " integer numbers: ");
14        for(int i = 0; i < size; i++)
15            list[i] = read.nextInt();
16
17        // Bubble sort logic
18        int temp=0;
19        for(int i=0;i<size-1;i++) {
20            for(int j=0;j<size-i-1;j++) {
21                if(list[j]<list[j+1]) {
22                    temp=list[j];
23                    list[j]=list[j+1];
24                    list[j+1]=temp;
25                    count++;
26                }
27            }
28        }
29
30        // Displaying sorted list
31        System.out.println("List of sorted elements: ");
32        for(int x:list) {
33            System.out.print(x + " ");
34        }
35        System.out.println("\nTotal number of Interchanges is " + count);
36    }
37 }
38 }
```