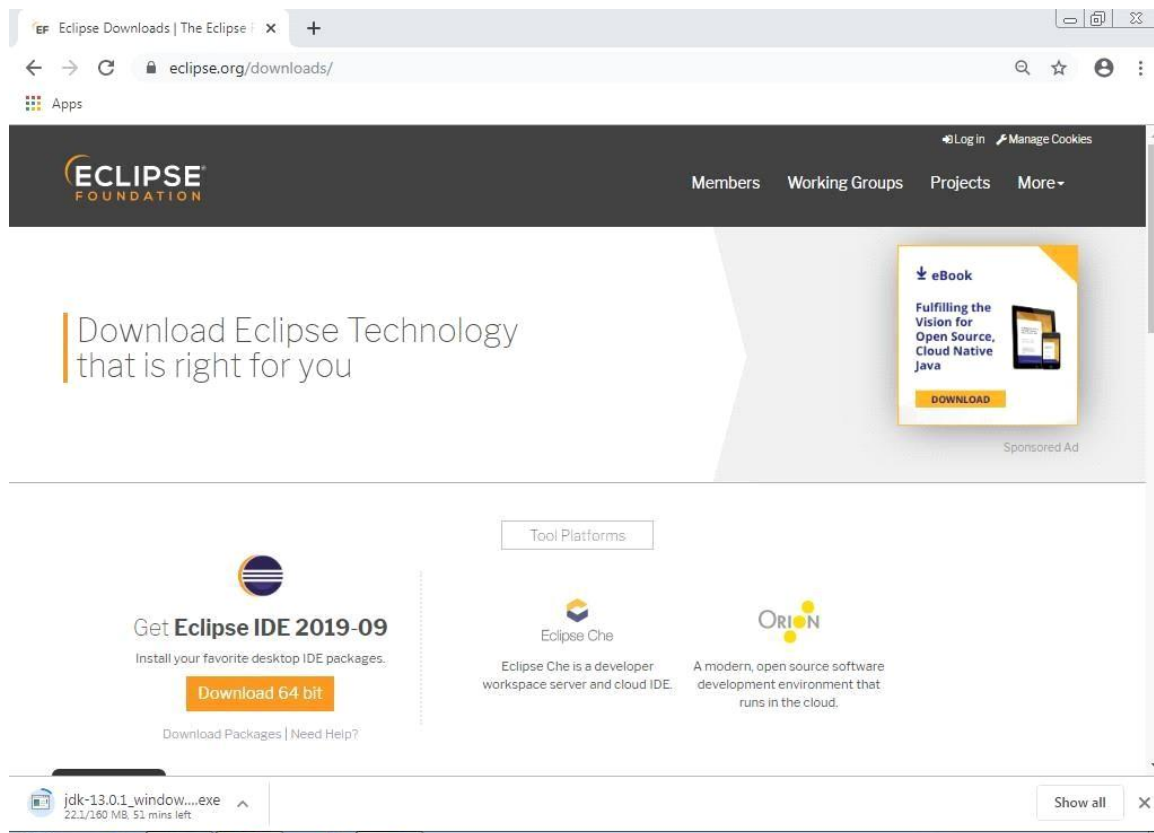


- Use Eclipse or Net bean platform and acquaint yourself with the various menus. Create a test project, add a test class, and run it. See how you can use auto suggestions, auto fill. Try code formatter and code refactoring like renaming variables, methods, and classes. Try debug step by step with a small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

Solution:

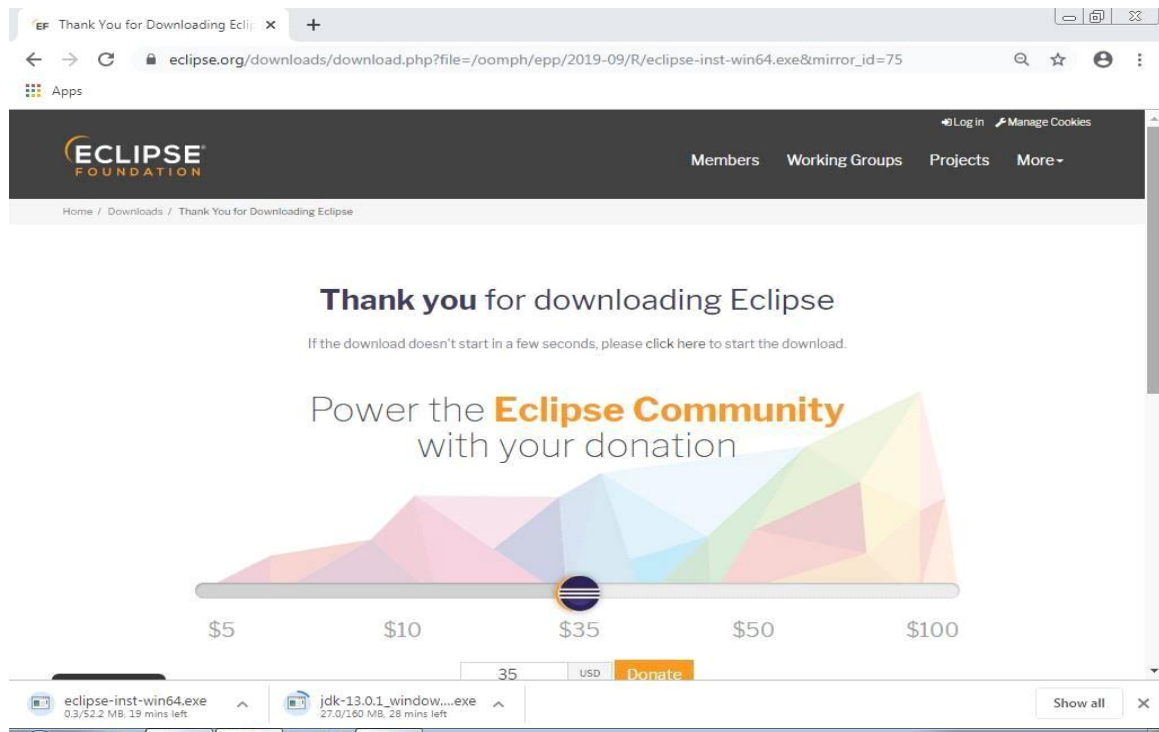
- **Step 1** - Install JDK in the computer.
- **Step 2** - Set the path in the Environment Variables from Advanced Setting of computer
- **Step 3** - Download Eclipse from Eclipse website
- **Step 4** - Install the Eclipse (follow the screen to install eclipse)



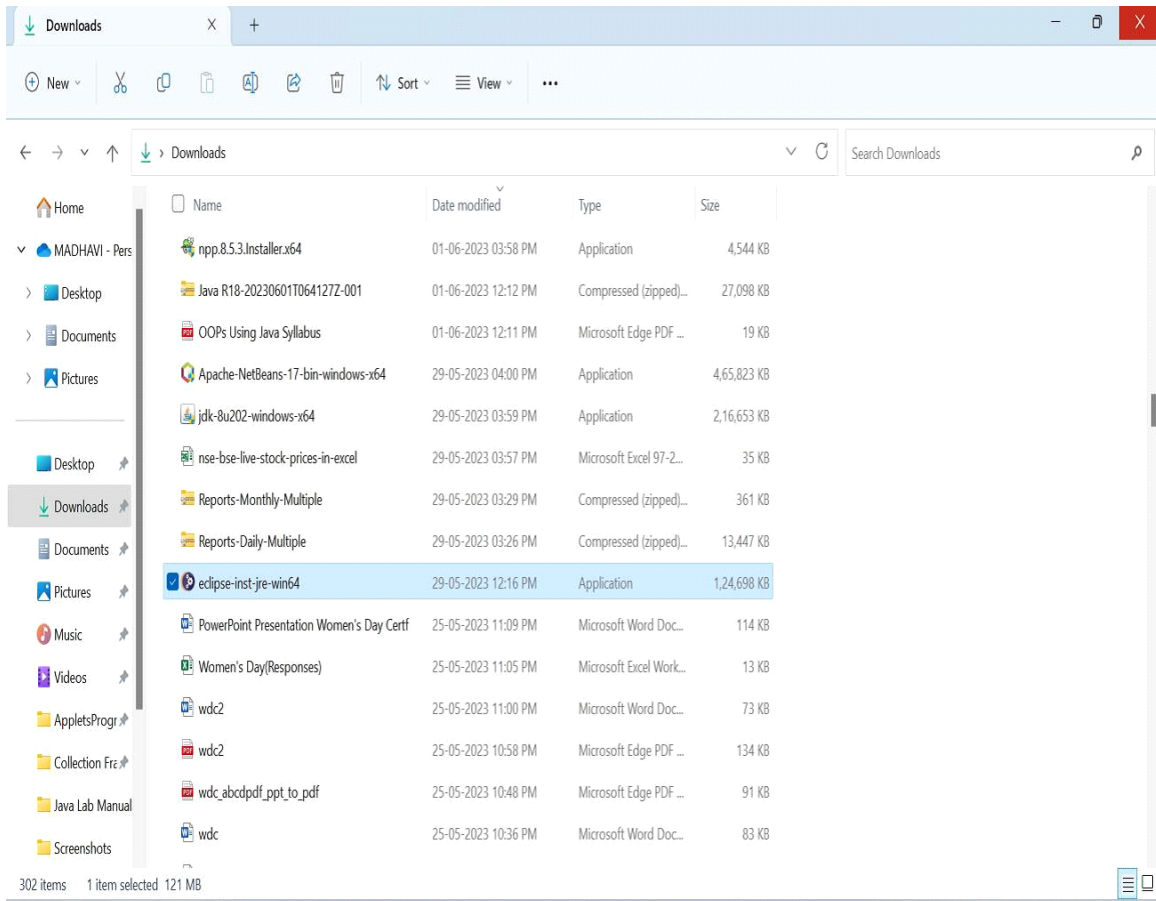
Select the suitable version based on your OS.

The screenshot shows a web browser window with the URL `eclipse.org/downloads/download.php?file=/oomph/epp/2019-09/R/eclipse-inst-win64.exe`. The page header includes the Eclipse Foundation logo and navigation links: Members, Working Groups, Projects, and More. Below the header, a disclaimer states: "All downloads are provided under the terms and conditions of the Eclipse Foundation Software User Agreement unless otherwise specified." The main content area features a large orange "Download" button. Below it, the download source is listed as "Download from: Taiwan - Computer Center, Shu-Te University (http)" and the file is identified as "File: eclipse-inst-win64.exe" with a SHA-512 hash. A link ">> Select Another Mirror" is provided. A purple banner reads "OR Get It Faster from our Members". Below this banner are three member logos: IBM (Blazingly fast downloads hosted by IBM Cloud), BLU AGE (Free and fast direct Eclipse downloads. Get more BLU AG...), and Obeo (Fast downloads hosted by Eclipse experts). Each member has a "Get it" button. On the right side, there is a Papyrus logo with the text "An Open-Source Model-Based Engineering Platform". Below this, a section titled "Other options for this file" lists: "All mirrors (xml)" and "Direct link to file (download starts immediately from best mirror)". A "Related Links" section lists: "Donate", "Becoming a mirror site", and "Updating and installing Eclipse components". At the bottom, a taskbar shows a file named "jdk-13.0.1_window....exe" with a size of 24.9/160 MB and 24 mins left. A "Show all" button is visible in the bottom right corner.

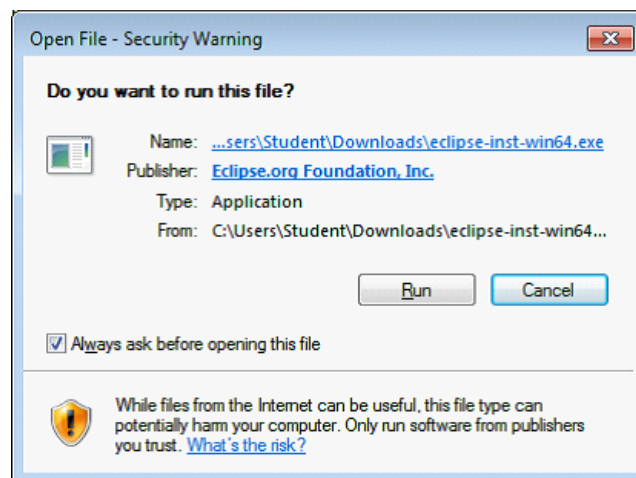
Then download get starts.



Double click on the Eclipse Application.



Click on Run in the Security Warning box.



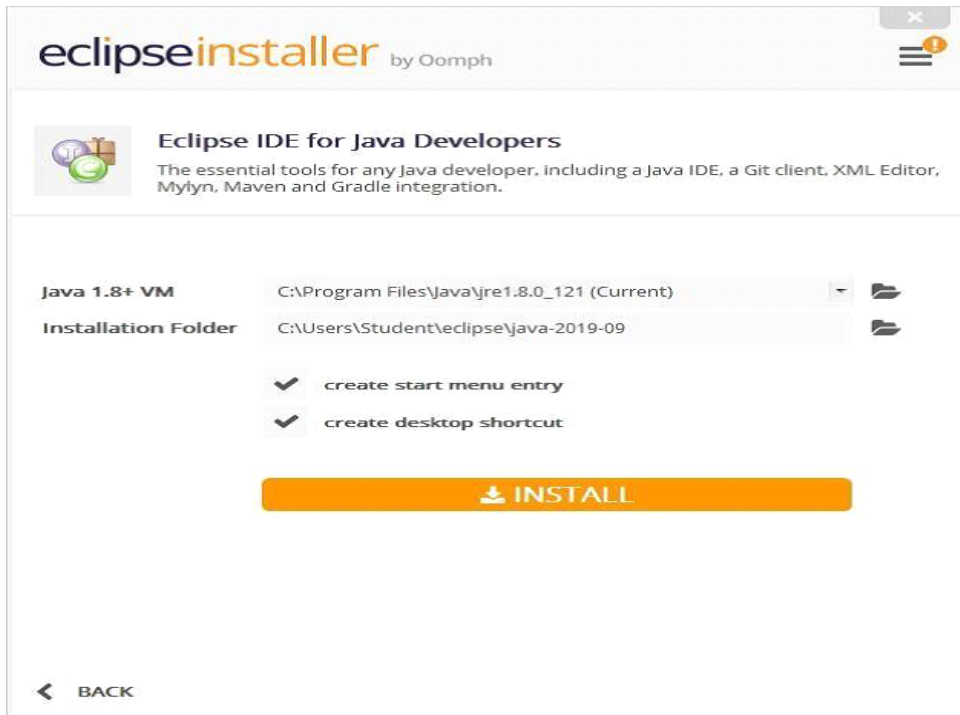
Then, the installation process begins.



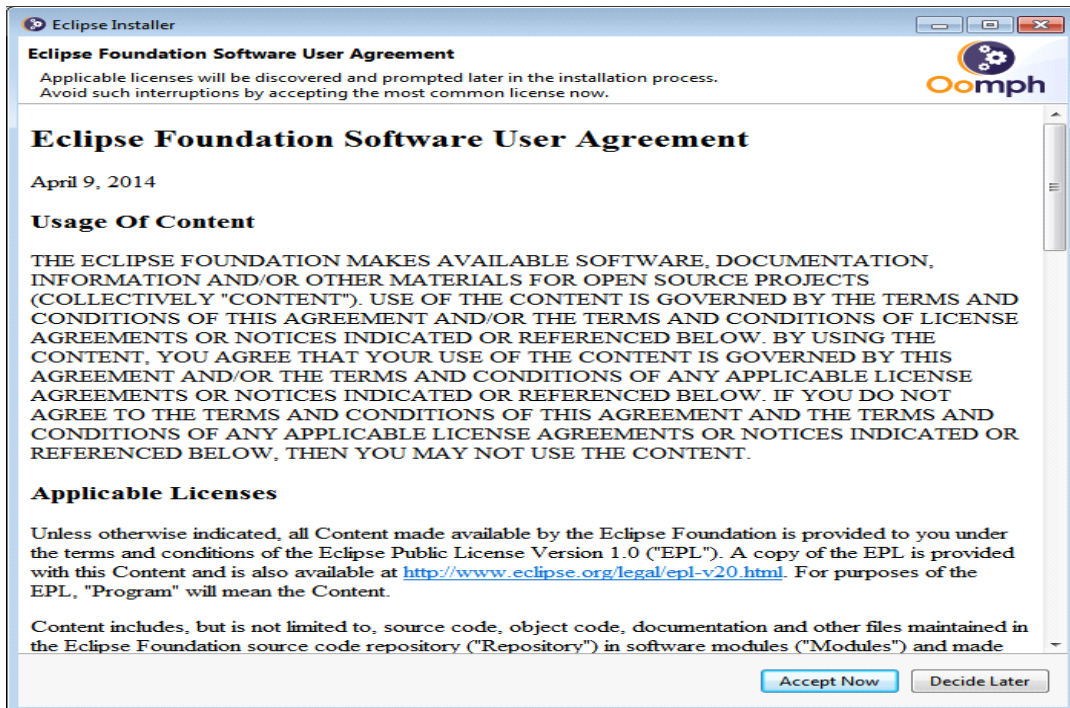
Click on Eclipse IDE for Java Developers.



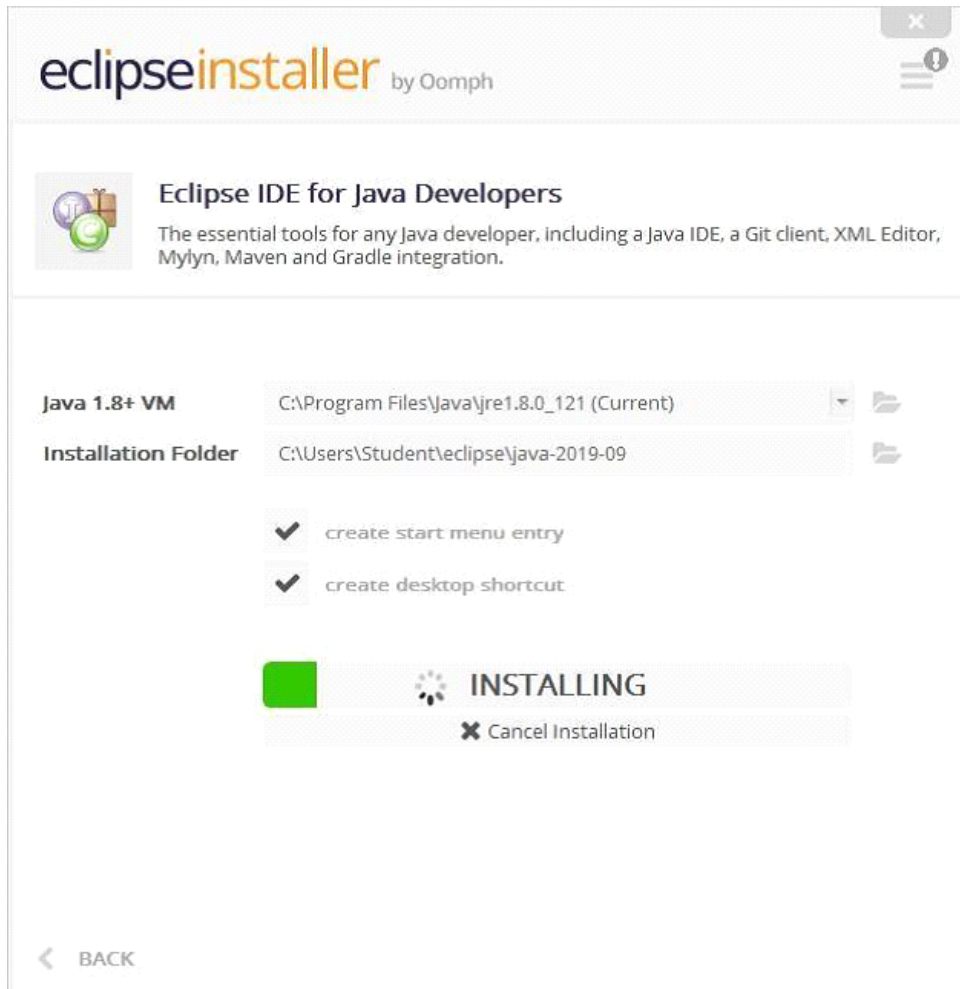
Click on Install button.



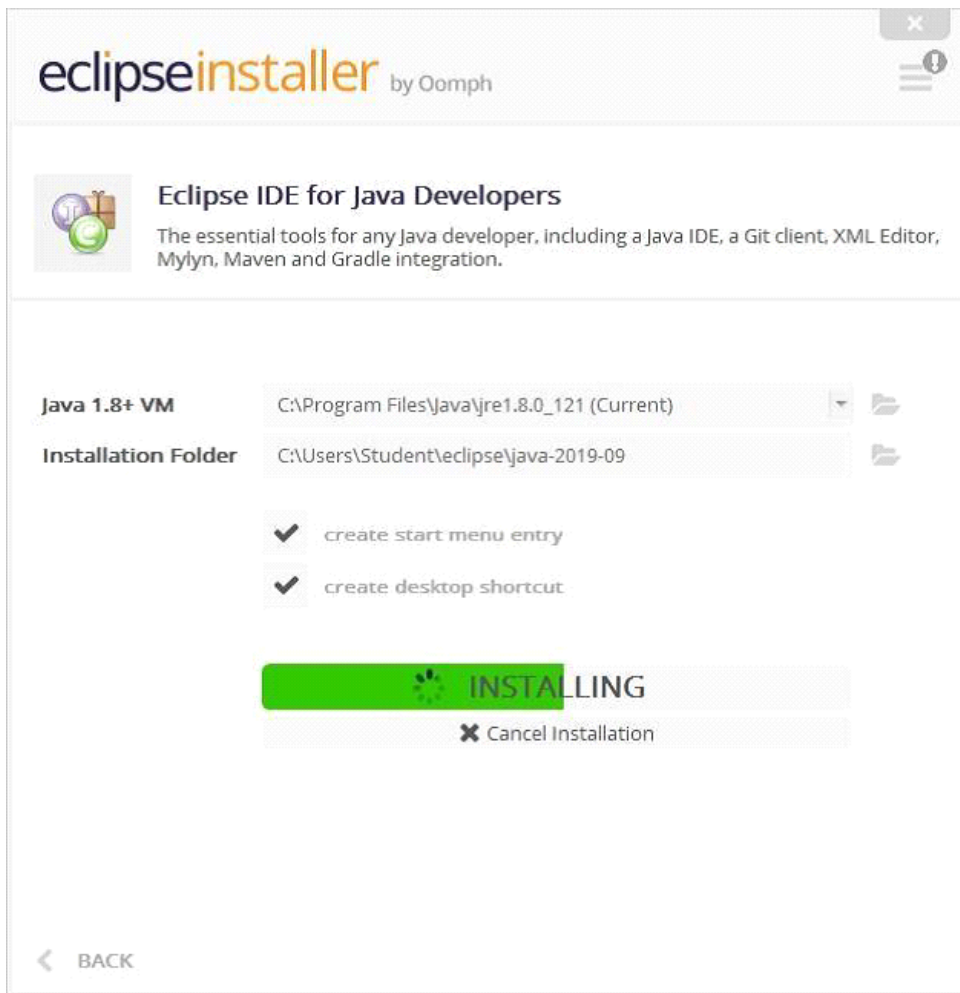
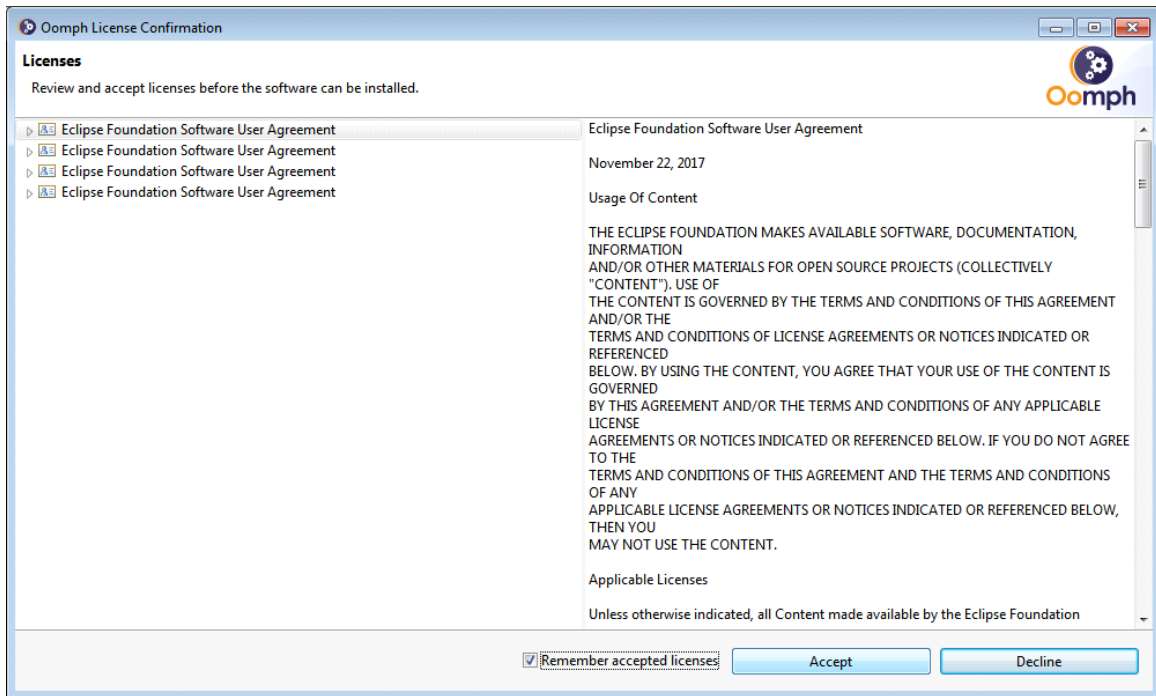
Click on Accept Now.



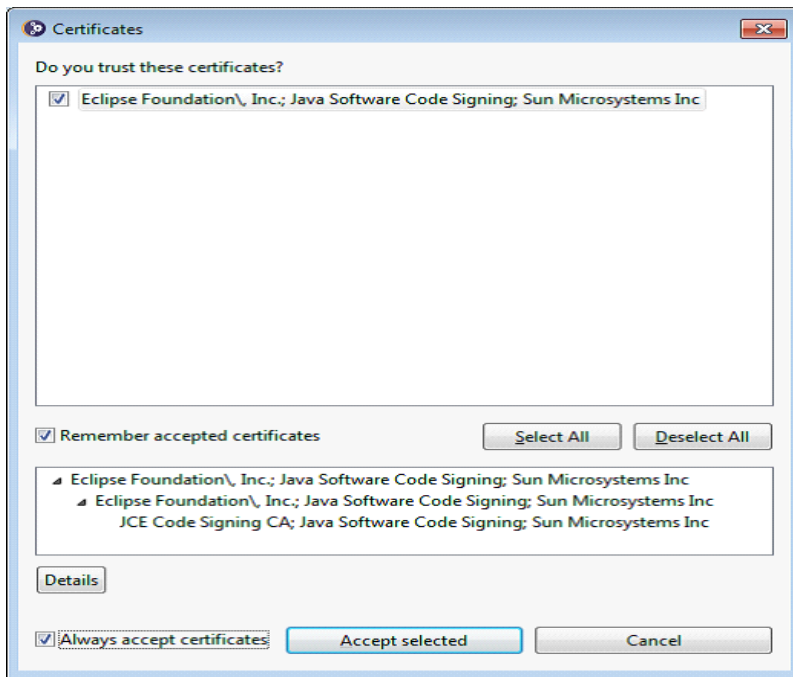
Then the Eclipse installation begins.



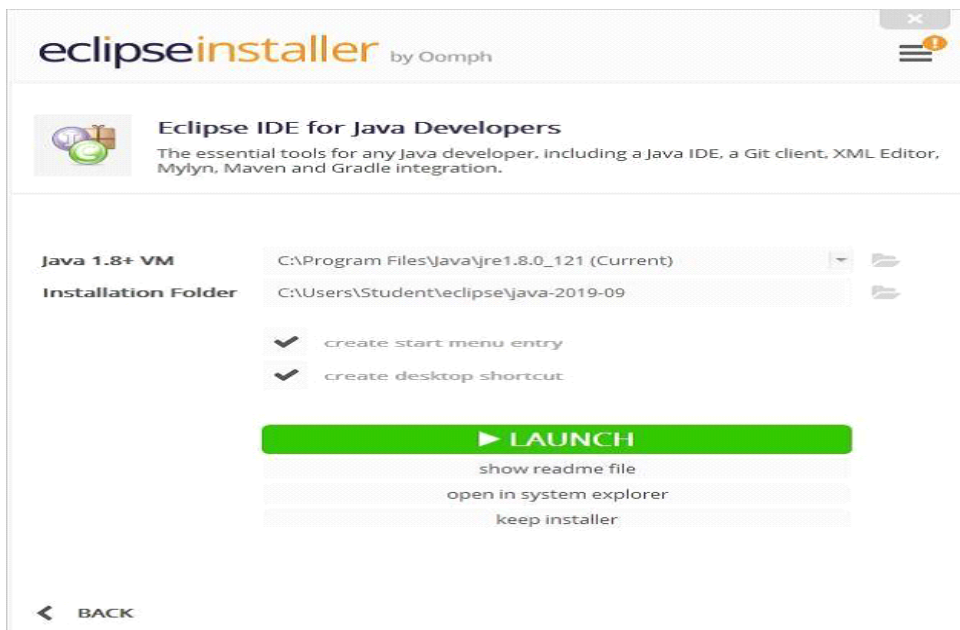
Click on Accept



Click on Select All and Accept Selected.

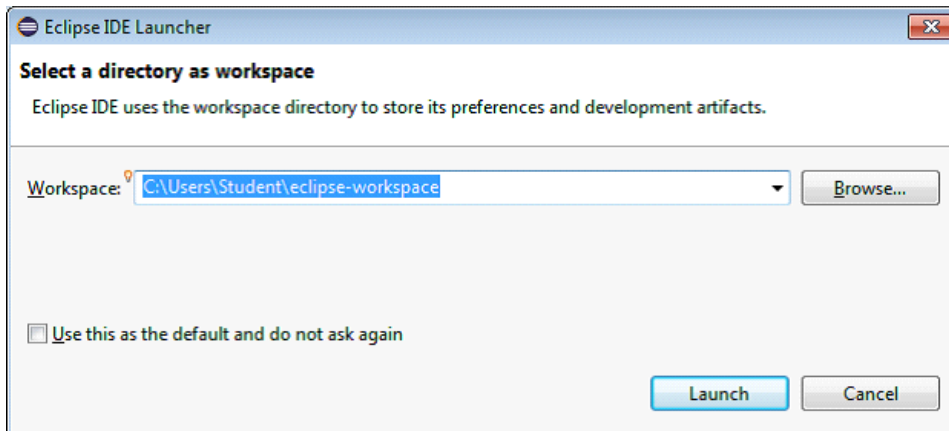


After completing, click on Launch to start the Eclipse IDE.

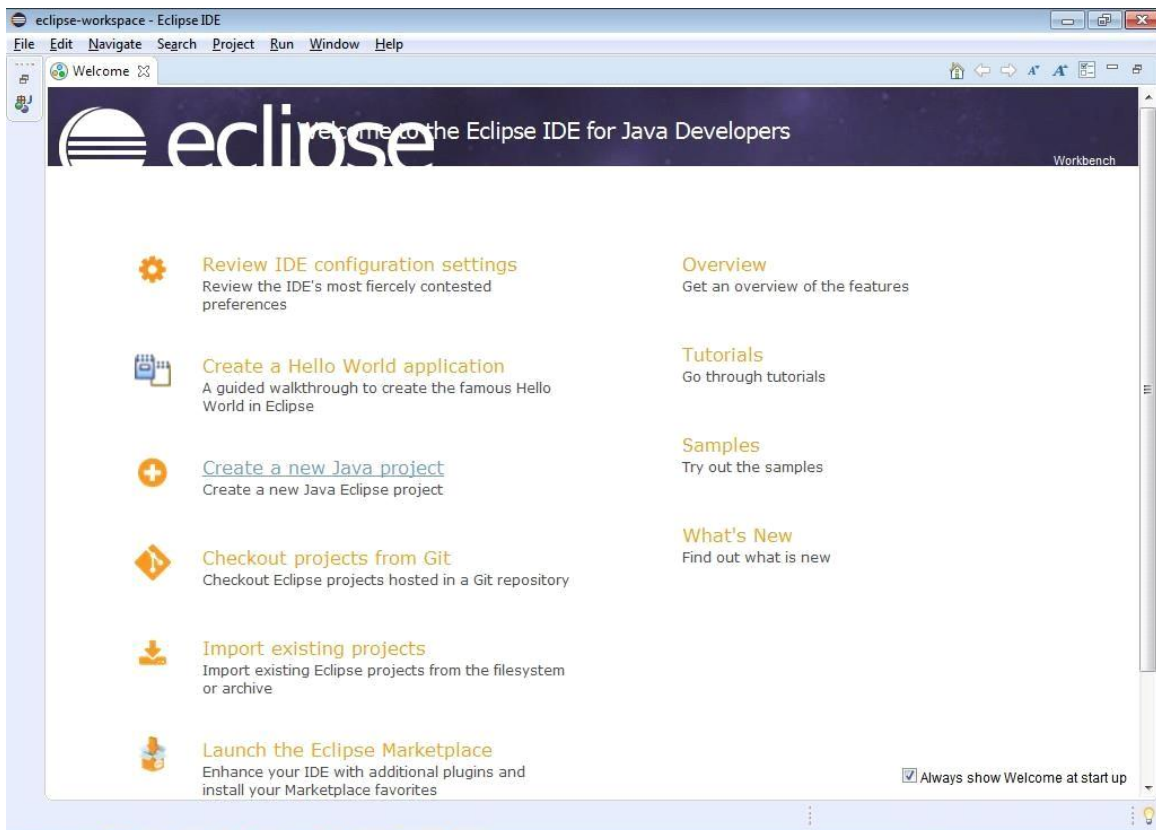


CREATING PROJECT AND CLASSES IN ECLIPSE IDE

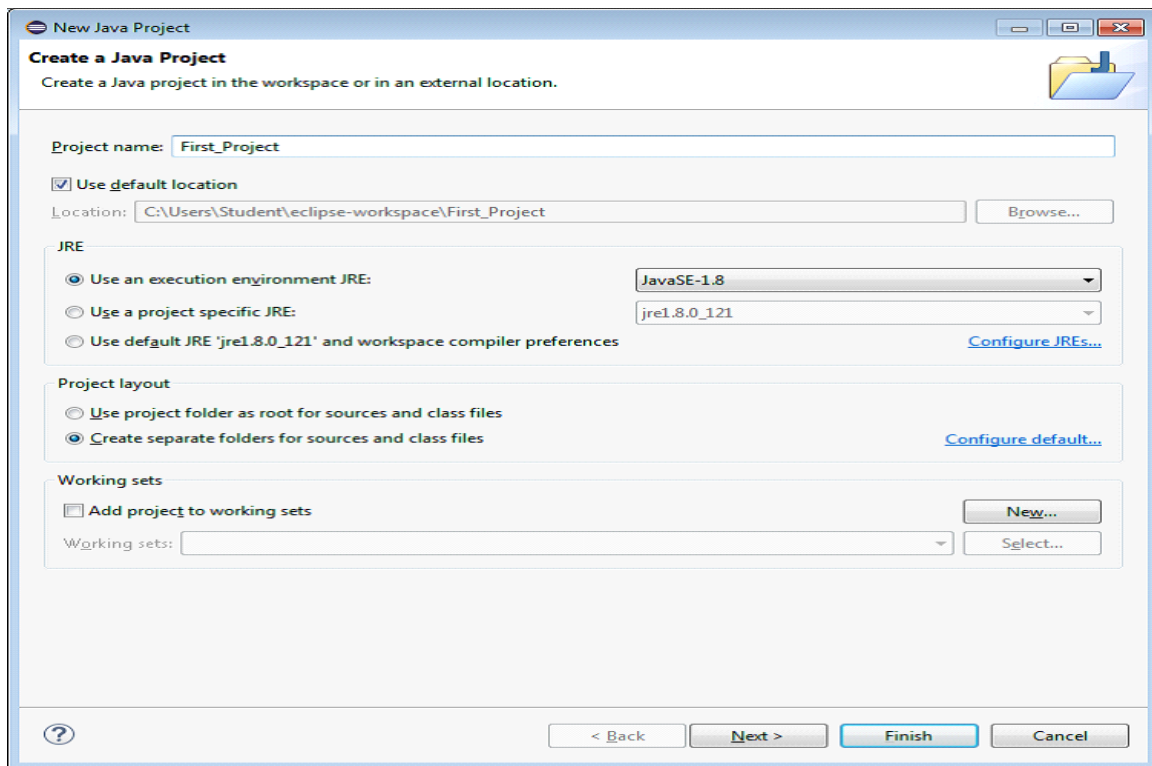
Browse the Workspace for storing the java project and click on Launch.



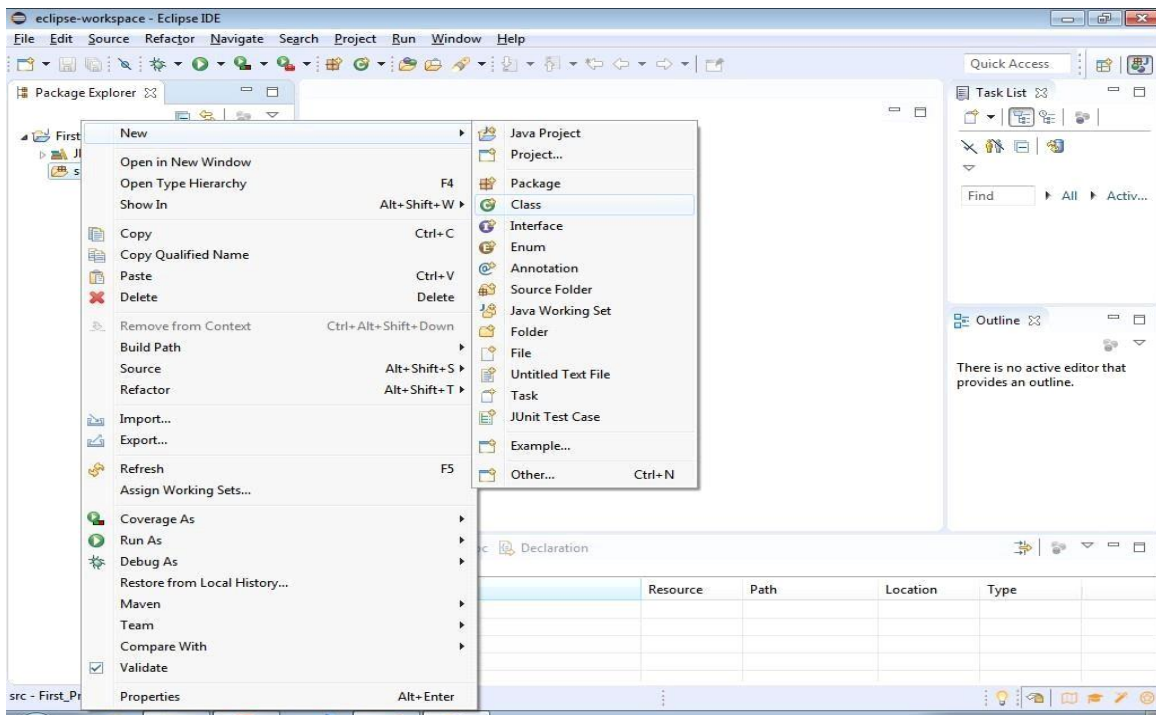
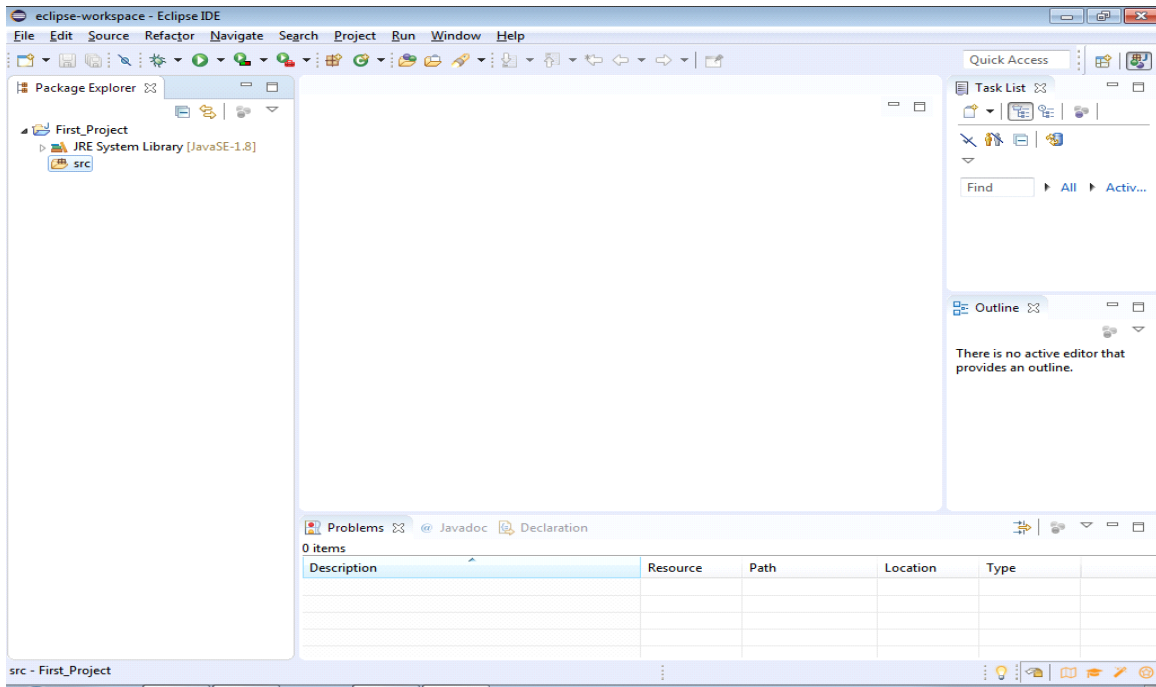
Select "Create a new Java project"



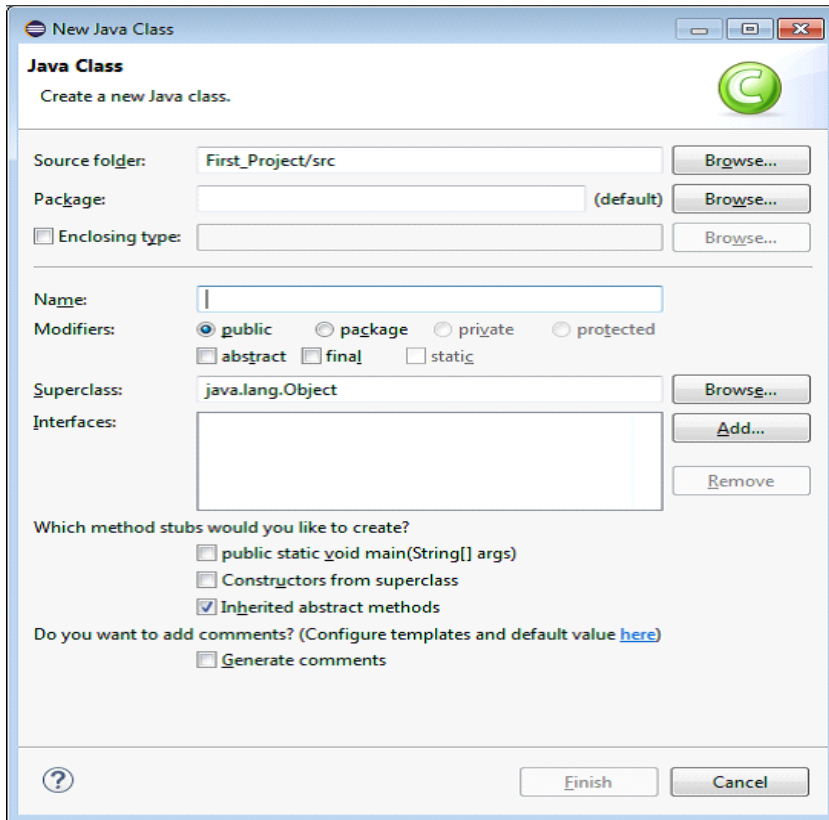
Type the project name and click on Finish.



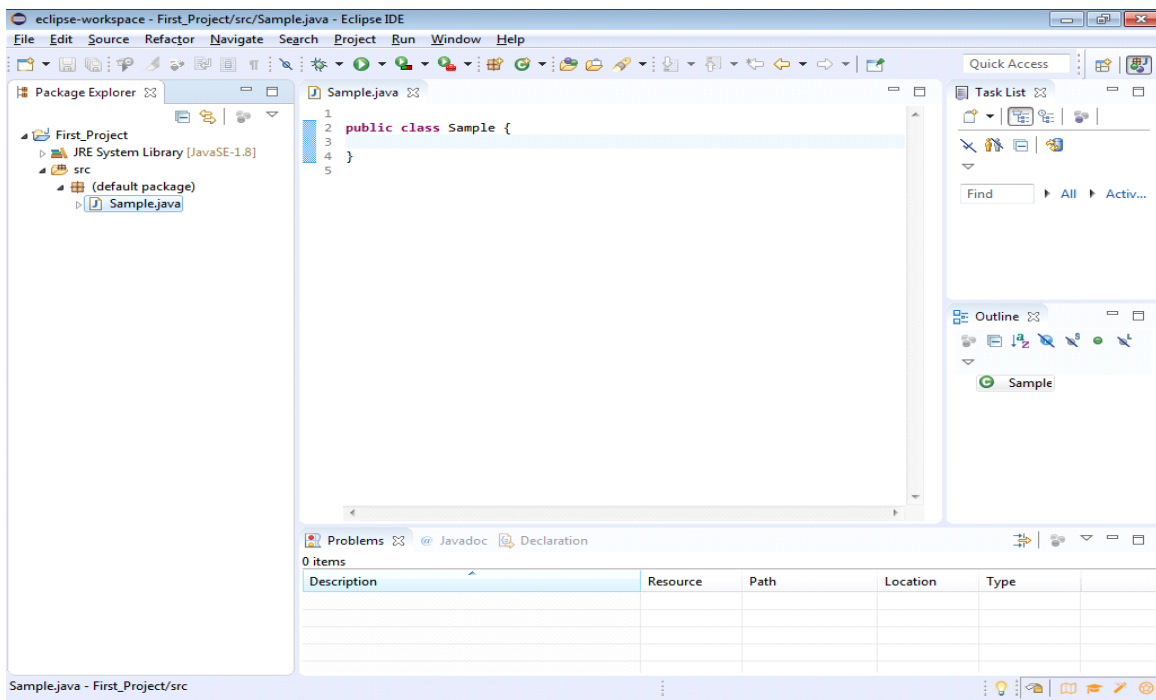
Now, create the class in src directory from Package Explorer window.



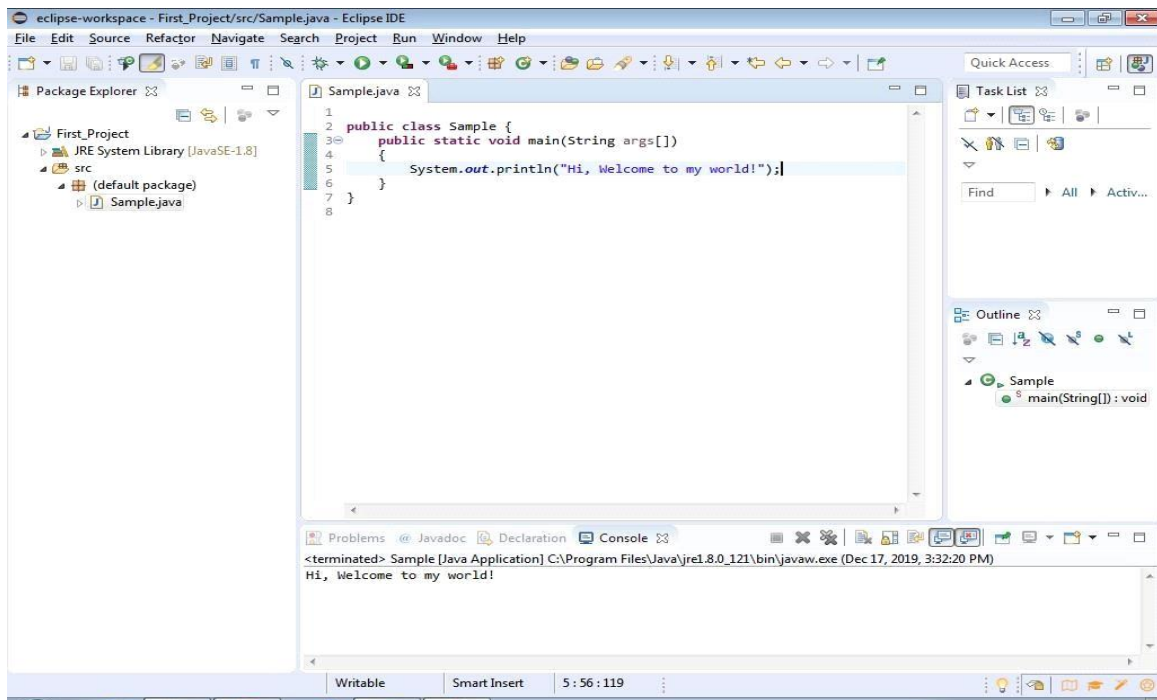
Type the class name and click on Finish.



Type the java code.



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



2. Write a Java program to demonstrate the OOP principles. [i.e., Encapsulation, Inheritance, Polymorphism and Abstraction]

Encapsulation example

```
public class BankAccount {
```

```
    // Private variables to hold account details
```

```
    private String accountNumber;
```

```
    private double balance;
```

```
    // Constructor to initialize account details
```

```
    public BankAccount(String accountNumber, double balance) {
```

```
        this.accountNumber = accountNumber;
```

```
        this.balance = balance;
```

```
    }
```

```
    // Method to deposit money into the account
```

```

public void deposit(double amount) {
    if (amount > 0) {
        balance += amount;
        System.out.println(amount + " deposited successfully.");
    } else {
        System.out.println("Invalid amount.");
    }
}

// Method to withdraw money from the account
public void withdraw(double amount) {
    if (amount > 0 && balance >= amount) {
        balance -= amount;
        System.out.println(amount + " withdrawn successfully.");
    } else {
        System.out.println("Insufficient balance or invalid amount.");
    }
}

// Method to get the current balance
public double getBalance() {
    return balance;
}
}

```

EncapsulationDemo.java

```

public class EncapsulationDemo {
    public static void main(String[] args) {
        // Creating a BankAccount object
        BankAccount account = new BankAccount("123456789", 1000.0);

        // Accessing the account details using public methods
        System.out.println("Initial balance: " + account.getBalance());

        // Depositing and withdrawing money
        account.deposit(500.0);
    }
}

```

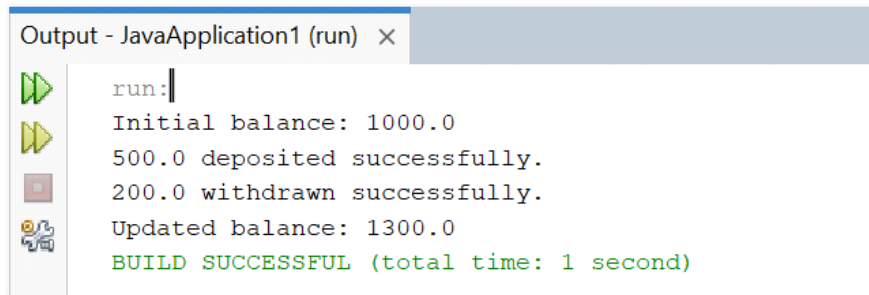


```

        account.withdraw(200.0);

        // Getting the updated balance
        System.out.println("Updated balance: " + account.getBalance());
    }
}

```



```

Output - JavaApplication1 (run) ×
run:
Initial balance: 1000.0
500.0 deposited successfully.
200.0 withdrawn successfully.
Updated balance: 1300.0
BUILD SUCCESSFUL (total time: 1 second)

```

Abstraction example

```

// Abstraction: Displaying information without exposing implementation details
abstract class Shape {

    String color;

```

```

// these are abstract methods

    abstract double area();

    public abstract String toString();

```

```

// abstract class can have the constructor

    public Shape(String color)

    {

        System.out.println("Shape constructor called");

        this.color = color;

    }

```

```
// this is a concrete method

public String getColor() { return color; }
}

class Circle extends Shape {

    double radius;

    public Circle(String color, double radius)
    {

        // calling Shape constructor

        super(color);

        System.out.println("Circle constructor called");

        this.radius = radius;

    }

    @Override double area()

    {

        return Math.PI * Math.pow(radius, 2);

    }

    @Override public String toString()

    {

        return "Circle color is " + super.getColor()

            + "and area is : " + area();

    }

}
```

```

    }
}

class Rectangle extends Shape {

    double length;

    double width;

    public Rectangle(String color, double length,
                     double width)
    {
        // calling Shape constructor
        super(color);

        System.out.println("Rectangle constructor called");

        this.length = length;

        this.width = width;
    }

    @Override double area() { return length * width; }

    @Override public String toString()
    {
        return "Rectangle color is " + super.getColor()
            + "and area is : " + area();
    }
}

```

```

public class Test {

    public static void main(String[] args)

    {

        Shape s1 = new Circle("Red", 2.2);

        Shape s2 = new Rectangle("Yellow", 2, 4);

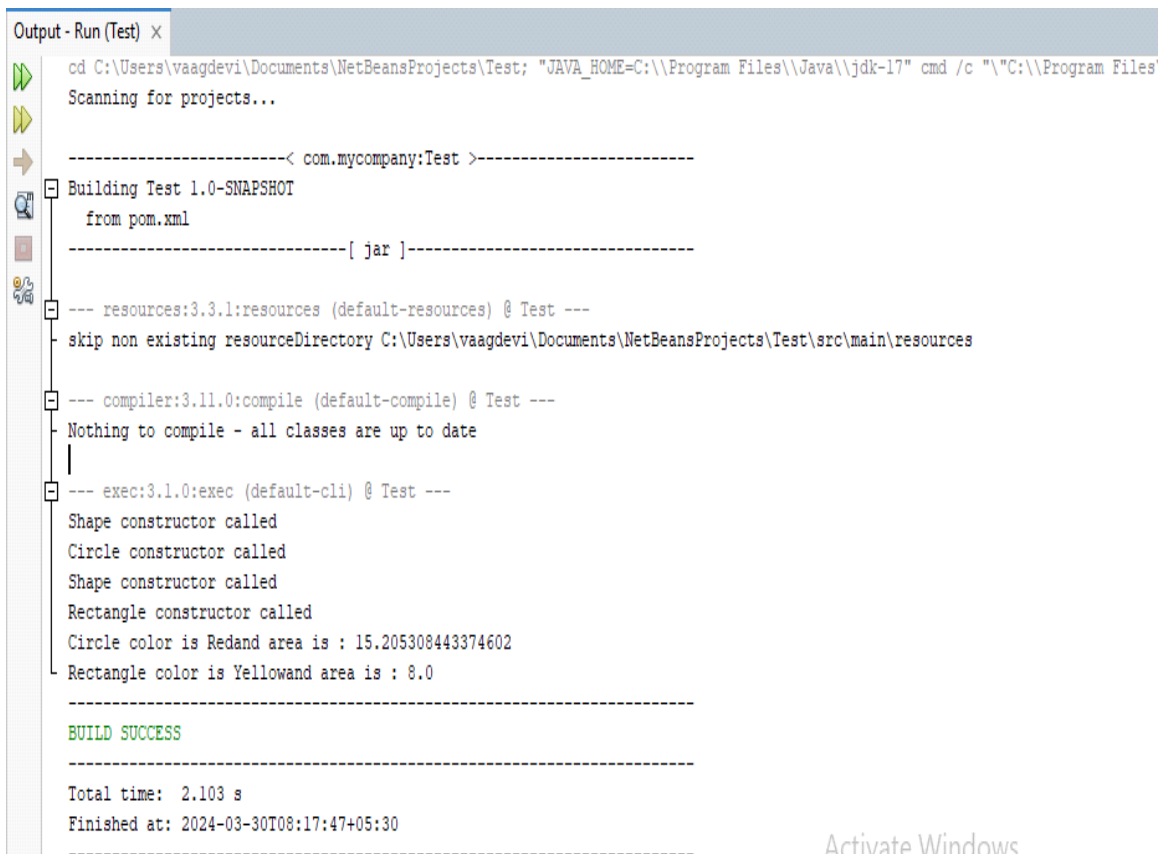

        System.out.println(s1.toString());

        System.out.println(s2.toString());

    }

}

```



```

Output - Run (Test) x
cd C:\Users\vaagdevi\Documents\NetBeansProjects\Test; "JAVA_HOME=C:\\Program Files\\Java\\jdk-17" cmd /c "\"C:\\Program Files
Scanning for projects...

-----< com.mycompany:Test >-----
Building Test 1.0-SNAPSHOT
  from pom.xml
-----[ jar ]-----

--- resources:3.3.1:resources (default-resources) @ Test ---
skip non existing resourceDirectory C:\Users\vaagdevi\Documents\NetBeansProjects\Test\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ Test ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ Test ---
Shape constructor called
Circle constructor called
Shape constructor called
Rectangle constructor called
Circle color is Redand area is : 15.205308443374602
Rectangle color is Yellowand area is : 8.0

BUILD SUCCESS

Total time: 2.103 s
Finished at: 2024-03-30T08:17:47+05:30

```

3. Write a Java program to handle checked and unchecked exceptions. Also, demonstrate the usage of custom exceptions in real time scenario.

Source Code:

// Custom Checked Exception

```
class InsufficientBalanceException extends Exception {  
    public InsufficientBalanceException(String message) {  
        super(message);  
    }  
}
```

// Custom Unchecked Exception

```
class InvalidWithdrawalAmountException extends RuntimeException {  
    public InvalidWithdrawalAmountException(String message) {  
        super(message);  
    }  
}
```

// BankAccount class representing a simple bank account

```
class BankAccount {  
    private double balance;  
  
    public BankAccount(double initialBalance) {  
        this.balance = initialBalance;  
    }  
}
```

// Method to withdraw money with checked exception handling

```
public void withdraw(double amount) throws InsufficientBalanceException {
```

```
        if (amount > balance) {  
            throw new InsufficientBalanceException("Insufficient balance in the account.");  
        }  
  
        balance -= amount;  
  
        System.out.println("Withdrawal of $" + amount + " successful. Remaining balance: $" +  
balance);  
    }  
}
```

// Method to deposit money without exception handling

```
public void deposit(double amount) {  
  
    balance += amount;  
  
    System.out.println("Deposit of $" + amount + " successful. Updated balance: $" +  
balance);  
}
```

// Method to withdraw money with unchecked exception handling

```
public void withdrawUnchecked(double amount) {  
  
    if (amount <= 0) {  
        throw new InvalidWithdrawalAmountException("Invalid withdrawal amount.");  
    }  
  
    if (amount > balance) {  
        throw new RuntimeException("Insufficient balance in the account.");  
    }  
  
    balance -= amount;  
  
    System.out.println("Unchecked withdrawal of $" + amount + " successful. Remaining  
balance: $" + balance);  
}
```

```
}  
}
```

```
public class ExceptionHandlingDemo {  
    public static void main(String[] args) {  
        try {  
            // Handling checked exception  
  
            BankAccount account = new BankAccount(1000);  
            account.withdraw(1200);  
  
            // Handling unchecked exception  
  
            account.withdrawUnchecked(-100);  
  
        } catch (InsufficientBalanceException e) {  
            System.out.println("Caught checked exception: " + e.getMessage());  
        } catch (InvalidWithdrawalAmountException e) {  
            System.out.println("Caught unchecked exception: " + e.getMessage());  
        } catch (RuntimeException e) {  
            System.out.println("Caught unchecked exception: " + e.getMessage());  
        } finally {  
            System.out.println("Execution completed.");  
        }  
    }  
}
```



```
Output - Run (ExceptionHandlingDemo) x
cd C:\Users\vaagdevi\Documents\NetBeansProjects\ExceptionHandlingDemo; "JAVA_HO
Scanning for projects...

-----< com.mycompany:ExceptionHandlingDemo >-----
Building ExceptionHandlingDemo 1.0-SNAPSHOT
  from pom.xml
-----[ jar ]-----

--- resources:3.3.1:resources (default-resources) @ ExceptionHandlingDemo ---
skip non existing resourceDirectory C:\Users\vaagdevi\Documents\NetBeansProject:

--- compiler:3.11.0:compile (default-compile) @ ExceptionHandlingDemo ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ ExceptionHandlingDemo ---
Caught checked exception: Insufficient balance in the account.
Execution completed.

BUILD SUCCESS

Total time:  1.846 s
Finished at: 2024-03-30T05:53:14+05:30
```

4. Write a Java program on Random Access File class to perform different read and write operations.

Source Code:

```
import java.io.RandomAccessFile;

public class RandomAccessFileDemo {

    public static void main(String[] args) {

        // File path for demonstration (change it as needed)

        String filePath = "randomAccessFileDemo.txt";
```

```

try {

    // Writing data to the file using RandomAccessFile
    writeToFile(filePath, "Hello, RandomAccessFile!");

    // Reading data from the file using RandomAccessFile
    String dataRead = readFromFile(filePath);
    System.out.println("Data read from file: " + dataRead);

    // Updating data at a specific position in the file
    updateFile(filePath, "Updated Data", 7);

    // Reading the updated data from the file
    dataRead = readFromFile(filePath);
    System.out.println("Updated data read from file: " + dataRead);

} catch (Exception e) {
    e.printStackTrace();
}

}

// Method to write data to a file using RandomAccessFile
private static void writeToFile(String filePath, String data) throws Exception {
    try (RandomAccessFile raf = new RandomAccessFile(filePath, "rw")) {
        raf.writeUTF(data);
    }
}

```

```
}
```

```
// Method to read data from a file using RandomAccessFile
```

```
private static String readFromFile(String filePath) throws Exception {
```

```
    try (RandomAccessFile raf = new RandomAccessFile(filePath, "r")) {
```

```
        return raf.readUTF();
```

```
    }
```

```
}
```

```
// Method to update data at a specific position in the file using RandomAccessFile
```

```
private static void updateFile(String filePath, String newData, long position) throws  
Exception {
```

```
    try (RandomAccessFile raf = new RandomAccessFile(filePath, "rw")) {
```

```
        // Move the file pointer to the specified position
```

```
        raf.seek(position);
```

```
        // Write the new data at the specified position
```

```
        raf.writeUTF(newData);
```

```
    }
```

```
}
```

```
}
```

```
Output - Run (RandomAccessFileDemo) x
[+] Building RandomAccessFileDemo 1.0-SNAPSHOT
    from pom.xml
    -----[ jar ]-----
    [-] --- resources:3.3.1:resources (default-resources) @ RandomAccessFileDemo ---
        skip non existing resourceDirectory C:\Users\vaagdevi\Documents\NetBeansProjects\RandomAccessFileDemo\src\main\resources
    [-] --- compiler:3.11.0:compile (default-compile) @ RandomAccessFileDemo ---
        Nothing to compile - all classes are up to date
    [-] --- exec:3.1.0:exec (default-cli) @ RandomAccessFileDemo ---
        Data read from file: Hello, RandomAccessFile!
        Updated data read from file: Hello!Updated DataFile!
    -----
    BUILD SUCCESS
    -----
    Total time: 2.552 s
    Finished at: 2024-03-30T06:21:20+05:30
    -----
```

5. Write a Java program to demonstrate the working of different collection classes. [Use package structure to store multiple classes].

Source Code:

```
// Define a package named 'collectionsdemo'

package collectionsdemo;

// Import necessary classes from the 'java.util' package

import java.util.ArrayList;

import java.util.LinkedList;

import java.util.HashMap;

import java.util.HashSet;

import java.util.Iterator;


// Create a class named 'CollectionDemo' in the 'collectionsdemo' package
```

```
public class CollectionDemo {

    public static void main(String[] args) {

        // Demonstrate working of different collection classes

        // ArrayList example

        ArrayList<String> arrayList = new ArrayList<>();

        arrayList.add("Apple");

        arrayList.add("Banana");

        arrayList.add("Orange");

        System.out.println("ArrayList: " + arrayList);

        // LinkedList example

        LinkedList<Integer> linkedList = new LinkedList<>();

        linkedList.add(10);

        linkedList.add(20);

        linkedList.add(30);

        System.out.println("LinkedList: " + linkedList);

        // HashMap example

        HashMap<Integer, String> hashMap = new HashMap<>();

        hashMap.put(1, "One");

        hashMap.put(2, "Two");

        hashMap.put(3, "Three");

        System.out.println("HashMap: " + hashMap)

        // HashSet example

        HashSet<Double> hashSet = new HashSet<>();
```

```
hashSet.add(1.5);

hashSet.add(2.5);

hashSet.add(3.5);

System.out.println("HashSet: " + hashSet);

// Iterate over the elements using Iterator

System.out.println("\nIterating over elements:");

// Iterate over ArrayList

System.out.println("ArrayList:");

Iterator<String> arrayListIterator = arrayList.iterator();

while (arrayListIterator.hasNext()) {

    System.out.println(arrayListIterator.next());

}

// Iterate over LinkedList

System.out.println("\nLinkedList:");

Iterator<Integer> linkedListIterator = linkedList.iterator();

while (linkedListIterator.hasNext()) {

    System.out.println(linkedListIterator.next());

}

// Iterate over HashMap

System.out.println("\nHashMap:");

Iterator<Integer> hashMapIterator = hashMap.keySet().iterator();

while (hashMapIterator.hasNext()) {

    int key = hashMapIterator.next();

    System.out.println("Key: " + key + ", Value: " + hashMap.get(key));

}
```

```

// Iterate over HashSet

System.out.println("\nHashSet:");

Iterator<Double> hashSetIterator = hashSet.iterator();

while (hashSetIterator.hasNext()) {

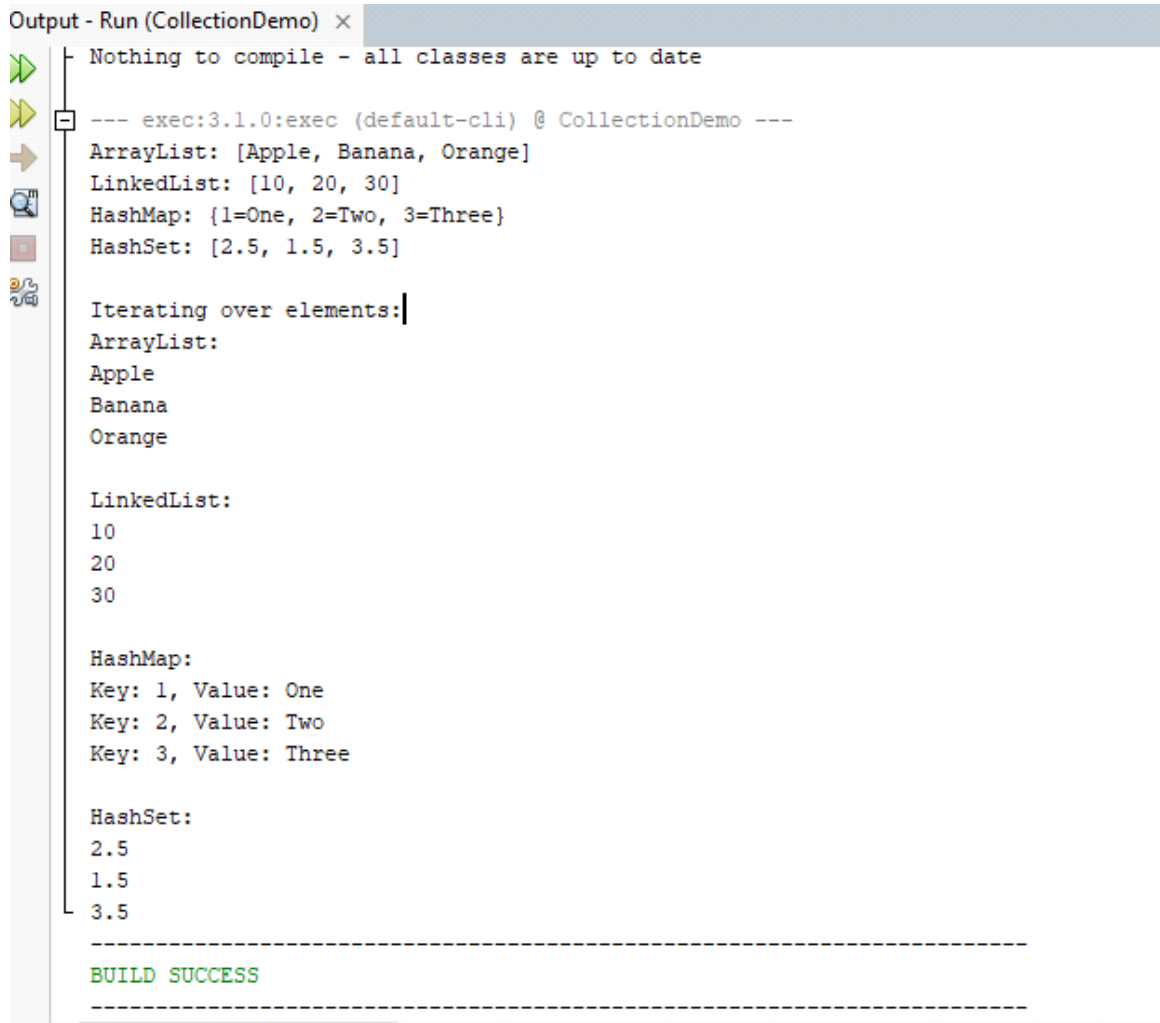
    System.out.println(hashSetIterator.next());

}

}

}

```



```

Output - Run (CollectionDemo) x
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ CollectionDemo ---
ArrayList: [Apple, Banana, Orange]
LinkedList: [10, 20, 30]
HashMap: {1=One, 2=Two, 3=Three}
HashSet: [2.5, 1.5, 3.5]

Iterating over elements:
ArrayList:
Apple
Banana
Orange

LinkedList:
10
20
30

HashMap:
Key: 1, Value: One
Key: 2, Value: Two
Key: 3, Value: Three

HashSet:
2.5
1.5
3.5

-----
BUILD SUCCESS
-----

```


6. Write a program to synchronize the threads acting on the same object. [Consider the example of any reservations like railway, bus, movie ticket booking, etc.]

Source Code:

```
class MovieBookingSystem {  
    private int availableSeats;  
  
    public MovieBookingSystem(int totalSeats) {  
        this.availableSeats = totalSeats;  
    }  
  
    // Synchronized method to book seats  
    public synchronized void bookSeats(int numSeats, String customerName) {  
        if (numSeats > 0 && numSeats <= availableSeats) {  
            System.out.println(customerName + " booked " + numSeats + " seat(s).");  
            availableSeats -= numSeats;  
            System.out.println("Available seats: " + availableSeats);  
        } else {  
            System.out.println("Sorry, " + customerName + ". Not enough seats available.");  
        }  
    }  
}  
  
class BookingThread extends Thread {
```

```

private MovieBookingSystem bookingSystem;

private int numSeats;

private String customerName;


    public BookingThread(MovieBookingSystem bookingSystem, int numSeats, String
customerName) {

        this.bookingSystem = bookingSystem;

        this.numSeats = numSeats;

        this.customerName = customerName;

    }


    @Override

    public void run() {

        bookingSystem.bookSeats(numSeats, customerName);

    }

}


public class MovieBookingDemo {

    public static void main(String[] args) {

        // Create a movie booking system with 50 available seats

        MovieBookingSystem bookingSystem = new MovieBookingSystem(50);


        // Create multiple threads for booking seats

        BookingThread thread1 = new BookingThread(bookingSystem, 3, "Customer1");

        BookingThread thread2 = new BookingThread(bookingSystem, 5, "Customer2");

```

```

        BookingThread thread3 = new BookingThread(bookingSystem, 2, "Customer3");

        // Start the threads

        thread1.start();

        thread2.start();

        thread3.start();

    }

}

```



```

Output - Run (MovieBookingDemo) x
cd C:\Users\vaagdevi\Documents\NetBeansProjects\MovieBookingDemo; "JAVA_HOME=C:\Program Files\Java\jdk-17" cmd /c "
Scanning for projects...

-----< com.mycompany:MovieBookingDemo >-----
Building MovieBookingDemo 1.0-SNAPSHOT
from pom.xml
-----[ jar ]-----

--- resources:3.3.1:resources (default-resources) @ MovieBookingDemo ---
skip non existing resourceDirectory C:\Users\vaagdevi\Documents\NetBeansProjects\MovieBookingDemo\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ MovieBookingDemo ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ MovieBookingDemo ---
Customer1 booked 3 seat(s).
Available seats: 47
Customer3 booked 2 seat(s).
Available seats: 45
Customer2 booked 5 seat(s).
Available seats: 40

BUILD SUCCESS

Total time: 1.885 s
Finished at: 2024-03-30T06:38:12+05:30

```

7. Write a program to perform CRUD operations on the student table in a database using JDBC.

Source Code:

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.Scanner;


public class StudentCRUD {


    // JDBC database URL, username, and password

    private static final String JDBC_URL =
"jdbc:mysql://localhost:3306/your_database_name";

    private static final String USERNAME = "your_username";

    private static final String PASSWORD = "your_password";


    // JDBC variables for opening, closing, and managing the database connection

    private static Connection connection;


    public static void main(String[] args) {

        try {

            // Open a connection

            connection = DriverManager.getConnection(JDBC_URL, USERNAME,
PASSWORD);


            // Create a Scanner for user input
```

```
Scanner scanner = new Scanner(System.in);
```

```
while (true) {
```

```
    System.out.println("\nStudent CRUD Operations:");
```

```
    System.out.println("1. Create Student");
```

```
    System.out.println("2. Read Students");
```

```
    System.out.println("3. Update Student");
```

```
    System.out.println("4. Delete Student");
```

```
    System.out.println("5. Exit");
```

```
    System.out.print("Choose an option: ");
```

```
    int choice = scanner.nextInt();
```

```
    switch (choice) {
```

```
        case 1:
```

```
            createStudent(scanner);
```

```
            break;
```

```
        case 2:
```

```
            readStudents();
```

```
            break;
```

```
        case 3:
```

```
            updateStudent(scanner);
```

```
            break;
```

```
        case 4:
```

```
            deleteStudent(scanner);
```

```
            break;
```

```
        case 5:
```

```

        System.out.println("Exiting program.");

        System.exit(0);

        break;

    default:

        System.out.println("Invalid option. Please choose again.");

    }

}

} catch (SQLException e) {

    e.printStackTrace();

} finally {

    try {

        if (connection != null && !connection.isClosed()) {

            connection.close();

        }

    } catch (SQLException e) {

        e.printStackTrace();

    }

}

}

private static void createStudent(Scanner scanner) throws SQLException {

    System.out.println("Enter student details:");

    System.out.print("Student ID: ");

    int studentId = scanner.nextInt();

```

```

scanner.nextLine(); // consume the newline

System.out.print("Student Name: ");

String studentName = scanner.nextLine();

// Prepare the SQL statement

String sql = "INSERT INTO student (id, name) VALUES (?, ?)";

try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {

    preparedStatement.setInt(1, studentId);

    preparedStatement.setString(2, studentName);

    // Execute the statement

    int rowsAffected = preparedStatement.executeUpdate();

    System.out.println(rowsAffected + " row(s) inserted.");

}

}

private static void readStudents() throws SQLException {

    // Prepare the SQL statement

    String sql = "SELECT id, name FROM student";

    try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {

        // Execute the query

        ResultSet resultSet = preparedStatement.executeQuery();

        // Process the result set

        while (resultSet.next()) {

```



```

        int studentId = resultSet.getInt("id");

        String studentName = resultSet.getString("name");

        System.out.println("Student ID: " + studentId + ", Name: " + studentName);

    }

}

```

```

private static void updateStudent(Scanner scanner) throws SQLException {

    System.out.println("Enter details for updating a student:");


    System.out.print("Enter student ID to update: ");

    int studentId = scanner.nextInt();

    scanner.nextLine(); // consume the newline


    System.out.print("Enter updated student name: ");

    String updatedName = scanner.nextLine();


    // Prepare the SQL statement

    String sql = "UPDATE student SET name = ? WHERE id = ?";

    try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {

        preparedStatement.setString(1, updatedName);

        preparedStatement.setInt(2, studentId);


        // Execute the statement

        int rowsAffected = preparedStatement.executeUpdate();
    }
}

```

```

        if (rowsAffected > 0) {
            System.out.println(rowsAffected + " row(s) updated.");
        } else {
            System.out.println("No student found with ID: " + studentId);
        }
    }
}

```

```

private static void deleteStudent(Scanner scanner) throws SQLException {
    System.out.print("Enter student ID to delete: ");
    int studentId = scanner.nextInt();

    // Prepare the SQL statement
    String sql = "DELETE FROM student WHERE id = ?";
    try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
        preparedStatement.setInt(1, studentId);

        // Execute the statement
        int rowsAffected = preparedStatement.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println(rowsAffected + " row(s) deleted.");
        } else {
            System.out.println("No student found with ID: " + studentId);
        }
    }
}

```

}

Output:

Student CRUD Operations:

1. Create Student
2. Read Students
3. Update Student
4. Delete Student
5. Exit

Choose an option: 1

Enter student details:

Student ID: 101

Student Name: John Doe

1 row(s) inserted.

Student CRUD Operations:

1. Create Student
2. Read Students
3. Update Student
4. Delete Student
5. Exit

Choose an option: 2

Student Information:

Student ID: 101, Name: John Doe

Student CRUD Operations:

1. Create Student
2. Read Students

3. Update Student

4. Delete Student

5. Exit

Choose an option: 3

Enter details for updating a student:

Enter student ID to update: 101

Enter updated student name: John Smith

1 row(s) updated.

Student CRUD Operations:

1. Create Student

2. Read Students

3. Update Student

4. Delete Student

5. Exit

Choose an option: 4

Enter student ID to delete: 101

1 row(s) deleted.

Student CRUD Operations:

1. Create Student

2. Read Students

3. Update Student

4. Delete Student

5. Exit

Choose an option: 5

Exiting program.

8. 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.class" 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("");
        }
}

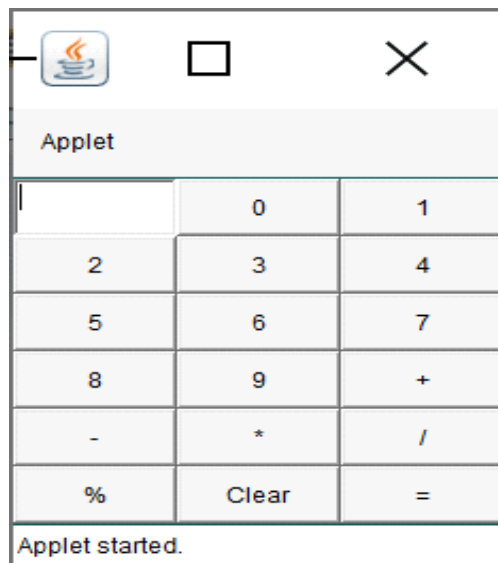
```

```

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:



9. 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:

