DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

II B.Tech II Semester

Subject Name: OBJECT ORIENTED PROGRAMMING THROUGH JAVA LAB
Subject Code: C0513
Regulations: MR-22

Lab Manual

Academic Year: 2024-25

MALLA REDDY ENGINEERING COLLEGE (AUTONOMOUS)
MAIN CAMPUS
(An UGC Autonomous Institution, Approved by AICTE and Affiliated to JNTUH, Hyderabad, Accredited by NAAC with 'A++' Grade (III Cycle))
NBA Accredited Programmes – UG (CE, EEE, ME, ECE, & CSE), PG (CE-SE, EEE, EPS, ME-TE)
Maisammaguda(H), Gundlapochampally Village, Medchal Mandal,
Medchal-Malkajgiri District, Telangana State – 500100

MALLA REDDY ENGINEERING COLLEGE (AUTONOMOUS)

MR22 – ACADEMIC REGULATIONS (CBCS)

for B.Tech. (REGULAR) DEGREE PROGRAMME

Applicable for the students of B.Tech. (Regular) programme admitted from the Academic Year 2022-23 onwards

The B.Tech. Degree of Jawaharlal Nehru Technological University Hyderabad, Hyderabad shall be conferred on candidates who are admitted to the programme and who fulfill all the requirements for the award of the Degree.

VISION OF THE INSTITUTE

To be a premier center of professional education and research, offering quality programs in a socio-economic and ethical ambience.

MISSION OF THE INSTITUTE

- To impart knowledge of advanced technologies using state-of-the-art infrastructural facilities.
- To inculcate innovation and best practices in education, training and research.
- To meet changing socio-economic needs in an ethical ambience.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING – ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

DEPARTMENT VISION

To attain global standards in Computer Science and Engineering education, training and research to meet the growing needs of the industry with socio-economic and ethical considerations.

DEPARTMENT MISSION

- To impart quality education and research to undergraduate and postgraduate students in Computer Science and Engineering.
- To encourage innovation and best practices in Computer Science and Engineering utilizing state-of-the-art facilities.
- To develop entrepreneurial spirit and knowledge of emerging technologies based on ethical values and social relevance.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Graduates will demonstrate technical skills, competency in AI & ML and exhibit team management capability with proper communication in a job environment

PEO2: Graduates will function in their profession with social awareness and responsibility

PEO3: Graduates will interact with their peers in other disciplines in industry and society and contribute to the economic growth of the country

PEO4: Graduates will be successful in pursuing higher studies in engineering or management

PROGRAMME OUTCOMES (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

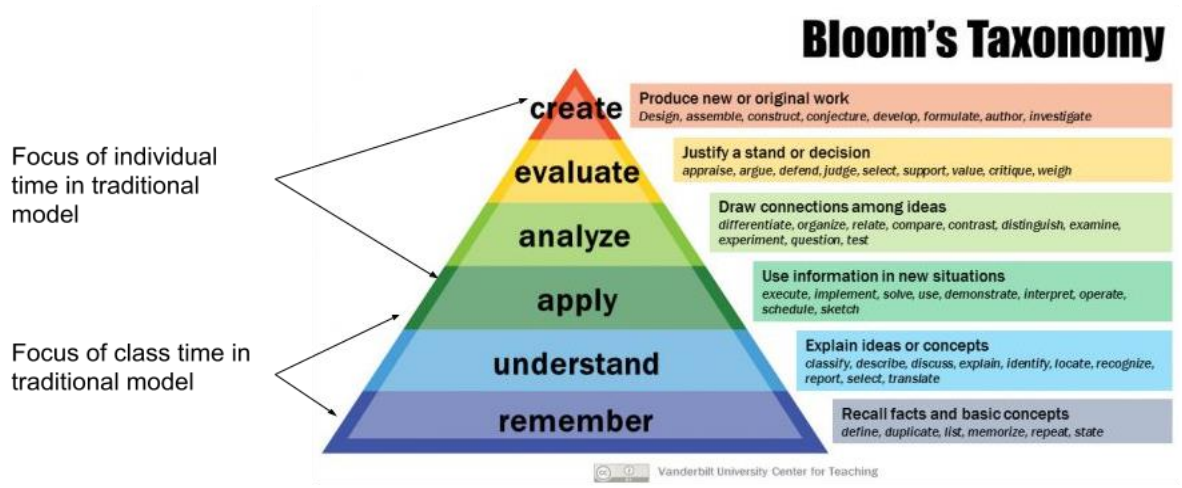
PROGRAMME SPECIFIC OUTCOMES (PSOs)

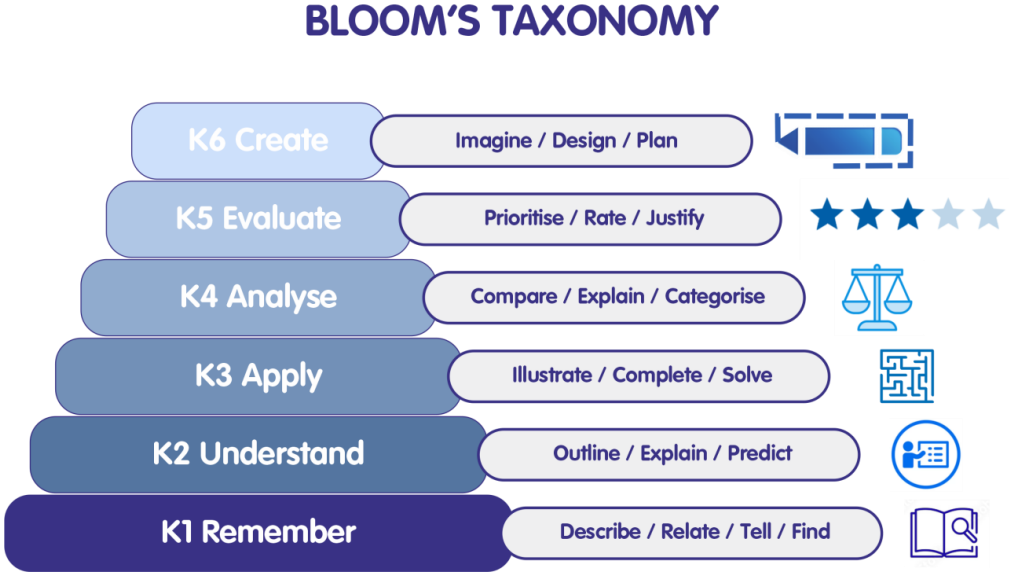
PSO1: Design and develop intelligent automated systems applying mathematical, analytical, programming and operational skills to solve real world problems

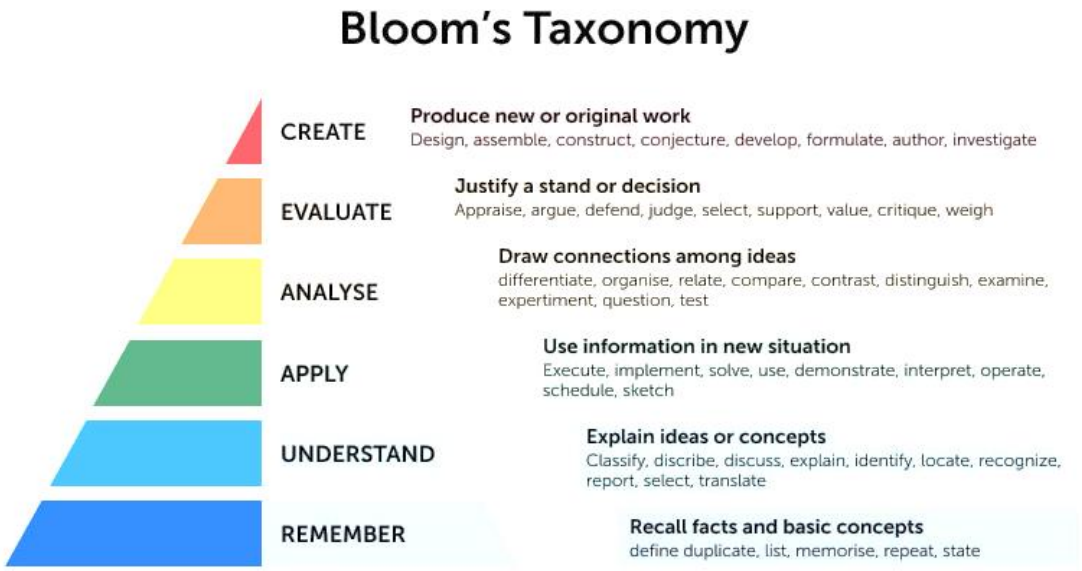
PSO2: Apply machine learning techniques, software tools to conduct experiments, interpret data and to solve complex problems

PSO3: Implement engineering solutions for the benefit of society by the use of AI and ML

BLOOM'S TAXONOMY (BT) TRIANGLE &
BLOOM'S ACTION VERBS







BLOOM'S ACTION VERBS

REVISED Bloom's Taxonomy Action Verbs

Definitions	I. Remembering	II. Understanding	III. Applying	IV. Analyzing	V. Evaluating	VI. Creating
Bloom's Definition	Exhibit memory of previously learned material by recalling facts, terms, basic concepts, and answers.	Demonstrate understanding of facts and ideas by organizing, comparing, translating, interpreting, giving descriptions, and stating main ideas.	Solve problems to new situations by applying acquired knowledge, facts, techniques and rules in a different way.	Examine and break information into parts by identifying motives or causes. Make inferences and find evidence to support generalizations.	Present and defend opinions by making judgments about information, validity of ideas, or quality of work based on a set of criteria.	Compile information together in a different way by combining elements in a new pattern or proposing alternative solutions.
Verbs	• Choose • Define • Find • How • Label • List • Match • Name • Omit • Recall • Relate • Select • Show • Spell • Tell • What • When • Where • Which • Who • Why	• Classify • Compare • Contrast • Demonstrate • Explain • Extend • Illustrate • Infer • Interpret • Outline • Relate • Rephrase • Show • Summarize • Translate	• Apply • Build • Choose • Construct • Develop • Experiment with • Identify • Interview • Make use of • Model • Organize • Plan • Select • Solve • Utilize	• Analyze • Assume • Categorize • Classify • Compare • Conclusion • Contrast • Discover • Dissect • Distinguish • Divide • Examine • Function • Inference • Inspect • List • Motive • Relationships • Simplify • Survey • Take part in • Test for • Theme	• Agree • Appraise • Assess • Award • Choose • Compare • Conclude • Criteria • Criticize • Decide • Deduct • Defend • Determine • Disprove • Estimate • Evaluate • Explain • Importance • Influence • Interpret • Judge • Justify • Mark • Measure • Opinion • Perceive • Prioritize • Prove • Rate • Recommend • Rule on • Select • Support • Value	• Adapt • Build • Change • Choose • Combine • Compile • Compose • Construct • Create • Delete • Design • Develop • Discuss • Elaborate • Estimate • Formulate • Happen • Imagine • Improve • Invent • Make up • Maximize • Minimize • Modify • Original • Originate • Plan • Predict • Propose • Solution • Solve • Suppose • Test • Theory

Anderson, L. W., & Krathwohl, D. R. (2001). A taxonomy for learning, teaching, and assessing, Abridged Edition. Boston, MA: Allyn and Bacon.

2022-23 Onwards (MR-22)	MALLA REDDY ENGINEERING COLLEGE (AUTONOMOUS)	B.Tech. IV Semester		
Code: C0513	OBJECT ORIENTED PROGRAMMING THROUGH JAVA LAB	L	T	P
Credits: 1		-	-	2

Course Objectives:

- To understand OOP principles.
- To understand the Exception Handling mechanism.
- To understand Java collection framework.
- To understand multithreaded programming.
- To understand swing controls in Java.

List of Experiments

1. 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.
2. Write a Java program to demonstrate the OOP principles. [i.e., Encapsulation, Inheritance, Polymorphism and Abstraction]
3. Write a Java program to handle checked and unchecked exceptions. Also, demonstrate the usage of custom exceptions in real time scenario.
4. Write a Java program on Random Access File class to perform different read and write operations.
5. Write a Java program to demonstrate the working of different collection classes. [Use package structure to store multiple classes].
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.]
7. Write a program to perform CRUD operations on the student table in a database using JDBC.
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.
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]

REFERENCE BOOKS:

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

Software Requirements:

- Eclipse IDE / Netbeans IDE, Java Compiler

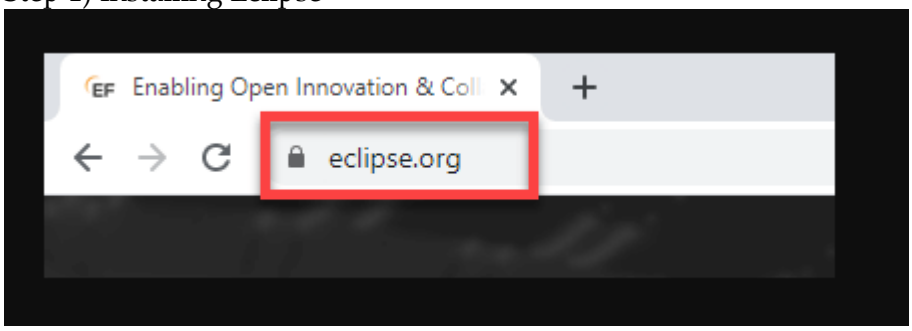
Course Outcomes:

- Able to write the programs for solving real world problems using Java OOP principles.
- Able to write programs using Exceptional Handling approach.
- Able to write multithreaded applications.
- Able to write GUI programs using swing controls in Java

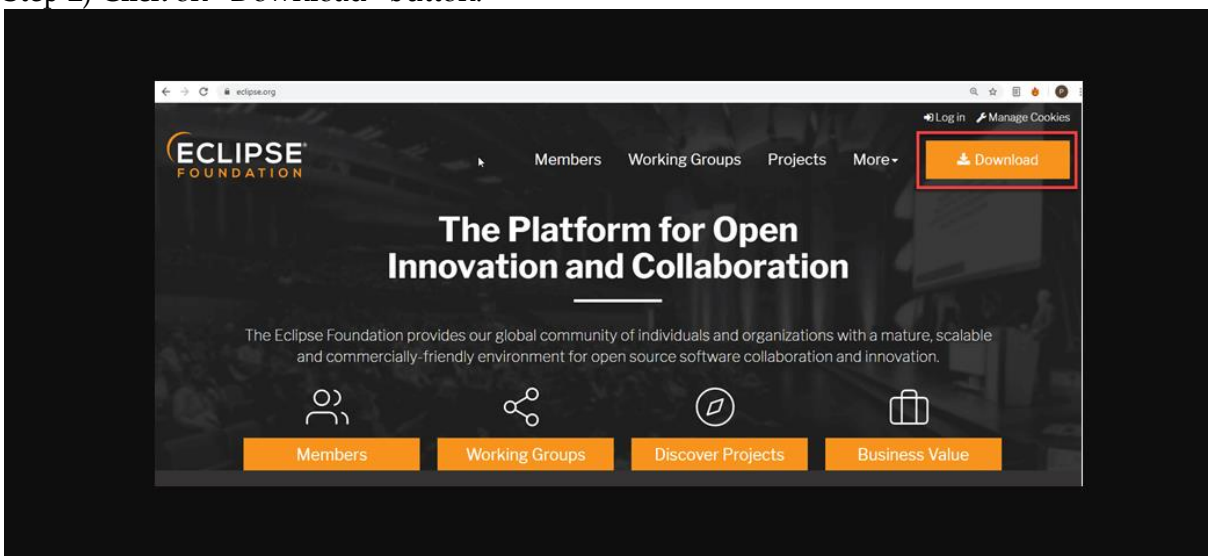
CO- PO, PSO Mapping (3/2/1 indicates strength of correlation) 3-Strong, 2-Medium, 1-Weak															
COs	**Programme Outcomes (POs)**												**PSOs**		
	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**
CO1		3	3	2									3	3	
CO2	3	3	3	2									3	2	
CO3	3	3	3	2									3	2	
CO4	3	3	3	2									3	2	
CO5	3	3	3	2									3	2	

1. 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

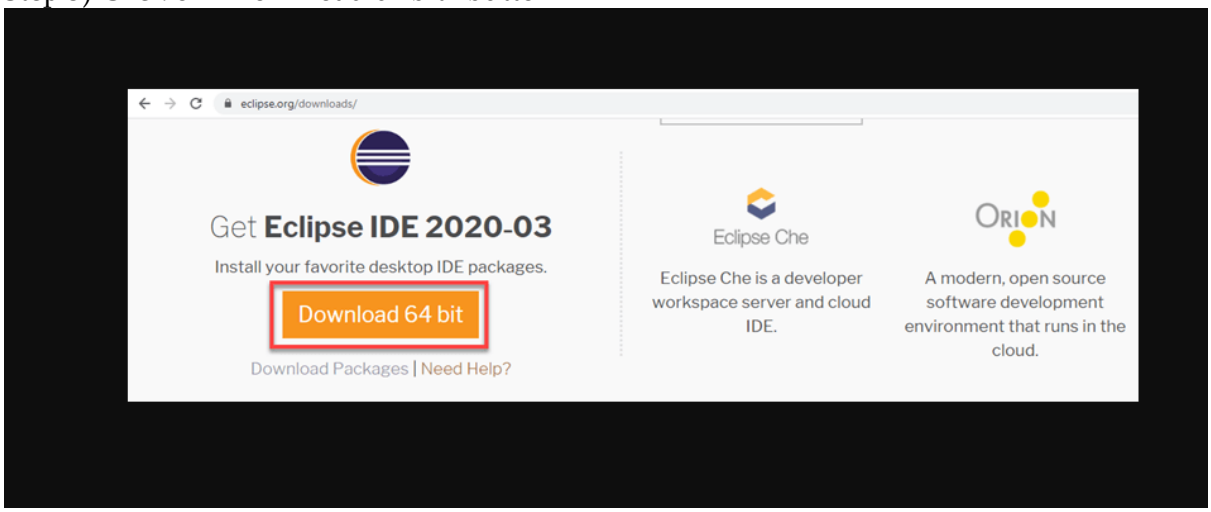
Eclipse Download and Installation Steps:
Step 1) Installing Eclipse



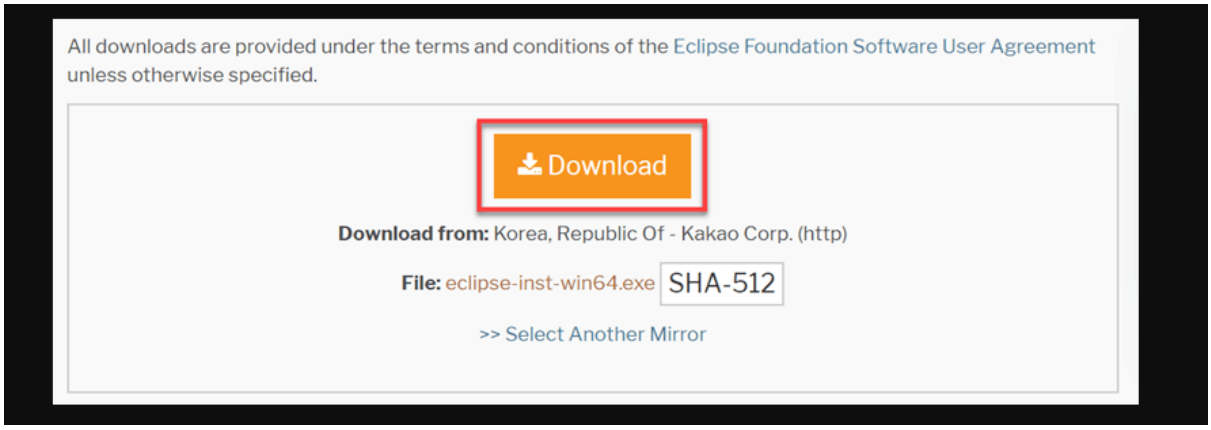
Step 2) Click on "Download" button.



Step 3) Click on "Download 64 bit" button

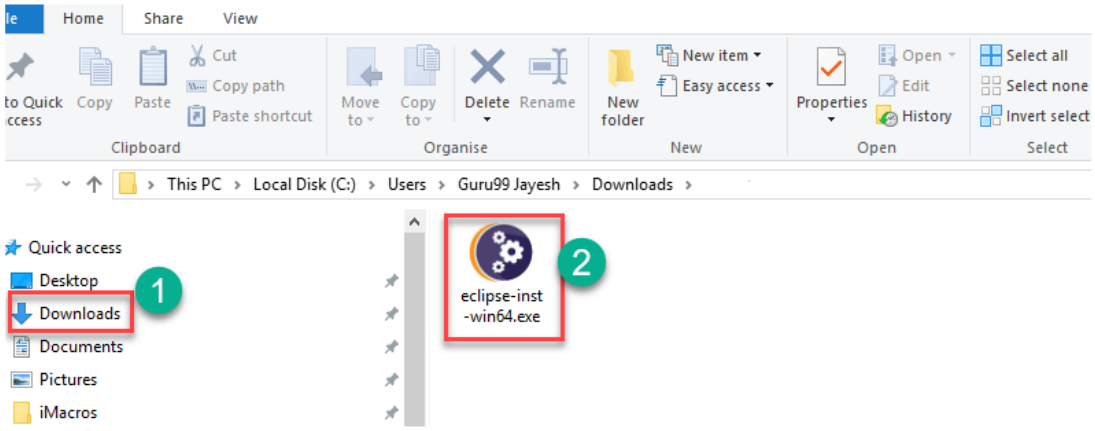


Step 4) Click on "Download" button

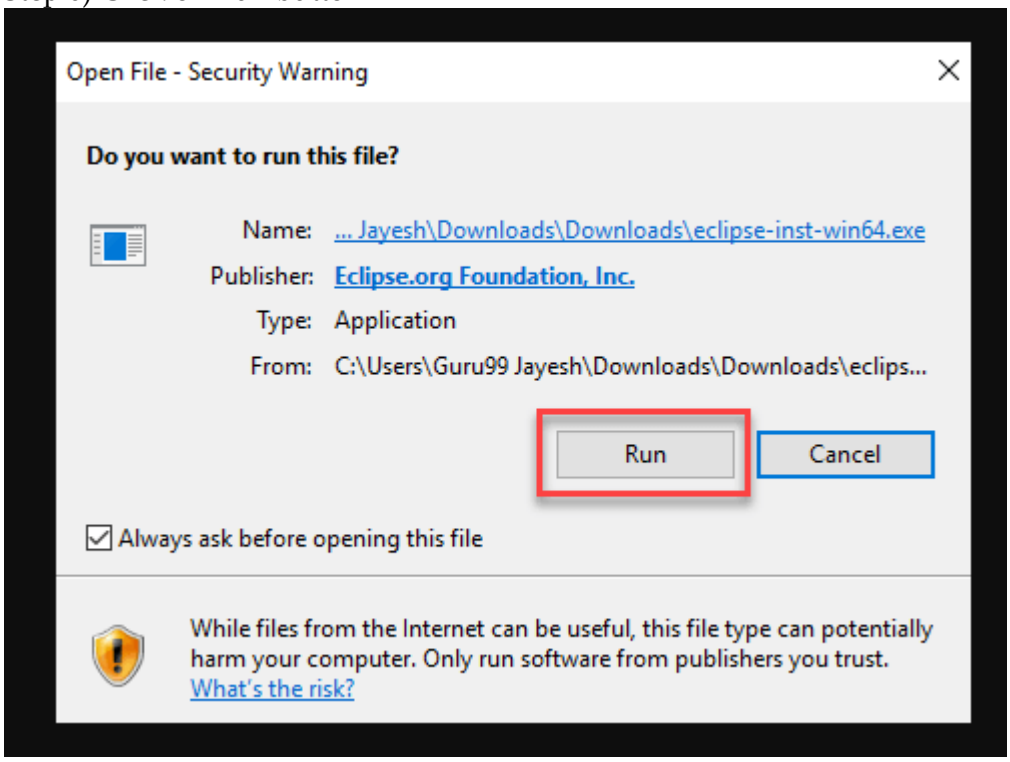


Step 5) Install Eclipse.

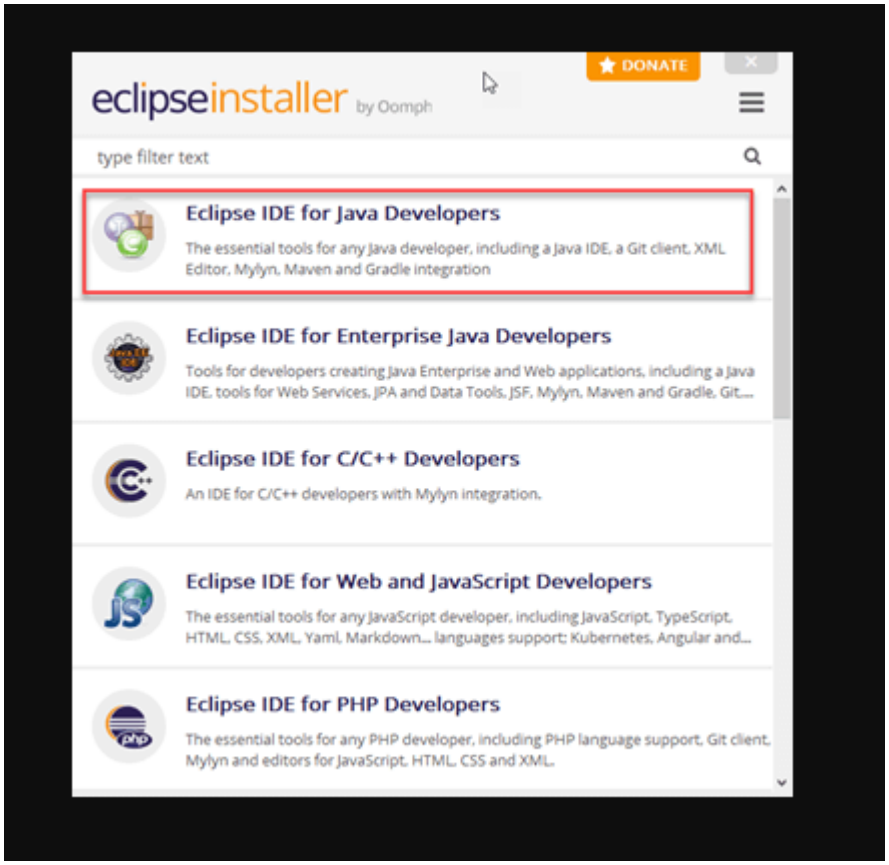
Click on "downloads" in Windows file explorer.
Click on "eclipse-inst-win64.exe" file.



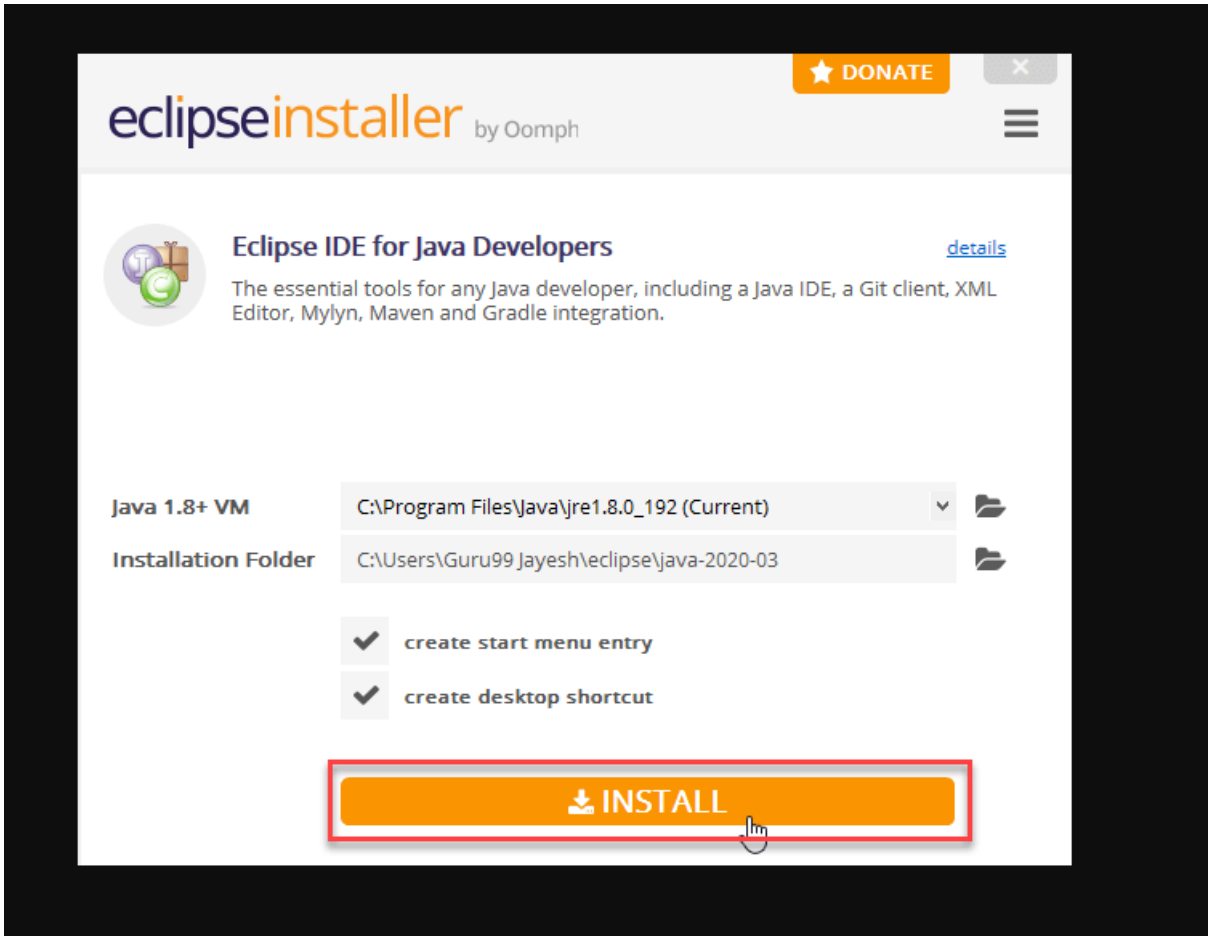
Step 6) Click on Run button



Step 7) Click on "Eclipse IDE for Java Developers"

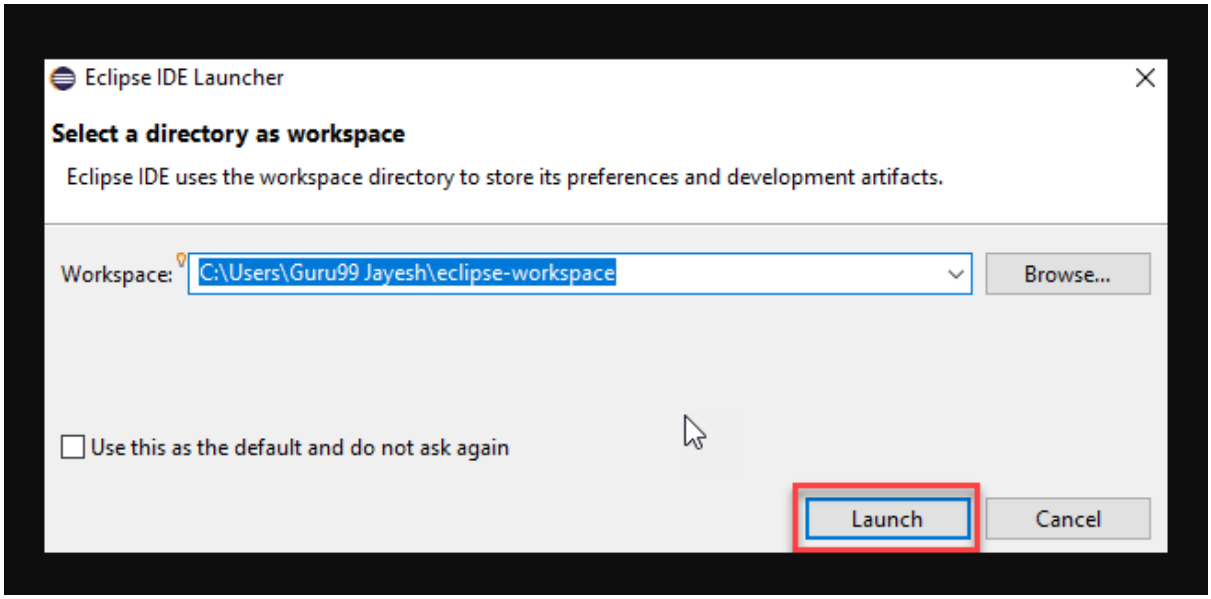


Step 8) Click on "INSTALL" button

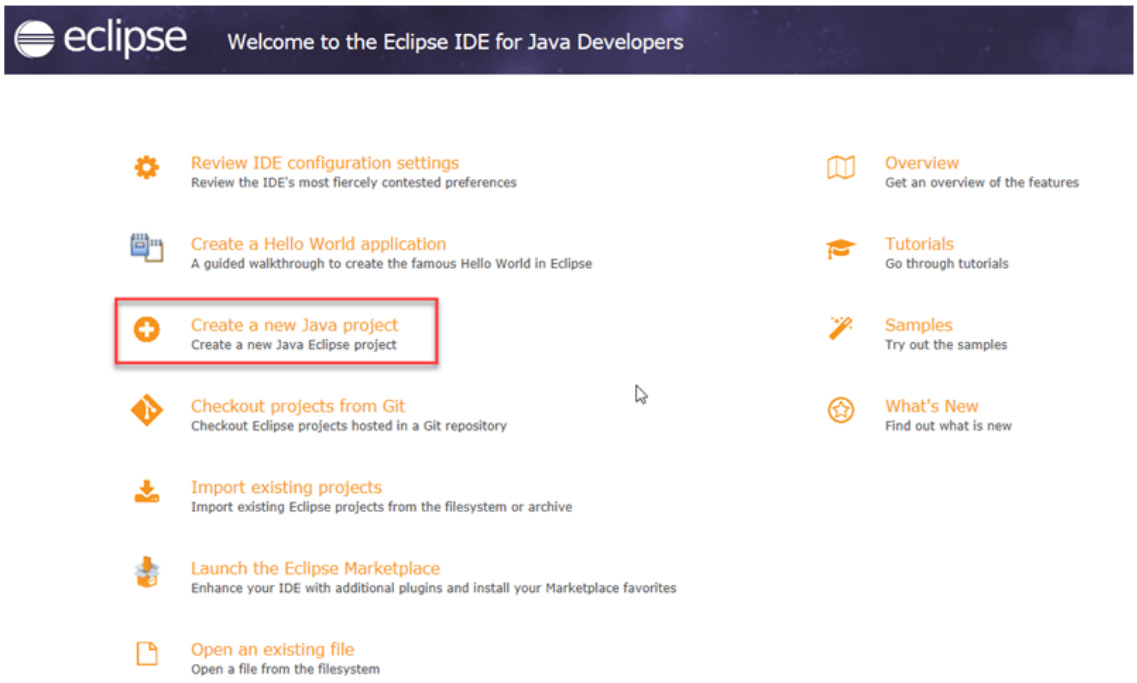


Step 9) Click on "LAUNCH" button.

Step 10) Click on "Launch" button.

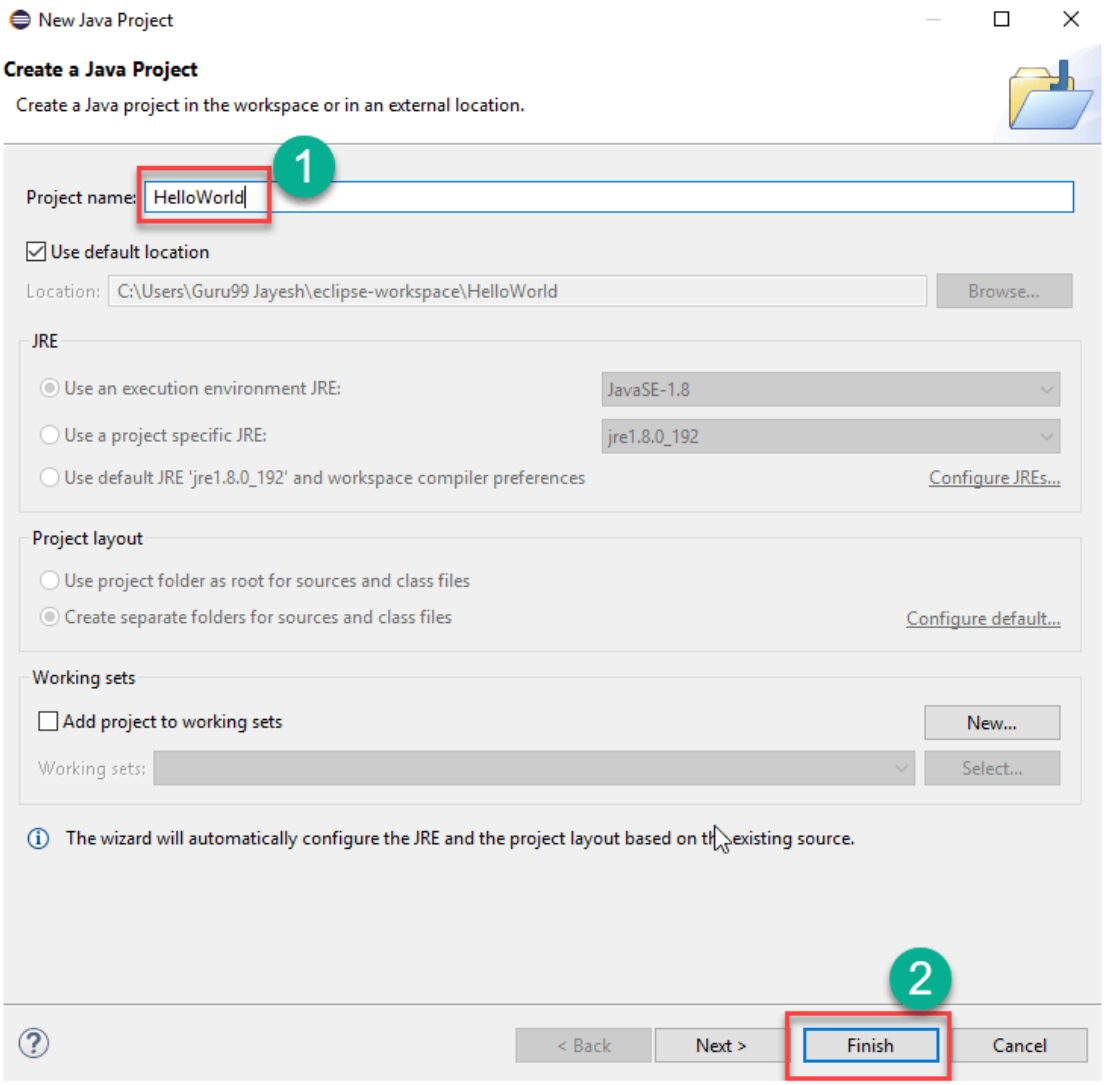


Step 11) Click on "Create a new Java project" link.



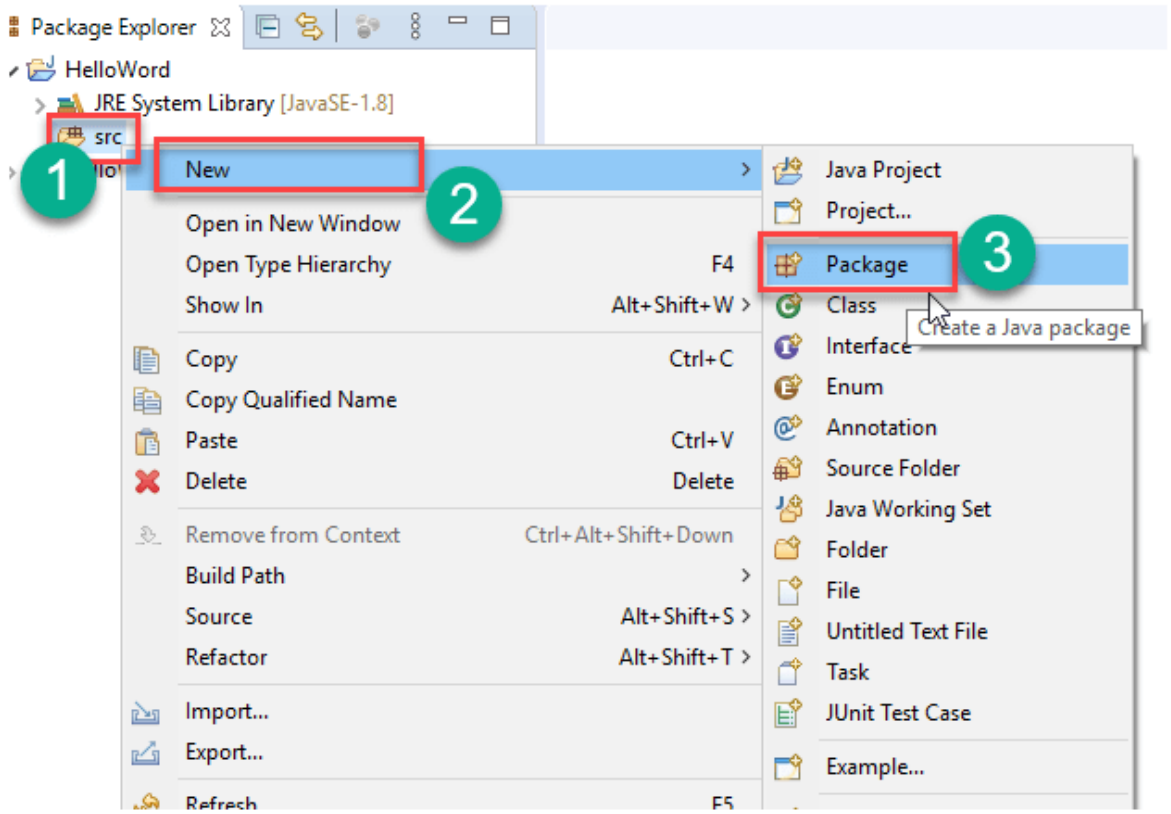
Step 12) Create a new Java Project

Write project name.
Click on "Finish button".



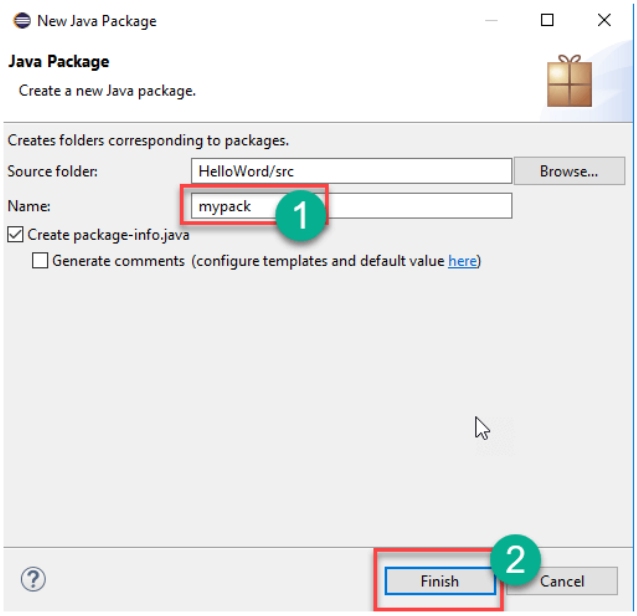
Step 13) Create Java Package.

Goto “src”.
Click on “New”.
Click on “Package”.



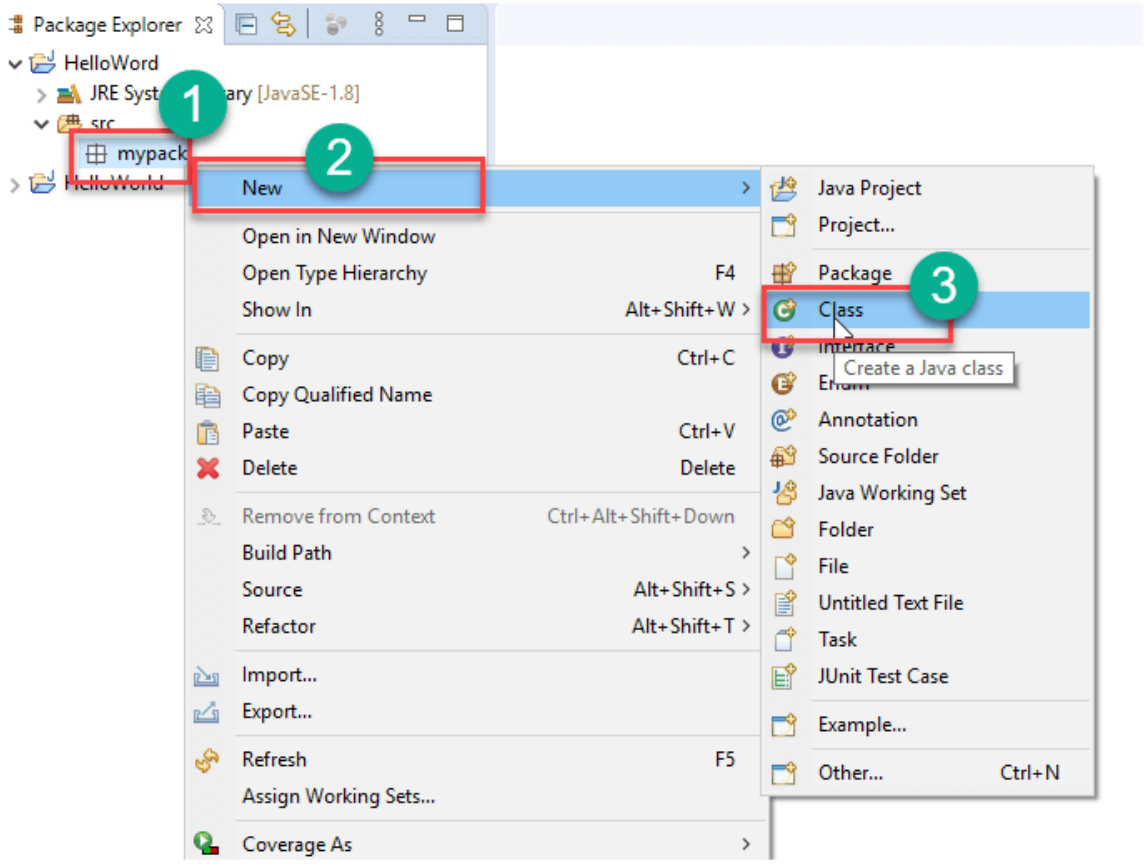
Step 14) Writing package name.

Write name of the package
Click on Finish button.



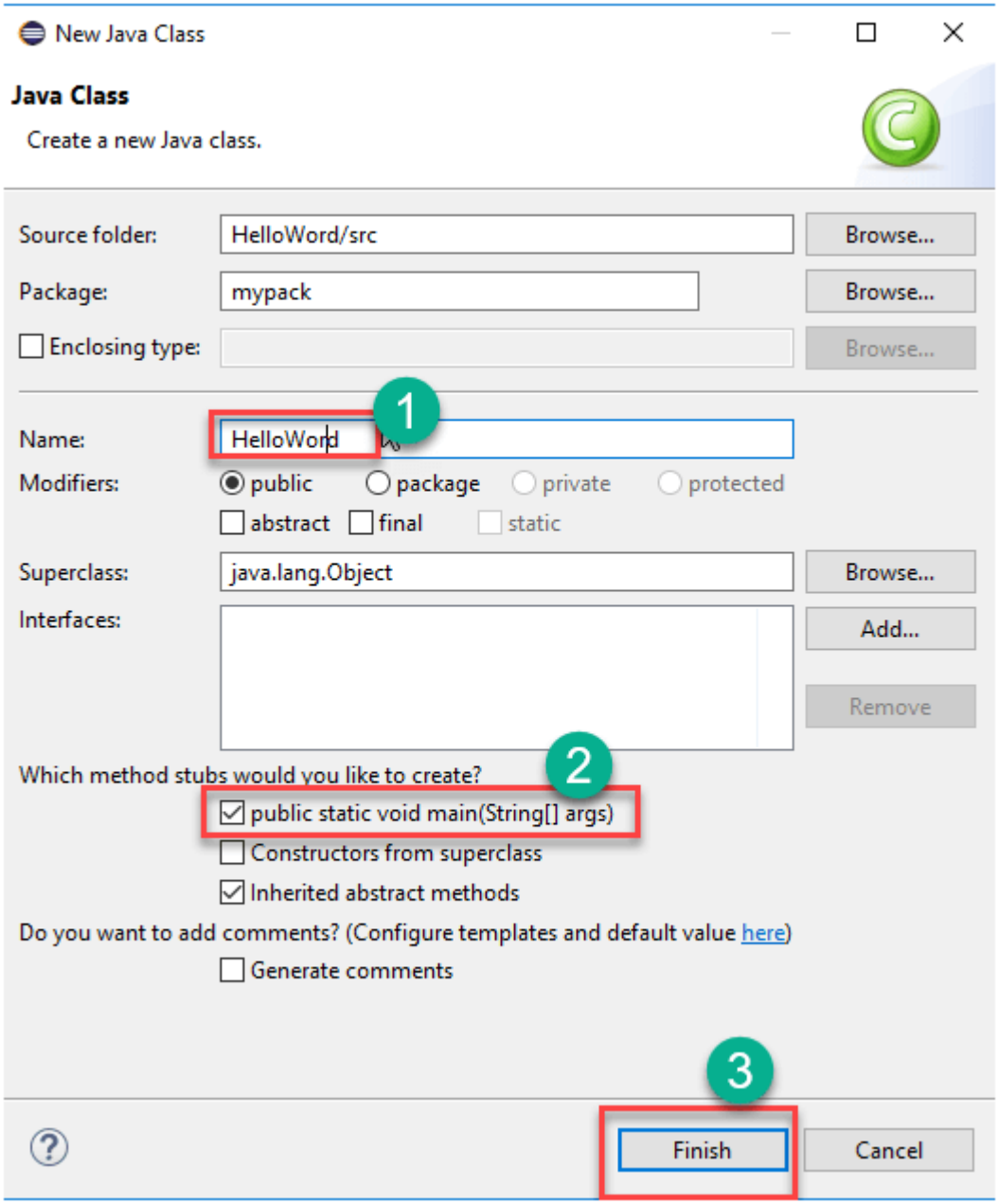
Step 15) Creating Java Class

Click on package you have created.
Click on "New".
Click on "Class".



Step 16) Defining Java Class.

Write class name
Click on "public static void main (String[] args)" checkbox.
Click on "Finish" button.


New Java Class
Java Class
Create a new Java class.
Source folder:
HelloWord/src
Browse...
Package:
mypack
Browse...
Enclosing type:
Browse...
1
Name:
HelloWord
Modifiers:
public
package
private
protected
abstract
final
static
Superclass:
java.lang.Object
Browse...
Interfaces:
Add...
Remove
2
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
3
Finish
Cancel

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

```
// Encapsulation: Using private access modifiers to hide internal implementation details
class Animal {
    private String name;

    public Animal(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void makeSound() {
        System.out.println(name + " makes a sound");
    }
}

// Inheritance: Subclass Dog inheriting from superclass Animal
class Dog extends Animal {
    public Dog(String name) {
        super(name);
    }

    @Override
    public void makeSound() {
        System.out.println(getName() + " barks");
    }
}

// Polymorphism: Method overriding
class Cat extends Animal {
    public Cat(String name) {
        super(name);
    }

    @Override
    public void makeSound() {
        System.out.println(getName() + " meows");
    }
}

// Abstraction: Abstract class Shape with abstract method area()
abstract class Shape {
    public abstract double area();
```

```
}

// Concrete subclass Circle implementing abstract method area()
class Circle extends Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double area() {
        return Math.PI * radius * radius;
    }
}

public class Main {
    public static void main(String[] args) {
        // Encapsulation and Inheritance example
        Dog dog = new Dog("Buddy");
        dog.makeSound(); // Output: Buddy barks

        // Polymorphism example
        Animal cat = new Cat("Whiskers");
        cat.makeSound(); // Output: Whiskers meows

        // Abstraction example
        Circle circle = new Circle(5);
        System.out.println("Area of circle: " + circle.area()); // Output: Area of circle:
78.53981633974483
    }
}
```

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

```
// Custom exception class
class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
}

// Class to demonstrate exception handling
public class ExceptionHandlingDemo {
    // Method to simulate unchecked exception
    public static void simulateUncheckedException() {
        int[] numbers = {1, 2, 3};
        System.out.println(numbers[5]); // Accessing index out of bounds
    }

    // Method to simulate checked exception
    public static void simulateCheckedException() throws InterruptedException {
        Thread.sleep(1000); // InterruptedException is a checked exception
    }

    // Method to simulate custom exception
    public static void validateAge(int age) throws InvalidAgeException {
        if (age < 0 || age > 120) {
            throw new InvalidAgeException("Age must be between 0 and 120.");
        }
        System.out.println("Valid age: " + age);
    }

    public static void main(String[] args) {
        // Handling unchecked exception
        try {
            simulateUncheckedException();
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Unchecked Exception: " + e.getMessage());
        }

        // Handling checked exception
        try {
            simulateCheckedException();
        } catch (InterruptedException e) {
            System.out.println("Checked Exception: " + e.getMessage());
        }

        // Handling custom exception
        try {
            validateAge(150);
```

```
        } catch (InvalidAgeException e) {
            System.out.println("Custom Exception: " + e.getMessage());
        }
    }
}
```

```
Unchecked Exception: Index 5 out of bounds for length 3
Checked Exception: sleep interrupted
Custom Exception: Age must be between 0 and 120.
```

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

```
import java.io.IOException;
import java.io.RandomAccessFile;

public class RandomAccessFileDemo {
   public static void main(String[] args) {
      try {
         // Writing data to a file
         RandomAccessFile file = new RandomAccessFile("data.txt", "rw");
         // Write integers to the file
         file.writeInt(100);
         file.writeInt(200);
         file.writeInt(300);
         file.close();

         // Reading data from the file
         file = new RandomAccessFile("data.txt", "r");
         // Read and print the first integer
         int firstInt = file.readInt();
         System.out.println("First Integer: " + firstInt);
         // Seek to the position of the second integer
         file.seek(4); // 4 bytes per integer
         // Read and print the second integer
         int secondInt = file.readInt();
         System.out.println("Second Integer: " + secondInt);
         // Seek to the position of the third integer
         file.seek(8); // 8 bytes for two integers
         // Read and print the third integer
         int thirdInt = file.readInt();
         System.out.println("Third Integer: " + thirdInt);
         file.close();
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

```
First Integer: 100
Second Integer: 200
Third Integer: 300
```

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

```
import java.util.ArrayList;
import java.util.HashMap;


public class CollectionDemo {
   public static void main(String[] args) {
      // ArrayList Demo
      ArrayList<String> list = new ArrayList<>();
      list.add("Apple");
      list.add("Banana");
      list.add("Orange");
      System.out.println("ArrayList Elements:");
      for (String fruit : list) {
         System.out.println(fruit);
      }


      // HashMap Demo
      HashMap<Integer, String> map = new HashMap<>();
      map.put(1, "Java");
      map.put(2, "Python");
      map.put(3, "C++");
      System.out.println("\nHashMap Elements:");
      for (Integer key : map.keySet()) {
         System.out.println("Key: " + key + ", Value: " + map.get(key));
      }
   }
}
```

```
ArrayList Elements:
Apple
Banana
Orange

HashMap Elements:
Key: 1, Value: Java
Key: 2, Value: Python
Key: 3, Value: C++
```

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

```
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class TicketBooking {
   private int availableSeats = 10; // Total available seats
   private Lock lock = new ReentrantLock(); // Lock for synchronization


   // Method to book tickets
   public void bookTickets(String name, int numSeats) {
      lock.lock(); // Acquire the lock
      try {
         if (numSeats > availableSeats) {
            System.out.println("Sorry, " + name + ". Not enough seats available.");
         } else {
            System.out.println(name + " booked " + numSeats + " tickets.");
            availableSeats -= numSeats;
         }
      } finally {
         lock.unlock(); // Release the lock
      }
   }
}

class BookingThread extends Thread {
   private TicketBooking ticketBooking;
   private String name;
   private int numSeats;

   public BookingThread(TicketBooking ticketBooking, String name, int numSeats) {
      this.ticketBooking = ticketBooking;
      this.name = name;
      this.numSeats = numSeats;
   }

   @Override
   public void run() {
      ticketBooking.bookTickets(name, numSeats);
   }
}

public class Main {
   public static void main(String[] args) {
      TicketBooking ticketBooking = new TicketBooking();


      // Creating multiple threads to book tickets
```

```
        BookingThread thread1 = new BookingThread(ticketBooking, "Alice", 3);
        BookingThread thread2 = new BookingThread(ticketBooking, "Bob", 4);
        BookingThread thread3 = new BookingThread(ticketBooking, "Charlie", 5);


        // Start the threads
        thread1.start();
        thread2.start();
        thread3.start();
    }
}
```

```
Alice booked 3 tickets.
Bob booked 4 tickets.
Sorry, Charlie. Not enough seats available.
```

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

```
import java.sql.*;

public class StudentDatabase {
   public static void main(String[] args) {
      // Step 1: Connect to the database
      String url = "jdbc:mysql://localhost:3306/mydatabase";
      String username = "root";
      String password = "password";

      try (Connection connection = DriverManager.getConnection(url, username, password))
{
         // Step 2: Create a Student Table (if not exists)
         String createTableQuery = "CREATE TABLE IF NOT EXISTS student (id INT
AUTO_INCREMENT PRIMARY KEY, name VARCHAR(255), age INT)";
         try (Statement statement = connection.createStatement()) {
            statement.executeUpdate(createTableQuery);
         }

         // Step 3: Perform CRUD operations
         // Insert a new student
         String insertQuery = "INSERT INTO student (name, age) VALUES (?, ?)";
         try (PreparedStatement preparedStatement =
connection.prepareStatement(insertQuery)) {
            preparedStatement.setString(1, "John");
            preparedStatement.setInt(2, 20);
            preparedStatement.executeUpdate();
         }

         // Read student records
         String selectQuery = "SELECT * FROM student";
         try (Statement statement = connection.createStatement();
             ResultSet resultSet = statement.executeQuery(selectQuery)) {
            System.out.println("Student Records:");
            while (resultSet.next()) {
               int id = resultSet.getInt("id");
               String name = resultSet.getString("name");
               int age = resultSet.getInt("age");
               System.out.println("ID: " + id + ", Name: " + name + ", Age: " + age);
            }
         }

         // Update a student record
         String updateQuery = "UPDATE student SET age = ? WHERE name = ?";
         try (PreparedStatement preparedStatement =
connection.prepareStatement(updateQuery)) {
            preparedStatement.setInt(1, 21);
            preparedStatement.setString(2, "John");
            preparedStatement.executeUpdate();
```

```
            }

            // Delete a student record
            String deleteQuery = "DELETE FROM student WHERE name = ?";
            try (PreparedStatement preparedStatement =
connection.prepareStatement(deleteQuery)) {
                preparedStatement.setString(1, "John");
                preparedStatement.executeUpdate();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

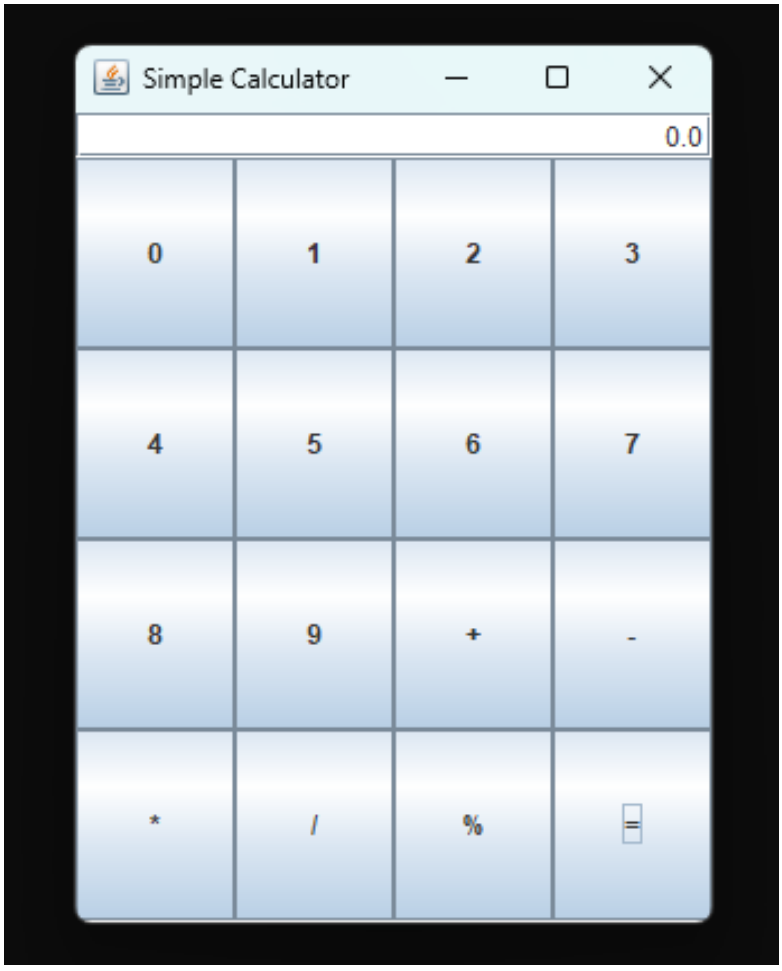
```
Student Records:
ID: 1, Name: John, Age: 20
```

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.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class SimpleCalculator extends JFrame implements ActionListener {
   private JTextField textField;
   private JButton[] digitButtons;
   private JButton addButton, subtractButton, multiplyButton, divideButton, percentButton,
equalsButton;
   private double num1 = 0, num2 = 0;
   private char operator;

   public SimpleCalculator() {
      setTitle("Simple Calculator");
      setSize(300, 400);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setLayout(new BorderLayout());

      // Text field to display result
      textField = new JTextField();
      textField.setHorizontalAlignment(JTextField.RIGHT);
      add(textField, BorderLayout.NORTH);

      // Panel for buttons
      JPanel buttonPanel = new JPanel();
      buttonPanel.setLayout(new GridLayout(4, 4));

      // Digit buttons
      digitButtons = new JButton[10];
      for (int i = 0; i < 10; i++) {
         digitButtons[i] = new JButton(Integer.toString(i));
         digitButtons[i].addActionListener(this);
         buttonPanel.add(digitButtons[i]);
      }

      // Operator buttons
      addButton = new JButton("+");
      subtractButton = new JButton("-");
      multiplyButton = new JButton("*");
      divideButton = new JButton("/");
      percentButton = new JButton("%");
      equalsButton = new JButton("=");
      addButton.addActionListener(this);
      subtractButton.addActionListener(this);
      multiplyButton.addActionListener(this);
      divideButton.addActionListener(this);
```

```
        percentButton.addActionListener(this);
        equalsButton.addActionListener(this);
        buttonPanel.add(addButton);
        buttonPanel.add(subtractButton);
        buttonPanel.add(multiplyButton);
        buttonPanel.add(divideButton);
        buttonPanel.add(percentButton);
        buttonPanel.add(equalsButton);

        add(buttonPanel, BorderLayout.CENTER);

        setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        JButton buttonClicked = (JButton) e.getSource();
        String buttonText = buttonClicked.getText();

        if (buttonText.matches("[0-9]")) {
            textField.setText(textField.getText() + buttonText);
        } else if (buttonText.equals("+") || buttonText.equals("-") || buttonText.equals("*") ||
buttonText.equals("/") || buttonText.equals("%")) {
            num1 = Double.parseDouble(textField.getText());
            operator = buttonText.charAt(0);
            textField.setText("");
        } else if (buttonText.equals("=")) {
            num2 = Double.parseDouble(textField.getText());
            double result = calculate(num1, num2, operator);
            textField.setText(Double.toString(result));
        }
    }


    private double calculate(double num1, double num2, char operator) {
        double result = 0;
        try {
            switch (operator) {
                case '+':
                    result = num1 + num2;
                    break;
                case '-':
                    result = num1 - num2;
                    break;
                case '*':
                    result = num1 * num2;
                    break;
                case '/':
                    if (num2 == 0) {
                        throw new ArithmeticException("Cannot divide by zero");
                    }
```

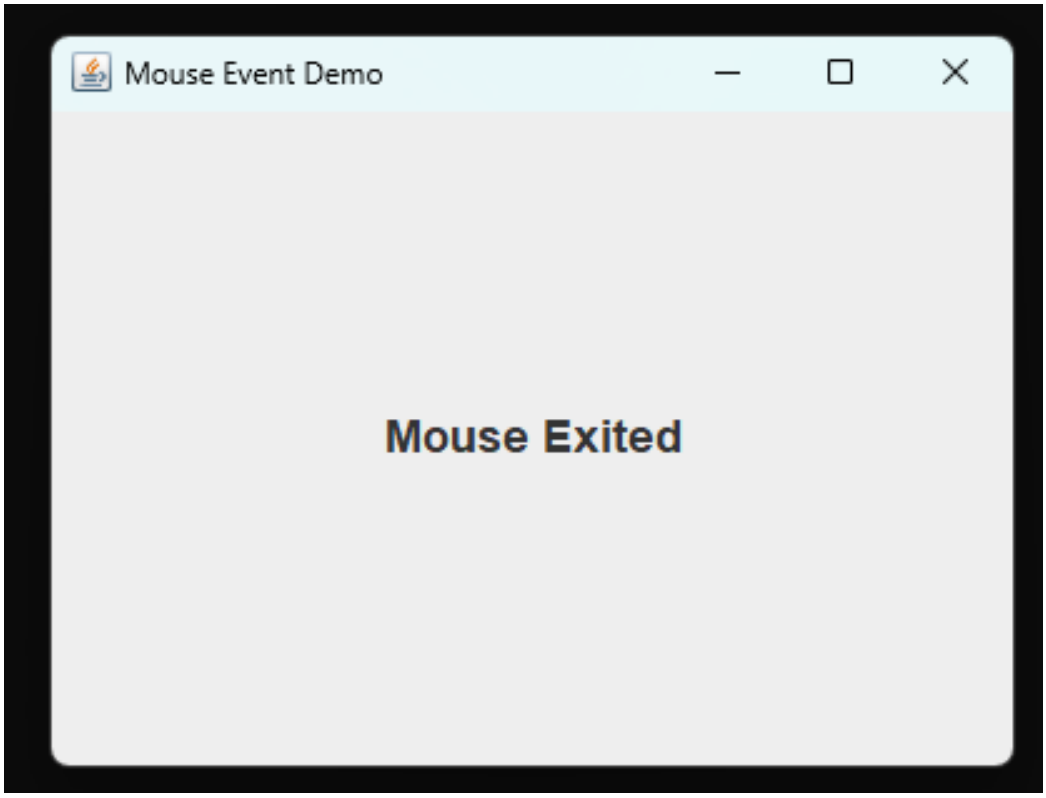
```
                result = num1 / num2;
                break;
            case '%':
                result = num1 % num2;
                break;
        }
    } catch (ArithmeticException e) {
        textField.setText("Error: " + e.getMessage());
    }
    return result;
  }

  public static void main(String[] args) {
    new SimpleCalculator();
  }
}
```



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]

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class MouseEventDemo extends JFrame {
   private JLabel eventLabel;

   public MouseEventDemo() {
      setTitle("Mouse Event Demo");
      setSize(400, 300);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setLocationRelativeTo(null);

      eventLabel = new JLabel("No mouse event", JLabel.CENTER);
      eventLabel.setFont(new Font("Arial", Font.BOLD, 18));
      add(eventLabel, BorderLayout.CENTER);

      addMouseListener(new MouseEventHandler());

      setVisible(true);
   }

   class MouseEventHandler extends MouseAdapter {
      @Override
      public void mouseClicked(MouseEvent e) {
         showEventName("Mouse Clicked");
      }

      @Override
      public void mousePressed(MouseEvent e) {
         showEventName("Mouse Pressed");
      }

      @Override
      public void mouseReleased(MouseEvent e) {
         showEventName("Mouse Released");
      }

      @Override
      public void mouseEntered(MouseEvent e) {
         showEventName("Mouse Entered");
      }

      @Override
      public void mouseExited(MouseEvent e) {
         showEventName("Mouse Exited");
      }
```

```
    }


    private void showEventName(String eventName) {
        eventLabel.setText(eventName);
    }

    public static void main(String[] args) {
        new MouseEventDemo();
    }
}
```



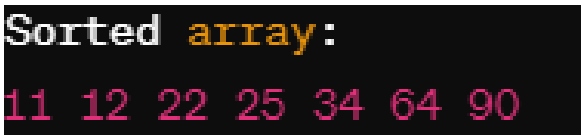
10. File Handling: This program reads data from a text file and counts the number of words in it.

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class WordCount {
   public static void main(String[] args) {
      try {
         File file = new File("input.txt");
         Scanner scanner = new Scanner(file);
         int wordCount = 0;
         while (scanner.hasNext()) {
            scanner.next();
            wordCount++;
         }
         System.out.println("Number of words in the file: " + wordCount);
         scanner.close();
      } catch (FileNotFoundException e) {
         System.out.println("File not found: " + e.getMessage());
      }
   }
}
```

```
Number of words in the file: 6
```

11. Sorting Algorithm: This program implements the bubble sort algorithm to sort an array of integers in ascending order.

```
public class BubbleSort {
   public static void main(String[] args) {
      int[] array = {64, 34, 25, 12, 22, 11, 90};

      for (int i = 0; i < array.length - 1; i++) {
         for (int j = 0; j < array.length - i - 1; j++) {
            if (array[j] > array[j + 1]) {
               int temp = array[j];
               array[j] = array[j + 1];
               array[j + 1] = temp;
            }
         }
      }

      System.out.println("Sorted array:");
      for (int num : array) {
         System.out.print(num + " ");
      }
   }
}
```



13. Sorting Algorithm: This program implements the bubble sort algorithm to sort an array of integers in ascending order.

```
public class MultiThreadingDemo {
   public static void main(String[] args) {
      Thread thread1 = new Thread(() -> {
         for (int i = 1; i <= 10; i++) {
            System.out.println("Thread 1: " + i);
         }
      });

      Thread thread2 = new Thread(() -> {
         for (int i = 1; i <= 10; i++) {
            System.out.println("Thread 2: " + i);
         }
      });

      thread1.start();
      thread2.start();
   }
}
```

```
Thread 1: 1
Thread 1: 2
Thread 1: 3
Thread 1: 4
Thread 1: 5
Thread 1: 6
Thread 1: 7
Thread 1: 8
Thread 1: 9
Thread 1: 10
Thread 2: 1
Thread 2: 2
Thread 2: 3
Thread 2: 4
Thread 2: 5
Thread 2: 6
Thread 2: 7
Thread 2: 8
Thread 2: 9
Thread 2: 10
```