TERMWORK 01
Code:

```
import java.util.ArrayList;
import java.util.Scanner;

class Product {
    private int id;
    private String name;
    private double price;
    private int quantity;

    public Product(int id, String name, double price, int quantity) {
        this.id = id;
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }

    public int getId() { return id; }
    public String getName() { return name; }
    public double getPrice() { return price; }
    public int getQuantity() { return quantity; }

    public void setPrice(double price) { this.price = price; }
    public void setQuantity(int quantity) { this.quantity = quantity; }

    @Override
    public String toString() {
        return "ID: " + id + ", Name: " + name +
                ", Price: $" + price + ", Quantity: " + quantity;
    }
}

public class InventoryManagement {
    private static ArrayList<Product> inventory = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        while (true) {
            System.out.println("\n1. Add Product");
            System.out.println("2. View All Products");
            System.out.println("3. Search Product by Name");
            System.out.println("4. Update Product");
            System.out.println("5. Remove Product");
```

```
            System.out.println("6. Exit");
            System.out.print("Choose an option: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch (choice) {
                case 1 -> addProduct();
                case 2 -> viewAllProducts();
                case 3 -> searchProduct();
                case 4 -> updateProduct();
                case 5 -> removeProduct();
                case 6 -> {
                    System.out.println("Exiting...");
                    System.exit(0);
                }
                default -> System.out.println("Invalid choice, try again.");
            }
        }
    }

    private static void addProduct() {
        System.out.print("Enter Product ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        System.out.print("Enter Product Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter Price: ");
        double price = scanner.nextDouble();
        System.out.print("Enter Quantity: ");
        int quantity = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        inventory.add(new Product(id, name, price, quantity));
        System.out.println("Product added successfully.");
    }

    private static void viewAllProducts() {
        if (inventory.isEmpty()) {
            System.out.println("No products available.");
            return;
        }
        System.out.println("\nAvailable Products:");
        for (Product product : inventory) {
            System.out.println(product);
```

```
    }
}

private static void searchProduct() {
    System.out.print("Enter Product Name to search: ");
    String name = scanner.nextLine();
    boolean found = false;

    for (Product product : inventory) {
        if (product.getName().equalsIgnoreCase(name)) {
            System.out.println(product);
            found = true;
        }
    }
    if (!found) {
        System.out.println("Product not found.");
    }
}
private static void updateProduct() {
    System.out.print("Enter Product ID to update: ");
    int id = scanner.nextInt();
    scanner.nextLine(); // Consume newline

    for (Product product : inventory) {
        if (product.getId() == id) {
            System.out.print("Enter new price: ");
            double newPrice = scanner.nextDouble();
            System.out.print("Enter new quantity: ");
            int newQuantity = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            product.setPrice(newPrice);
            product.setQuantity(newQuantity);
            System.out.println("Product updated successfully.");
            return;
        }
    }
    System.out.println("Product not found.");
}
private static void removeProduct() {
    System.out.print("Enter Product ID to remove: ");
    int id = scanner.nextInt();
    scanner.nextLine(); // Consume newline
```

```
        for (Product product : inventory) {
            if (product.getId() == id) {
                inventory.remove(product);
                System.out.println("Product removed successfully.");
                return;
            }
        }
        System.out.println("Product not found.");
    }
}
```

Sample Output: (put here your output)

TERMWORK02
Code:

```
import java.util.*;
public class TaskScheduler {

    // Define a task class
    static class Task {
        int taskId;
        String priority;
        List<Integer> dependencies;

        public Task(int taskId, String priority, List<Integer> dependencies) {
            this.taskId = taskId;
            this.priority = priority;
            this.dependencies = dependencies;
        }

        public boolean isReadyToExecute(Set<Integer> completedTasks) {
            return completedTasks.containsAll(dependencies);
        }
    }

    // Task Scheduler using Deque
    private Deque<Task> deque;
    private Set<Integer> completedTasks;

    public TaskScheduler() {
        deque = new LinkedList<>();
        completedTasks = new HashSet<>();
    }
```

```
// Add a task to the scheduler
public void addTask(Task task) {
    if (task.priority.equals("HIGH")) {
        deque.addFirst(task); // Add high priority task to the front
    } else {
        deque.addLast(task); // Add low priority task to the back
    }
}

// Process tasks from the Deque
public void processTasks() {
    while (!deque.isEmpty()) {
        Task task = deque.peek(); // Look at the task at the front of the deque

        if (task.isReadyToExecute(completedTasks)) {
            deque.poll(); // Remove task from deque as it is now ready to execute
            System.out.println("Executing Task ID: " + task.taskId);
            completedTasks.add(task.taskId); // Mark the task as completed
        } else {
            // If task is not ready (depends on others), move it to the back
            System.out.println("Task ID: " + task.taskId + " is waiting for dependencies.");
            deque.poll(); // Remove from the front and add to the back
            deque.addLast(task);
        }
    }
}

public static void main(String[] args) {
    TaskScheduler scheduler = new TaskScheduler();

    // Define some tasks
    Task task1 = new Task(1, "HIGH", Arrays.asList());
    Task task2 = new Task(2, "LOW", Arrays.asList(1)); // Depends on Task 1
    Task task3 = new Task(3, "HIGH", Arrays.asList(1)); // Depends on Task 1
    Task task4 = new Task(4, "LOW", Arrays.asList(2)); // Depends on Task 2

    // Add tasks to the scheduler
    scheduler.addTask(task1);
    scheduler.addTask(task2);
    scheduler.addTask(task3);
    scheduler.addTask(task4);

    // Process tasks
    scheduler.processTasks();
```

```
    }
}
```

Sample Output: (put here your output)

TERMWORK 03
Code:

```
import java.util.LinkedList;
import java.util.Queue;

class Storage {
    private final int MAX_CAPACITY = 5;
    private final Queue<Integer> queue = new LinkedList<>();

    public synchronized void produce(int item) throws InterruptedException {
        while (queue.size() == MAX_CAPACITY) {
            System.out.println("Storage is full. Producer is waiting...");
            wait();
        }
        queue.add(item);
        System.out.println("Produced item: " + item);
        notifyAll();  // Notify consumers that an item is available
    }

    public synchronized void consume() throws InterruptedException {
        while (queue.isEmpty()) {
            System.out.println("Storage is empty. Consumer is waiting...");
            wait();
        }
        int item = queue.poll();
        System.out.println("Consumed item: " + item);
        notifyAll();  // Notify producers that there is space available
    }
}

class Producer implements Runnable {
    private final Storage storage;

    public Producer(Storage storage) {
        this.storage = storage;
    }

    @Override
    public void run() {
```

```
        int item = 0;
        while (true) {
            try {
                storage.produce(++item);
                Thread.sleep(500);  // Simulating production delay
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }
}

class Consumer implements Runnable {
    private final Storage storage;

    public Consumer(Storage storage) {
        this.storage = storage;
    }

    @Override
    public void run() {
        while (true) {
            try {
                storage.consume();
                Thread.sleep(700);  // Simulating consumption delay
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }
}

public class ProducerConsumerExample {
    public static void main(String[] args) {
        Storage storage = new Storage();

        Thread producerThread = new Thread(new Producer(storage));
        Thread consumerThread = new Thread(new Consumer(storage));

        producerThread.start();
        consumerThread.start();
    }
}
```

Sample Output: (put here your output)

TERMWORK 04
Code:

```
import java.io.*;
import java.util.Scanner;

public class StreamDemo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;
        do {
            System.out.println("\n--- Stream Operations Menu ---");
            System.out.println("1. FileStream Operations");
            System.out.println("2. ByteArrayStream Operations");
            System.out.println("3. BufferedStream Operations");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch (choice) {
                case 1:
                    fileStreamOperations(scanner);
                    break;
                case 2:
                    byteArrayStreamOperations(scanner);
                    break;
                case 3:
                    bufferedStreamOperations(scanner);
                    break;
                case 4:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } while (choice != 4);
    }

    private static void fileStreamOperations(Scanner scanner) {
        System.out.println("\n--- FileStream Operations ---");
        try (FileOutputStream fos = new FileOutputStream("Fops.txt");
```

```
            FileInputStream fis = new FileInputStream("Fops.txt")) {

            System.out.print("Enter text to write to file: ");
            String input = scanner.nextLine();
            fos.write(input.getBytes());
            System.out.println("Data written to file.");

            System.out.println("Reading data from file:");
            int data;
            while ((data = fis.read()) != -1) {
                System.out.print((char) data);
            }
            System.out.println();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static void byteArrayStreamOperations(Scanner scanner) {
        System.out.println("\n--- ByteArrayStream Operations ---");
        try {
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            System.out.print("Enter text to write to byte array: ");
            String input = scanner.nextLine();
            baos.write(input.getBytes());

            System.out.println("Data written to byte array.");

            ByteArrayInputStream bais = new ByteArrayInputStream(baos.toByteArray());
            System.out.println("Reading data from byte array:");
            int data;
            while ((data = bais.read()) != -1) {
                System.out.print((char) data);
            }
            System.out.println();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static void bufferedStreamOperations(Scanner scanner) {
        System.out.println("\n--- BufferedStream Operations ---");
        try (BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream("Bops.txt"));
```

```
         BufferedInputStream bis = new BufferedInputStream(new FileInputStream("Bops.txt")))
{

        System.out.print("Enter text to write using buffered stream: ");
        String input = scanner.nextLine();
        bos.write(input.getBytes());
        bos.flush();
        System.out.println("Buffered data written to file.");

        System.out.println("Reading buffered data from file:");
        int data;
        while ((data = bis.read()) != -1) {
            System.out.print((char) data);
        }
        System.out.println();
    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
```

Sample Output: (put here your output)

TERMWORK 05
Code:

```
import java.util.Scanner;

public class TW5A {

   // Functional Interface
   @FunctionalInterface
   interface TextTransformer {
      String transform(String text);
   }

   @SuppressWarnings("ConvertToTryWithResources")
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      // Define transformations using block lambda expressions
      TextTransformer toUpperCase = (text) -> {
         return text.toUpperCase();
      };
```

```
TextTransformer reverseText = (text) -> {
    StringBuilder reversed = new StringBuilder();
    for (int i = text.length() - 1; i >= 0; i--) {
        reversed.append(text.charAt(i));
    }
    return reversed.toString();
};

TextTransformer countVowels = (text) -> {
    int count = 0;
    for (char c : text.toLowerCase().toCharArray()) {
        if ("aeiou".indexOf(c) != -1) {
            count++;
        }
    }
    return "Number of vowels: " + count;
};

// Menu and user input loop
while (true) {
    System.out.println("\nEnter a string (or type 'exit' to quit): ");
    String input = scanner.nextLine();

    if (input.equalsIgnoreCase("exit")) {
        System.out.println("Goodbye!");
        break;
    }

    System.out.println("Choose a transformation:");
    System.out.println("1. Convert to Uppercase");
    System.out.println("2. Reverse Text");
    System.out.println("3. Count Vowels");

    int choice = scanner.nextInt();
    scanner.nextLine(); // Consume newline character

    String result;
    result = switch (choice) {
        case 1 -> toUpperCase.transform(input);
        case 2 -> reverseText.transform(input);
        case 3 -> countVowels.transform(input);
        default -> "Invalid choice. Try again!";
    };
```

```
            System.out.println("Result: " + result);
        }
        scanner.close();
    }
}
```

Sample Output: (put here your output)

TERMWORK 06
Code:

```
import java.util.Scanner;

public class TW5B {

    // Functional interface for a mathematical operation
    @FunctionalInterface
    interface Operation {
        double apply(double a, double b);
    }

    // Method to perform a calculation using the provided operation
    @SuppressWarnings("NonPublicExported")
    public static double calculate(double a, double b, Operation operation) {
        return operation.apply(a, b);
    }

    @SuppressWarnings("ConvertToTryWithResources")
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Predefined operations using lambda expressions
        Operation addition = (a, b) -> a + b;
        Operation subtraction = (a, b) -> a - b;
        Operation multiplication = (a, b) -> a * b;
        Operation division = (a, b) -> {
            if (b == 0) {
                throw new ArithmeticException("Division by zero is not allowed.");
            }
            return a / b;
        };

        // Display menu
```

```
System.out.println("Simple Lambda Calculator");
System.out.println("Choose an operation:");
System.out.println("1. Addition");
System.out.println("2. Subtraction");
System.out.println("3. Multiplication");
System.out.println("4. Division");
System.out.println("5. Custom Operation");

int choice = scanner.nextInt();

System.out.println("Enter the first number:");
double num1 = scanner.nextDouble();
System.out.println("Enter the second number:");
double num2 = scanner.nextDouble();

double result;
try {
    switch (choice) {
        case 1 -> {
            result = calculate(num1, num2, addition);
            System.out.println("Result: " + result);
        }
        case 2 -> {
            result = calculate(num1, num2, subtraction);
            System.out.println("Result: " + result);
        }
        case 3 -> {
            result = calculate(num1, num2, multiplication);
            System.out.println("Result: " + result);
        }
        case 4 -> {
            result = calculate(num1, num2, division);
            System.out.println("Result: " + result);
        }
        case 5 -> {
            System.out.println("Enter a custom operation (e.g., a + b):");
            scanner.nextLine(); // Consume newline
            String customOperation = scanner.nextLine();
            Operation custom = (a, b) -> {
                if (customOperation.equals("a + b")) return a + b;
                if (customOperation.equals("a - b")) return a - b;
                if (customOperation.equals("a * b")) return a * b;
                if (customOperation.equals("a / b")) return b != 0 ? a / b : Double.NaN;
                throw new IllegalArgumentException("Unknown operation");
```

```
                };
                result = calculate(num1, num2, custom);
                System.out.println("Result: " + result);
            }
            default -> System.out.println("Invalid choice. Please select a valid operation.");
        }
    } catch (Exception e) {
        System.out.println("Error: " + e.getMessage());
    }

    scanner.close();
  }
}
```

Sample Output: (put here your output)

TERMWORK 07

```
package gui;

import java.awt.Color;
import javax.swing.JOptionPane;
/**
 *
 * @author Prasad Pujar
 */
public class TW6 extends javax.swing.JFrame {

    /**
     * Creates new form TW6
     */
    public TW6() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {
```

```
        jLabel1 = new javax.swing.JLabel();
        jPanel1 = new javax.swing.JPanel();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jButton1 = new javax.swing.JButton();
        jButton4 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Color Changer");
        setBackground(new java.awt.Color(255, 255, 255));
        setBounds(new java.awt.Rectangle(200, 200, 0, 0));
        getContentPane().setLayout(new java.awt.GridLayout(6, 1, 5, 5));

        jLabel1.setFont(new java.awt.Font("Cooper Black", 0, 24)); // NOI18N
        jLabel1.setForeground(new java.awt.Color(255, 0, 153));
        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText("COLOR CHANGER APP");
        jLabel1.setToolTipText("");
        getContentPane().add(jLabel1);

        jPanel1.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 2,
true));

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 280, Short.MAX_VALUE)
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 45, Short.MAX_VALUE)
        );

        getContentPane().add(jPanel1);

        jButton2.setFont(new java.awt.Font("Castellar", 1, 18)); // NOI18N
        jButton2.setForeground(new java.awt.Color(255, 0, 0));
        jButton2.setText("RED");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
```

```
        });
        getContentPane().add(jButton2);

        jButton3.setFont(new java.awt.Font("Castellar", 1, 18)); // NOI18N
        jButton3.setForeground(new java.awt.Color(0, 153, 0));
        jButton3.setText("GREEN");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton3);

        jButton1.setFont(new java.awt.Font("Castellar", 1, 18)); // NOI18N
        jButton1.setForeground(new java.awt.Color(0, 0, 255));
        jButton1.setText("BLUE");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton1);

        jButton4.setFont(new java.awt.Font("Castellar", 1, 18)); // NOI18N
        jButton4.setText("RESET");
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton4);

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton1ActionPerformed
        // TODO add your handling code here:
        jPanel1.setBackground(Color.BLUE);
        jButton1.setBackground(new Color(153, 204, 255));
        jLabel1.setForeground(new Color(0, 51, 204));
        JOptionPane.showMessageDialog(null, "You Clicked BLUE Button");
    }//GEN-LAST:event_jButton1ActionPerformed
```

```
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton2ActionPerformed
        // TODO add your handling code here:
        jPanel1.setBackground(Color.RED);
        jButton2.setBackground(new Color(255, 204, 204));
        jLabel1.setForeground(new Color(204, 0, 51));
        JOptionPane.showMessageDialog(null, "You Clicked RED Button ");
    }//GEN-LAST:event_jButton2ActionPerformed

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton3ActionPerformed
        // TODO add your handling code here:
        jPanel1.setBackground(Color.GREEN);
        jButton3.setBackground(new Color(204, 255, 204));
        jLabel1.setForeground(new Color(0, 153, 0));
        JOptionPane.showMessageDialog(null, "You Clicked GREEN Button");
    }//GEN-LAST:event_jButton3ActionPerformed

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton4ActionPerformed
        // TODO add your handling code here:
        jPanel1.setBackground(new Color(222, 222, 222));
        jButton1.setBackground(Color.lightGray);
        jButton2.setBackground(Color.lightGray);
        jButton3.setBackground(Color.lightGray);
        jLabel1.setForeground(new Color(255, 0, 153));
    }//GEN-LAST:event_jButton4ActionPerformed

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(TW6.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
```

```
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(TW6.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(TW6.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(TW6.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new TW6().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    // End of variables declaration//GEN-END:variables
}
```

Sample Output: (put here your output)

TERMWORK 08
Code:

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

public class CalculatorGUI {
    private final JFrame frame;
```

```
private final JTextField textField;
private double num1, num2, result;
private String operator = "";

public CalculatorGUI() {
    // Create the main frame
    frame = new JFrame("Simple Calculator");
    frame.setSize(300, 200);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setLayout(new BorderLayout());

    // Create the text field for displaying the result
    textField = new JTextField();
    textField.setFont(new Font("Cambria", Font.BOLD, 20));
    textField.setPreferredSize(new Dimension(40,30));
    textField.setEditable(false);
    frame.add(textField, BorderLayout.NORTH);

    // Create the panel for buttons
    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(4, 4, 5, 5));

    // Create buttons and add action listeners
    String[] buttonLabels = {
        "7", "8", "9", "/",
        "4", "5", "6", "*",
        "1", "2", "3", "-",
        "0", "C", "=", "+"
    };

    for (String label : buttonLabels) {
        JButton button = new JButton(label);
        button.setFont(new Font("Cambria", Font.BOLD, 15));
        button.addActionListener(new ButtonClickListener());
        panel.add(button);
    }

    frame.add(panel, BorderLayout.CENTER);
    frame.setVisible(true);
}

// Inner class for handling button clicks
private class ButtonClickListener implements ActionListener {
    @Override
```

```
        public void actionPerformed(ActionEvent e) {
            String command = e.getActionCommand();

            if (command.charAt(0) >= '0' && command.charAt(0) <= '9') {
                // If a number is clicked, append it to the text field
                textField.setText(textField.getText() + command);
            } else if (command.equals("C")) {
                // Clear the text field
                textField.setText("");
                num1 = num2 = result = 0;
                operator = "";
            } else if (command.equals("=")) {
                // Calculate the result
                num2 = Double.parseDouble(textField.getText());
                switch (operator) {
                    case "+" -> result = num1 + num2;
                    case "-" -> result = num1 - num2;
                    case "*" -> result = num1 * num2;
                    case "/" -> result = num1 / num2;
                }
                textField.setText(String.valueOf(result));
                operator = ""; // Reset operator
            } else {
                // Store the first number and operator
                if (!operator.isEmpty()) {
                    return; // Prevent multiple operators
                }
                num1 = Double.parseDouble(textField.getText());
                operator = command; // Set operator
                textField.setText(""); // Clear text field for the next number
            }
        }
    }

    public static void main(String[] args) {
        new CalculatorGUI(); // Create the calculator GUI
    }
}
```

Sample Output: (put here your output)

TERMWORK 10
Code: (Java)

```
package com.mycompany.exp9;
```

```
import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

/**
 *
 * @author Prasad Pujar
 */
public class Exp9 extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String name = request.getParameter("name");
        String age = request.getParameter("age");

        // Set response content type
        response.setContentType("text/html");

        // Write response
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h1>Greetings!</h1>");
        out.println("<p>Hello, " + name + "!</p>");
        out.println("<p>You are " + age + " years old.</p>");
        out.println("<p>Welcome to Advanced JAVA LAB's Servlet Programming Concept.</p>");
        out.println("</body></html>");
    }
}

(HTML)
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Input Form</title>
</head>
<body>
    <h1>Enter Your Details</h1>
    <form action="Exp9" method="post">
```

```
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required><br><br>
        <label for="age">Age:</label>
        <input type="number" id="age" name="age" required><br><br>
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

(web.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="6.0" xmlns="https://jakarta.ee/xml/ns/jakartaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd">
    <servlet>
        <servlet-name>Exp9</servlet-name>
        <servlet-class>com.mycompany.exp9.Exp9</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Exp9</servlet-name>
        <url-pattern>/Exp9</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
</web-app>
```

Sample Output: (put here your output)