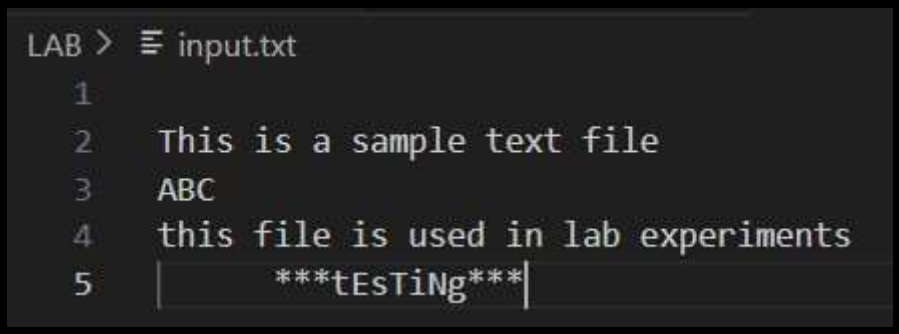
| S.No | Experiment Title | Date | Sign |
|---|---|---|---|
| 6 | **Write a program in java to sort the content of a given text file** | | |
| 7 | **Convert the content of a given file into the uppercase content of the same file.** | | |
| 8 | **Develop an analog clock using applet/java AWT components.** | | |
| 9 | **Develop a scientific calculator using swings.** | | |
| 10 | **Create an editor like MS-Word using swings** | | |
| | | | |
| | | | |
| | | | |

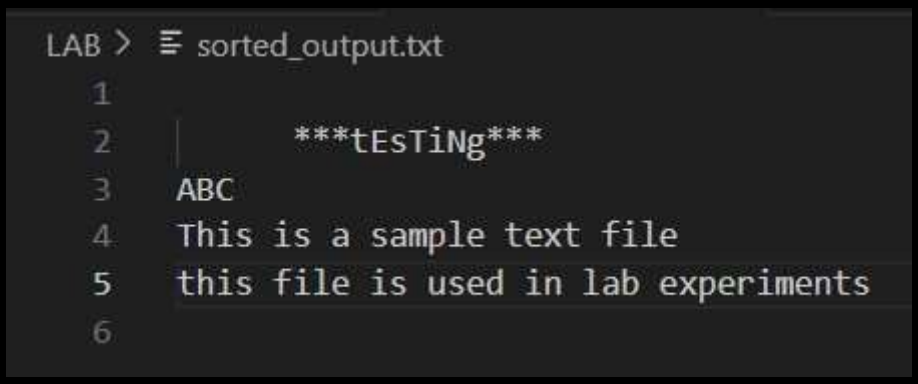
```
import java.io.*;
import java.util.*;

public class sortTextFile {
    public static void main(String[] args) {
        List<String> lines = new ArrayList<>();

        // Read lines from file
        try (BufferedReader reader = new BufferedReader(new FileReader("D:\\docs ND
stuff\\2nd yr coding stuff\\JAVA lab\\LAB\\input.txt"))) {
            String line;
            while ((line = reader.readLine()) != null) {
                lines.add(line);
            }
        } catch (IOException e) {
            System.out.println("Error reading file: " + e.getMessage());
        }

        // Sort lines
        Collections.sort(lines);




        // Write sorted lines to new file
        try (BufferedWriter writer = new BufferedWriter(new FileWriter("D:\\docs ND
stuff\\2nd yr coding stuff\\JAVA lab\\LAB\\sorted_output.txt"))) {
            for (String line : lines) {
                writer.write(line);
                writer.newLine();
            }
        } catch (IOException e) {
            System.out.println("Error writing to file: " + e.getMessage());
        }
    }
}
```

## OUTPUT:

### Unsorted File




### Sorted File



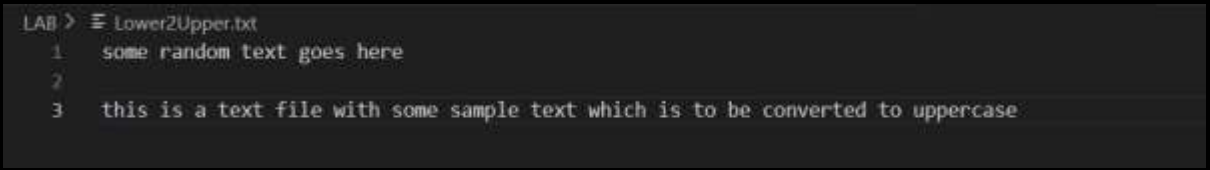

## LEARNING OUTCOMES:

- Understanding the importance of sorting algorithms in organizing data.
- Learning how to read lines from a file and store them in a data structure (e.g., an ArrayList).
- Gaining experience in using sorting techniques (e.g., Collections.sort()) to arrange data in a specific order (e.g., alphabetically or numerically).
- Understanding the significance of proper file handling (opening, reading, writing, and closing files) in Java.
- Applying problem-solving skills to create a program that efficiently sorts and writes the content back to the file.

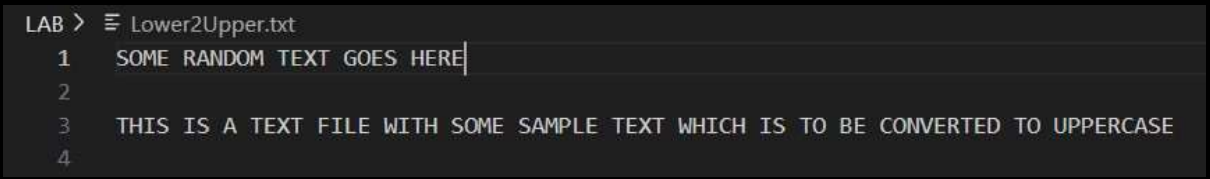
```
import java.io.*;
import java.util.*;

public class fileTextUppercase {

    public static void main(String[] args) {
        List<String> lines = new ArrayList<>();

        // Read lines from file
        try (BufferedReader reader = new BufferedReader(new FileReader("D:\\docs ND
stuff\\2nd yr coding stuff\\JAVA lab\\LAB\\Lower2Upper.txt"))) {
            String line;
            while ((line = reader.readLine()) != null) {
                lines.add(line.toUpperCase());
            }
        } catch (IOException e) {
            System.out.println("Error reading file: " + e.getMessage());
        }


// Write uppercase lines back to the same file
        try (BufferedWriter writer = new BufferedWriter(new FileWriter("D:\\docs ND
stuff\\2nd yr coding stuff\\JAVA lab\\LAB\\Lower2Upper.txt"))) {
            for (String line : lines) {
                writer.write(line);
                writer.newLine();
            }
        } catch (IOException e) {
            System.out.println("Error writing to file: " + e.getMessage());
        }
    }
}
```

## OUTPUT:

### Lowercase File




### Uppercase File



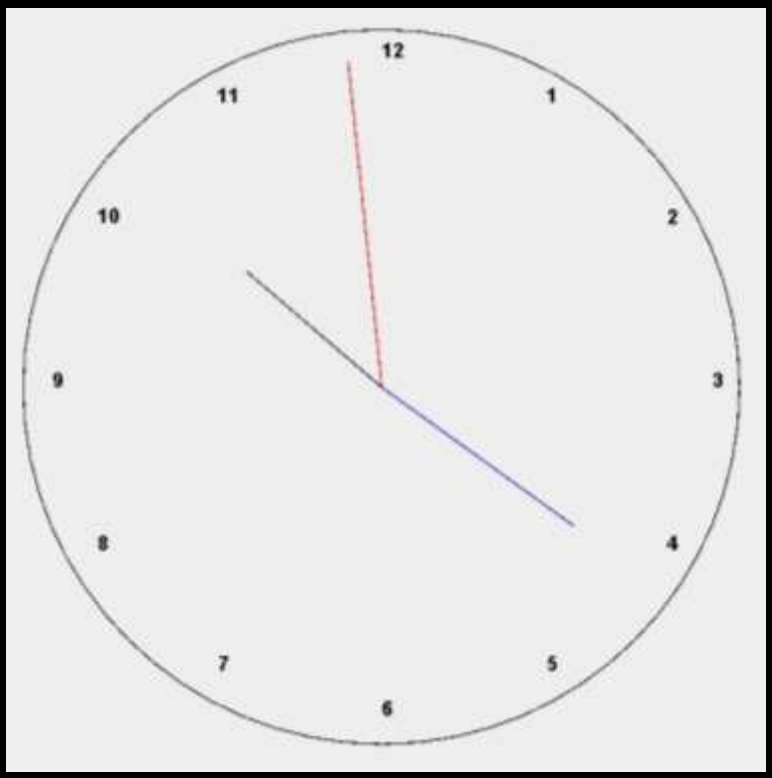

## LEARNING OUTCOMES:

- Understanding the usage of the toUpperCase() method in Java to transform all characters of a string to uppercase.
- Learning how to read and write file content using Java I/O classes (e.g., FileReader, BufferedReader, FileWriter, and BufferedWriter).
- Gaining practical knowledge of file manipulation by converting lowercase text to uppercase.

```
import java.awt.*;
import java.util.*;
import javax.swing.Timer;
import javax.swing.*;

public class AnalogClock extends JPanel {
    private int hours = 0, minutes = 0, seconds = 0;

    public AnalogClock() {
        Timer timer = new Timer(1000, e -> repaint());
        timer.start();
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        Calendar now = Calendar.getInstance();
        hours = now.get(Calendar.HOUR_OF_DAY);
        minutes = now.get(Calendar.MINUTE);
        seconds = now.get(Calendar.SECOND);

        int clockRadius = Math.min(getWidth(), getHeight()) / 3;
        int clockX = getWidth() / 2;
        int clockY = getHeight() / 2;

        // Draw the clock face
        g.setColor(Color.BLACK);
        g.drawOval(clockX - clockRadius, clockY - clockRadius, 2 * clockRadius, 2 *
clockRadius);

        // Draw the numbers
        g.setFont(new Font("default", Font.BOLD, 14));
        for (int i = 1; i <= 12; i++) {
            double angle = Math.toRadians(90 - (i * 30));
            int numberX = clockX + (int) ((clockRadius - 20) * Math.cos(angle));
            int numberY = clockY - (int) ((clockRadius - 20) * Math.sin(angle));
            g.drawString(Integer.toString(i), numberX, numberY);
        }

        // Draw the hour hand
        double hourAngle = Math.toRadians(90 - (hours * 30 + minutes * 0.5));
```

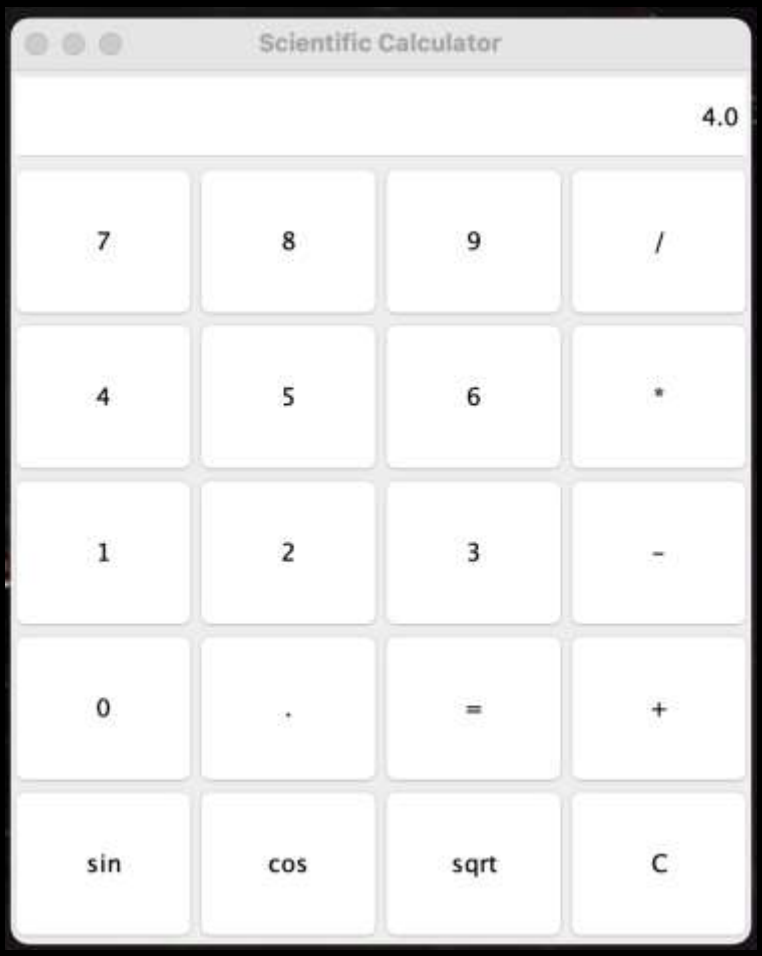
```
        int hourHandLength = clockRadius / 2;
        g.setColor(Color.BLACK);
        g.drawLine(clockX, clockY,
                clockX + (int) (hourHandLength * Math.cos(hourAngle)),
                clockY - (int) (hourHandLength * Math.sin(hourAngle)));

        // Draw the minute hand
        double minuteAngle = Math.toRadians(90 - (minutes * 6));
        int minuteHandLength = clockRadius * 2 / 3;
        g.setColor(Color.BLUE);
        g.drawLine(clockX, clockY,
                clockX + (int) (minuteHandLength * Math.cos(minuteAngle)),
                clockY - (int) (minuteHandLength * Math.sin(minuteAngle)));

        // Draw the second hand
        double secondAngle = Math.toRadians(90 - (seconds * 6));
        int secondHandLength = clockRadius - 20;
        g.setColor(Color.RED);
        g.drawLine(clockX, clockY,
                clockX + (int) (secondHandLength * Math.cos(secondAngle)),
                clockY - (int) (secondHandLength * Math.sin(secondAngle)));
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 800);
        frame.add(new AnalogClock());
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    }
}
```

**OUTPUT:**




**LEARNING OUTCOMES:**

**OUTPUT:**




## LEARNING OUTCOMES:

- Understanding Event-Driven Java Programming: Passive comprehension of event-driven programming in Java Swing, enabling effective handling of user interactions like button clicks.

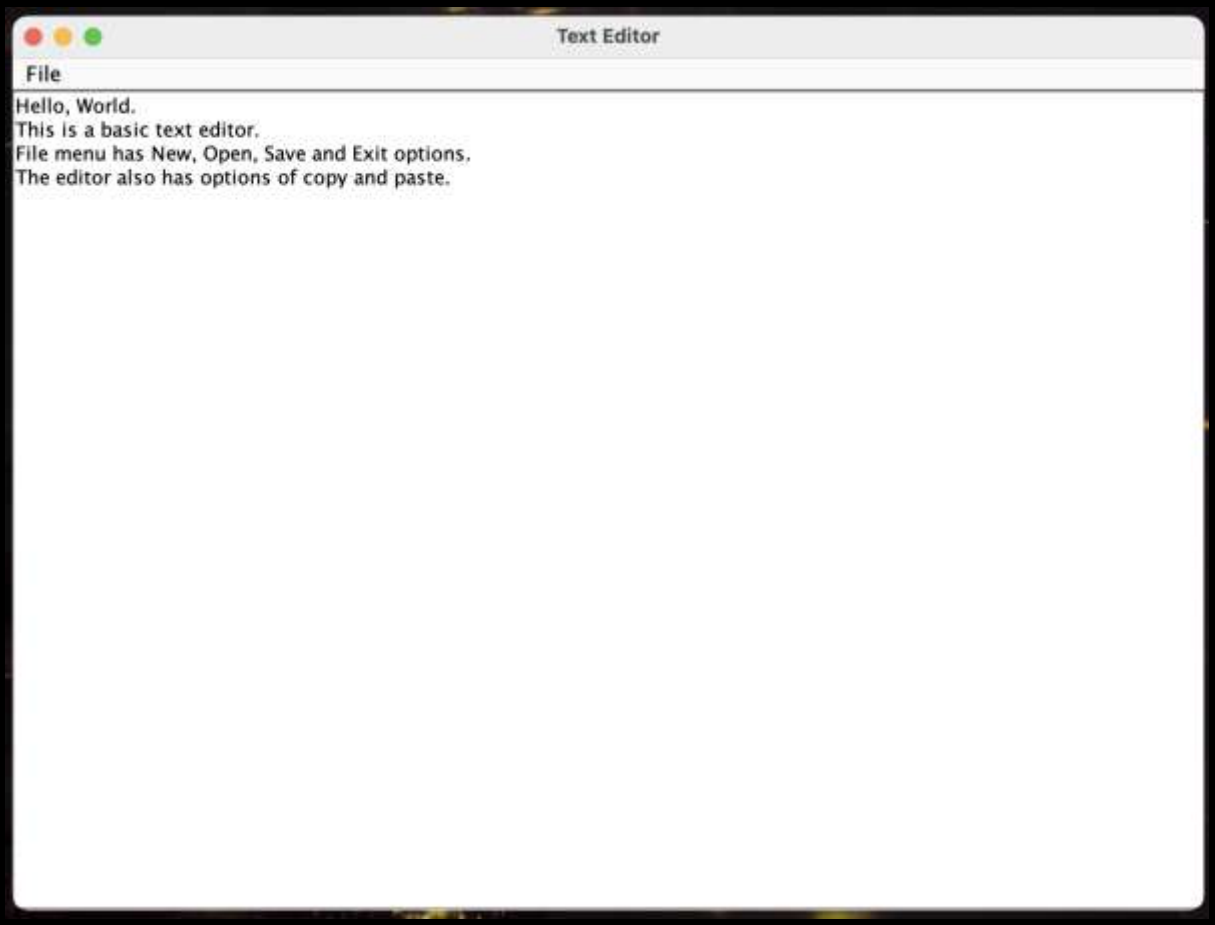
- Proficiency in GUI Development: Competence in designing and building GUIs using Java Swing components such as JFrame, JTextField, and JButton, fostering creation of visually appealing and functional interfaces.

- Data Handling and Manipulation Mastery: Ability to handle and manipulate data in Java for accurate calculations and dynamic results.

- Effective Error Handling Implementation: Application of techniques for error handling and input validation to ensure stability and reliability.

- UX Design Principles Understanding: Insight into UX design principles for creating user-friendly applications.

## OUTPUT:




## LEARNING OUTCOMES:

- Swing Component Understanding Achieved: Comprehension of Swing components like JTextArea, JMenuBar, JMenu, and JMenuItem for building interactive GUIs in Java applications is attained.

- File Handling Proficiency Acquired: Proficiency in file operations such as reading from and writing to files using BufferedReader, FileReader, BufferedWriter, and FileWriter classes is achieved.

- Event Handling Techniques Demonstrated: The code showcases event handling techniques using ActionListener interfaces to respond to user interactions with menu items, providing insights into managing event-driven programming in Java.

- UI Design Principles Understood: Designing a basic text editor interface imparts understanding of UI design principles like layout management and menu creation to enhance usability and functionality.