

Name: Vaishali Ramesh Kale

Email ID: kalevaishalir16@gmail.com

Day 20:

Task 1: Java IO Basics Write a program that reads a text file and counts the frequency of each word using FileReader and FileWriter.

Solution:::

1.Step-by-step Explanation:

- Reading the File: Use FileReader to read the content of a text file.
- Counting Word Frequency: Use a HashMap to store the frequency of each word.
- Writing the Result: Use FileWriter to write the word frequencies to an output file.

2.Instructions to Run: Create the Input File:

- Create a text file named input.txt and place it in the same directory as the Java program, or specify the correct path in the inputFile variable.
- Compile the Program: Open a terminal or command prompt, navigate to the directory containing the Java file, and compile the program using javac WordFrequencyCounter.java.
- Run the Program: Execute the compiled program using java WordFrequencyCounter.
- Check the Output: The program will generate an output.txt file in the same directory, containing the word frequencies.

CODE:::

```
import java.io.BufferedReader;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;


public class WordFrequencyCounter {


    public static void main(String[] args) {

        String inputFile = "input.txt"; // Input file path
```

```
String outputFile = "output.txt"; // Output file path

// Create a HashMap to store word frequencies
Map<String, Integer> wordCountMap = new HashMap<>();

try (BufferedReader reader = new BufferedReader(new FileReader(inputFile))) {
    String line;
    while ((line = reader.readLine()) != null) {
        String[] words = line.split("\\s+"); // Split line into words
        for (String word : words) {
            word = word.toLowerCase().replaceAll("[^a-zA-Z]", ""); // Normalize the word
            if (word.isEmpty()) {
                continue;
            }
            wordCountMap.put(word, wordCountMap.getOrDefault(word, 0) + 1);
        }
    }
} catch (IOException e) {
    e.printStackTrace();
}

// Write the word frequencies to an output file
try (FileWriter writer = new FileWriter(outputFile)) {
    for (Map.Entry<String, Integer> entry : wordCountMap.entrySet()) {
        writer.write(entry.getKey() + ": " + entry.getValue() + "\n");
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

```
System.out.println("Word frequency count has been written to " + outputFile);  
}  
}
```

OUTPUT:::

Input.txt file

```
1 Hello friends, welcome to wipro training.  
2 welcome to java programming.  
3
```

Output.txt file

```
1 java: 1  
2 wipro: 1  
3 training: 1  
4 hello: 1  
5 to: 2  
6 welcome: 2  
7 friends: 1  
8 programming: 1  
9
```

Word frequency count has been written to output.txt

...Program finished with exit code 0
Press ENTER to exit console.

Task 2: Serialization and Deserialization Serialize a custom object to a file and then deserialize it back to recover the object state.

Solution:::

Serialization is the process of converting an object's state into a byte stream so that it can be easily saved to a file or transmitted over a network. Deserialization is the reverse process of converting a byte stream back into a copy of the original object.

Step-by-Step Implementation

- Define a Custom Object: Create a class that implements Serializable.

- Serialize the Object: Write the object's state to a file using ObjectOutputStream.
- Deserialize the Object: Read the object's state from a file using ObjectInputStream.

CODE:::

```
import java.io.*;

// Define the custom object class
class Person implements Serializable {
    private static final long serialVersionUID = 1L;
    private String name;
    private int age;

    // Constructor
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    // Getters
    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    // toString method for displaying object information
    @Override
```

```
public String toString() {  
    return "Person{name='" + name + "', age=" + age + "}";  
}  
}
```

```
public class SerializeDeserializeExample {  
    public static void main(String[] args) {  
        String filename = "person.ser";  
  
        // Create a new Person object  
        Person person = new Person("Alice", 30);  
  
        // Serialize the object  
        try (FileOutputStream fileOut = new FileOutputStream(filename);  
            ObjectOutputStream out = new ObjectOutputStream(fileOut)) {  
            out.writeObject(person);  
            System.out.println("Serialized data is saved in " + filename);  
        } catch (IOException i) {  
            i.printStackTrace();  
        }  
  
        // Deserialize the object  
        Person deserializedPerson = null;  
        try (FileInputStream fileIn = new FileInputStream(filename);  
            ObjectInputStream in = new ObjectInputStream(fileIn)) {  
            deserializedPerson = (Person) in.readObject();  
        } catch (IOException i) {  
            i.printStackTrace();  
        } catch (ClassNotFoundException c) {
```

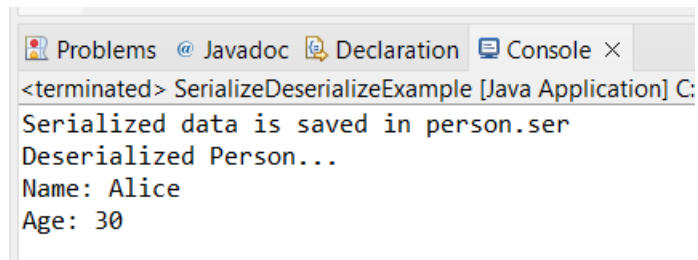
```

System.out.println("Person class not found");
c.printStackTrace();
}

// Display the deserialized object
System.out.println("Deserialized Person...");
System.out.println("Name: " + deserializedPerson.getName());
System.out.println("Age: " + deserializedPerson.getAge());
}
}

```

OUTPUT:::



```

<terminated> SerializeDeserializeExample [Java Application] C:
Serialized data is saved in person.ser
Deserialized Person...
Name: Alice
Age: 30

```

```

*****
*****

```

Task 3: New IO (NIO) Use NIO Channels and Buffers to read content from a file and write to another file.

Solution:::

Reading from a File: Use FileChannel and ByteBuffer to read data from a file.

Writing to a File: Use FileChannel and ByteBuffer to write data to a file.

CODE:::

```

package wipro;

import java.io.IOException;

import java.nio.ByteBuffer;

import java.nio.channels.FileChannel;

import java.nio.file.Path;

import java.nio.file.StandardOpenOption;

```

```

public class NIOExample {
    public static void main(String[] args) {
        Path inputPath = Path.of("input.txt"); // Input file path
        Path outputPath = Path.of("output.txt"); // Output file path

        // Use try-with-resources to ensure channels are closed after use
        try (FileChannel inputChannel = FileChannel.open(inputPath, StandardOpenOption.READ);
            FileChannel outputChannel = FileChannel.open(outputPath, StandardOpenOption.CREATE,
                StandardOpenOption.WRITE)) {

            ByteBuffer buffer = ByteBuffer.allocate(1024); // Allocate a buffer

            // Read data from the input file into the buffer
            while (inputChannel.read(buffer) > 0) {
                buffer.flip(); // Switch buffer from writing mode to reading mode
                outputChannel.write(buffer); // Write buffer content to output file
                buffer.clear(); // Clear buffer for the next read
            }

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

        System.out.println("File content has been copied from " + inputPath + " to " + outputPath);
    }
}

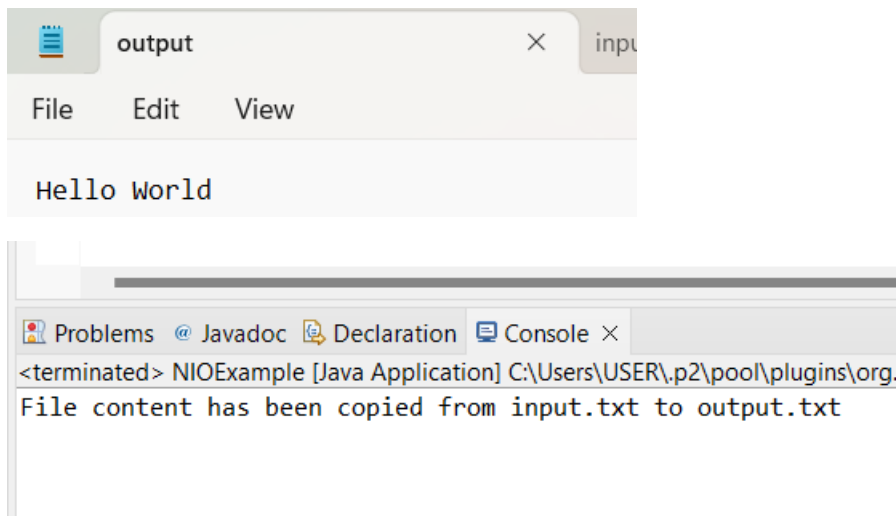
```

OUTPUT:::

Input.txt file



Output.txt file



Task 4: Java Networking Write a simple HTTP client that connects to a URL, sends a request, and displays the response headers and body.

Solution:::

- The `HttpURLConnection` class is used to send an HTTP request and receive an HTTP response from a specified URL.
- The response headers are retrieved using `getHeaderFields()` which returns a `Map<String, List<String>>` of header fields and their values.
- The response body is read using a `BufferedReader` wrapped around the `InputStream` obtained from `connection.getInputStream()`.
- For simplicity, the URL used in this example is `http://example.com`. Replace it with any valid URL to test with different endpoints.
- Proper exception handling is included to handle any `IOException` that might occur during the HTTP request/response process.

CODE:::

```
package wipro;
```



```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.List;
import java.util.Map;

public class SimpleHttpClient {
    public static void main(String[] args) {
        String urlString = "http://example.com"; // URL to connect to

        try {
            // Create a URL object
            URL url = new URL(urlString);

            // Open a connection to the URL
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();

            // Set the request method (GET by default)
            connection.setRequestMethod("GET");

            // Get the response code
            int responseCode = connection.getResponseCode();
            System.out.println("Response Code: " + responseCode);

            // Get response headers
            System.out.println("Response Headers:");
            for (Map.Entry<String, List<String>> header : connection.getHeaderFields().entrySet()) {
```

```
System.out.println(header.getKey() + ": " + header.getValue());  
}
```

```
// Read the response body
```

```
BufferedReader in = new BufferedReader(new  
InputStreamReader(connection.getInputStream()));
```

```
String inputLine;
```

```
StringBuilder responseBody = new StringBuilder();
```

```
while ((inputLine = in.readLine()) != null) {  
    responseBody.append(inputLine);  
}
```

```
in.close();
```

```
// Print the response body
```

```
System.out.println("Response Body:");
```

```
System.out.println(responseBody.toString());
```

```
} catch (IOException e) {
```

```
    e.printStackTrace();
```

```
}
```

```
}
```

```
}
```

OUTPUT:::

```
Problems Javadoc Declaration Console X
<terminated> SimpleHttpClient [Java Application] C:\Users\USER\p2\pool\plugins\org.eclipse.jst.j2ee.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\java
Response Code: 200
Response Headers:
null: [HTTP/1.1 200 OK]
X-Cache: [HIT]
Cache-Control: [max-age=604800]
Etag: ["3147526947+ident"]
Server: [ECAcc (dcd/7D25)]
Vary: [Accept-Encoding]
Expires: [Sun, 09 Jun 2024 13:24:54 GMT]
Last-Modified: [Thu, 17 Oct 2019 07:18:26 GMT]
Content-Length: [1256]
Age: [240144]
Date: [Sun, 02 Jun 2024 13:24:54 GMT]
Content-Type: [text/html; charset=UTF-8]
Response Body:
<!doctype html><html><head> <title>Example Domain</title> <meta charset="utf-8" /> <meta http-equiv="Content-type" cont
```


Task 5: Java Networking and Serialization Develop a basic TCP client and server application where the client sends a serialized object with 2 numbers and operation to be performed on them to the server, and the server computes the result and sends it back to the client. for eg, we could send 2, 2, "+" which would mean 2 + 2 .

Solution:::

Step-by-step Explanation

1. Define a Custom Serializable Object: Create a class that implements Serializable to encapsulate the two numbers and the operation.
2. Develop the Server: Create a server application that listens for incoming connections, deserializes the received object, performs the operation, and sends the result back.
3. Develop the Client: Create a client application that connects to the server, sends the serialized object, and receives and displays the result.

CODE:::

CODE for operation request:

```
import java.io.Serializable;

public class OperationRequest implements Serializable {

    private static final long serialVersionUID = 1L;

    private int number1;
```

```
private int number2;

private String operation;

public OperationRequest(int number1, int number2, String operation) {

this.number1 = number1;

this.number2 = number2;

this.operation = operation;

}

public int getNumber1() {

return number1;

}

public int getNumber2() {

return number2;

}

public String getOperation() {

return operation;

}

}
```

TCP Sever

```
import java.io.*;

import java.net.ServerSocket;

import java.net.Socket;


public class TCPServer {

public static void main(String[] args) {

int port = 12345;
```

```
try (ServerSocket serverSocket = new ServerSocket(port)) {  
    System.out.println("Server is listening on port " + port);  
  
    while (true) {  
        try (Socket socket = serverSocket.accept());  
        ObjectInputStream objectInputStream = new ObjectInputStream(socket.getInputStream());  
        ObjectOutputStream objectOutputStream = new  
        ObjectOutputStream(socket.getOutputStream()) {  
  
            // Read the request object  
            OperationRequest request = (OperationRequest) objectInputStream.readObject();  
            int number1 = request.getNumber1();  
            int number2 = request.getNumber2();  
            String operation = request.getOperation();  
            int result = 0;  
  
            // Perform the requested operation  
            switch (operation) {  
                case "+":  
                    result = number1 + number2;  
                    break;  
                case "-":  
                    result = number1 - number2;  
                    break;  
                case "*":  
                    result = number1 * number2;  
                    break;  
                case "/":  
                    if (number2 != 0) {
```

```

result = number1 / number2;
} else {
    objectOutputStream.writeObject("Error: Division by zero");
    continue;
}
break;
default:
    objectOutputStream.writeObject("Error: Invalid operation");
    continue;
}

// Send the result back to the client
objectOutputStream.writeObject("Result: " + result);
} catch (ClassNotFoundException e) {
    e.printStackTrace();
}
}
} catch (IOException e) {
    e.printStackTrace();
}
}
}
}

```

TCP Client::

```

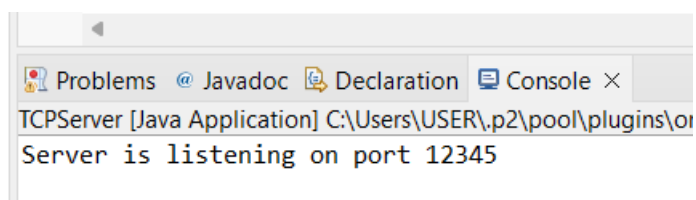
import java.io.*;
import java.net.Socket;

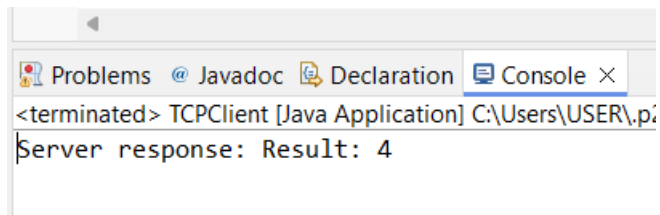
public class TCPClient {
    public static void main(String[] args) {

```

```
String hostname = "localhost";  
  
int port = 12345;  
  
try (Socket socket = new Socket(hostname, port);  
  
    ObjectOutputStream objectOutputStream = new  
        ObjectOutputStream(socket.getOutputStream());  
  
    ObjectInputStream objectInputStream = new ObjectInputStream(socket.getInputStream())) {  
  
    // Create a request object with numbers and an operation  
    OperationRequest request = new OperationRequest(2, 2, "+");  
  
    // Send the request object to the server  
    objectOutputStream.writeObject(request);  
  
    // Read the response from the server  
    String response = (String) objectInputStream.readObject();  
    System.out.println("Server response: " + response);  
  
} catch (IOException | ClassNotFoundException e) {  
    e.printStackTrace();  
}  
  
}
```

OUTPUT:::





```
<terminated> TCPClient [Java Application] C:\Users\USER\p;  
Server response: Result: 4
```


Task 6: Java 8 Date and Time API Write a program that calculates the number of days between two dates input by the user.

Solution:::

Explanation:

- LocalDate: Used to represent a date without a time-zone in the ISO-8601 calendar system (default).
- DateTimeFormatter: Used to parse and format date-time objects.
- ChronoUnit.DAYS.between: Calculates the number of days between two dates.
- Scanner: Used to capture user input from the console.
- program prompts the user to input two dates, parses these dates using LocalDate.parse, calculates the difference in days using ChronoUnit.DAYS.between, and then prints the result.

CODE:::

```
package wipro;  
  
import java.time.LocalDate;  
import java.time.format.DateTimeFormatter;  
import java.time.temporal.ChronoUnit;  
import java.util.Scanner;  
  
public class DaysBetweenDates {  
    public static void main(String[] args) {  
        // Create a scanner object for user input  
        Scanner scanner = new Scanner(System.in);  
  
        // Define the date format
```



```

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");

// Prompt user for the first date
System.out.print("Enter the first date (yyyy-MM-dd): ");
String firstDateString = scanner.nextLine();

// Prompt user for the second date
System.out.print("Enter the second date (yyyy-MM-dd): ");
String secondDateString = scanner.nextLine();

// Parse the input dates
LocalDate firstDate = LocalDate.parse(firstDateString, formatter);
LocalDate secondDate = LocalDate.parse(secondDateString, formatter);

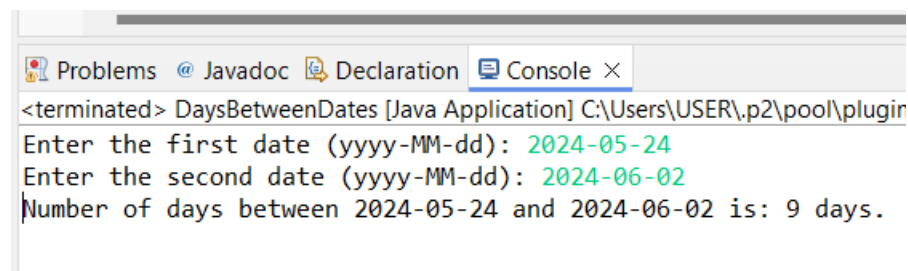
// Calculate the number of days between the two dates
long daysBetween = ChronoUnit.DAYS.between(firstDate, secondDate);

// Display the result
System.out.println("Number of days between " + firstDate + " and " + secondDate + " is: " +
daysBetween + " days.");

// Close the scanner
scanner.close();
}
}

```

OUTPUT:::



```

<terminated> DaysBetweenDates [Java Application] C:\Users\USER\p2\pool\plugin
Enter the first date (yyyy-MM-dd): 2024-05-24
Enter the second date (yyyy-MM-dd): 2024-06-02
Number of days between 2024-05-24 and 2024-06-02 is: 9 days.

```

```

*****
*****

```

Task 7: Timezone Create a timezone converter that takes a time in one timezone and converts it to another timezone.

Solution:::

Explanation:

- LocalDateTime: Represents a date-time without a time-zone in the ISO-8601 calendar system.
- ZonedDateTime: Represents a date-time with a time-zone in the ISO-8601 calendar system.
- ZoneId: Represents a time-zone identifier.
- DateTimeFormatter: Used to parse and format date-time objects.
- Scanner: Used to capture user input from the console.

CODE:::

```
import java.time.LocalDateTime;
import java.time.ZonedDateTime;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;

public class TimezoneConverter {
    public static void main(String[] args) {
        // Create a scanner object for user input
        Scanner scanner = new Scanner(System.in);

        // Define the date and time format
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");

        // Prompt user for the local date and time
        System.out.print("Enter the date and time (yyyy-MM-dd HH:mm): ");
        String localDateTimeString = scanner.nextLine();

        // Prompt user for the source timezone
```

```

System.out.print("Enter the source timezone (e.g., America/New_York): ");

String sourceTimeZone = scanner.nextLine();

// Prompt user for the target timezone
System.out.print("Enter the target timezone (e.g., Europe/London): ");
String targetTimeZone = scanner.nextLine();

// Parse the input local date and time
LocalDateTime localDateTime = LocalDateTime.parse(localDateTimeString, formatter);

// Create ZonedDateTime in the source timezone
ZonedDateTime sourceZonedDateTime = ZonedDateTime.of(localDateTime,
ZoneId.of(sourceTimeZone));

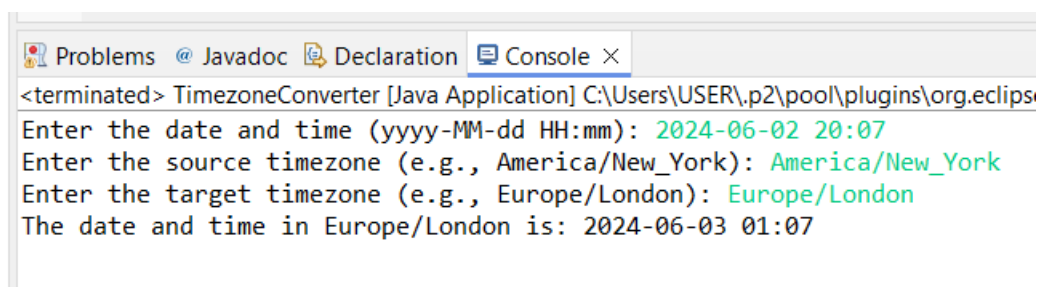
// Convert to the target timezone
ZonedDateTime targetZonedDateTime =
sourceZonedDateTime.withZoneSameInstant(ZoneId.of(targetTimeZone));

// Display the result
System.out.println("The date and time in " + targetTimeZone + " is: " +
targetZonedDateTime.format(formatter));

// Close the scanner
scanner.close();
}
}

```

OUTPUT:.....



The screenshot shows the Eclipse IDE's console window. The title bar includes tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The console output is as follows:

```

<terminated> TimezoneConverter [Java Application] C:\Users\USER\p2\pool\plugins\org.eclipse
Enter the date and time (yyyy-MM-dd HH:mm): 2024-06-02 20:07
Enter the source timezone (e.g., America/New_York): America/New_York
Enter the target timezone (e.g., Europe/London): Europe/London
The date and time in Europe/London is: 2024-06-03 01:07

```