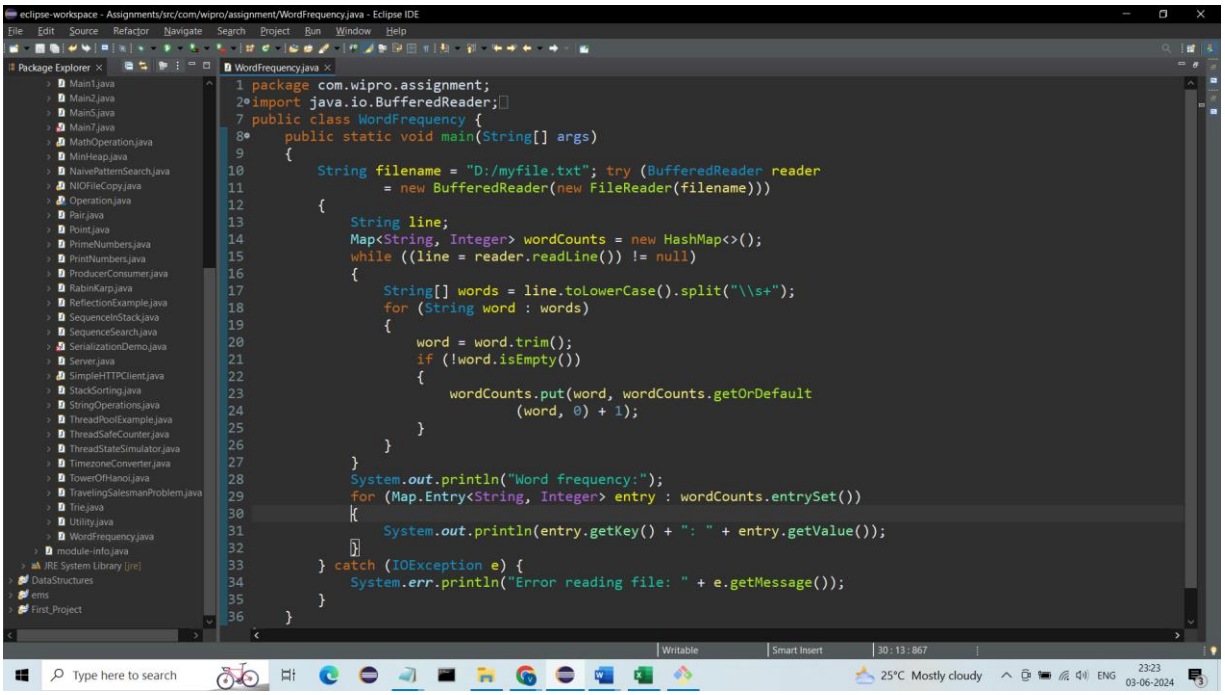
Task 1: Java IO Basics

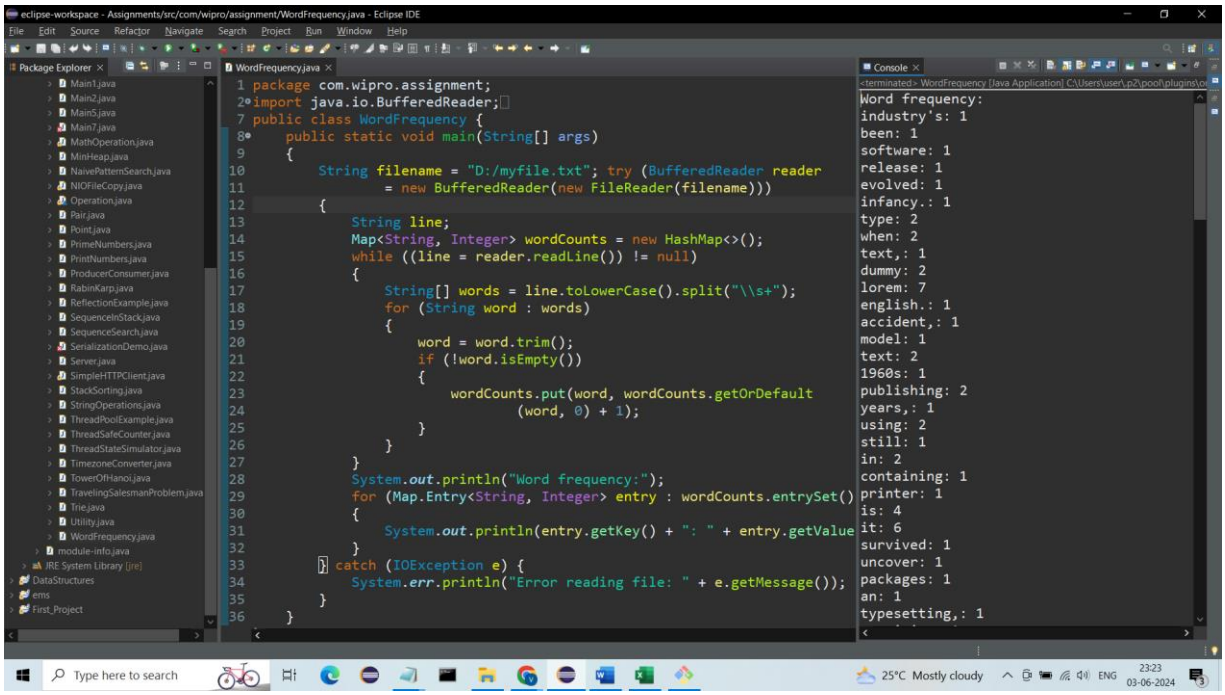
Write a program that reads a text file and counts the frequency of each word using FileReader and FileWriter.

Program: -



```java
package com.wipro.assignment;
import java.io.BufferedReader;
public class WordFrequency {
    public static void main(String[] args)
    {
        String filename = "D:/myfile.txt"; try (BufferedReader reader
                = new BufferedReader(new FileReader(filename)))
        {
            String line;
            Map<String, Integer> wordCounts = new HashMap<>();
            while ((line = reader.readLine()) != null)
            {
                String[] words = line.toLowerCase().split("\\s+");
                for (String word : words)
                {
                    word = word.trim();
                    if (!word.isEmpty())
                    {
                        wordCounts.put(word, wordCounts.getOrDefault
                                (word, 0) + 1);
                    }
                }
            }
            System.out.println("Word frequency:");
            for (Map.Entry<String, Integer> entry : wordCounts.entrySet())
            {
                System.out.println(entry.getKey() + ": " + entry.getValue());
            }
        } catch (IOException e) {
            System.err.println("Error reading file: " + e.getMessage());
        }
    }
```

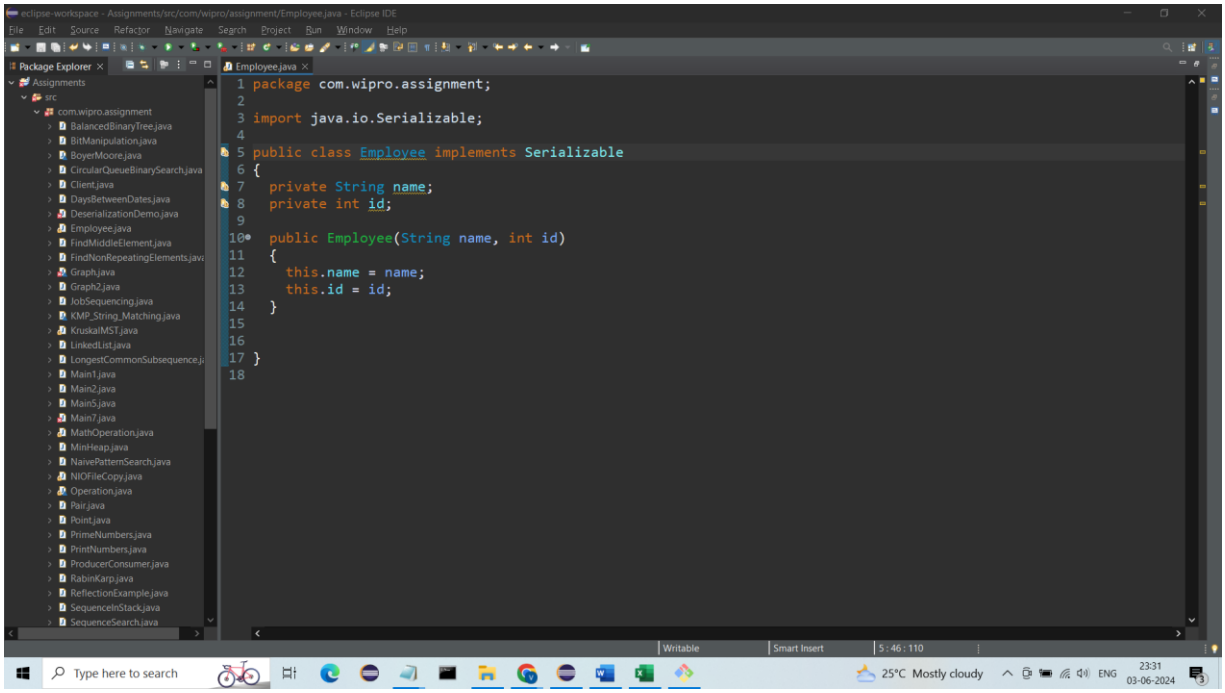
Output: -



Task 2: Serialization and Deserialization

Serialize a custom object to a file and then deserialize it back to recover the object state.

Custom Class (Employee):



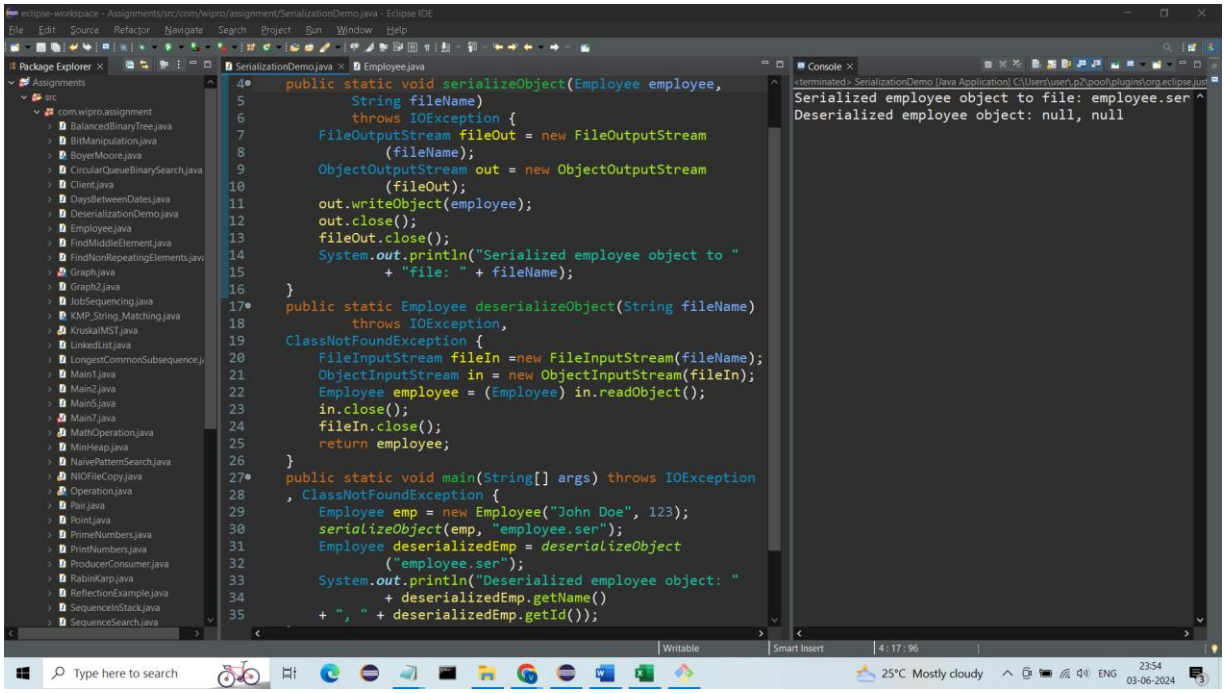
Serialization: -

```java
public static void serializeObject(Employee employee,
        String fileName)
        throws IOException {
    FileOutputStream fileOut = new FileOutputStream(fileName);
    ObjectOutputStream out = new ObjectOutputStream(fileOut);
    out.writeObject(employee);
    out.close();
    fileOut.close();
    System.out.println("Serialized employee object to file: "
    + fileName);
}
```

Deserialization: -

```java
public static Employee deserializeObject(String fileName) throws IOException,
ClassNotFoundException {
    FileInputStream fileIn = new FileInputStream(fileName);
    ObjectInputStream in = new ObjectInputStream(fileIn);
    Employee employee = (Employee) in.readObject();
    in.close();
    fileIn.close();
    return employee;
}
```

Output: -



Task 3: New IO (NIO)

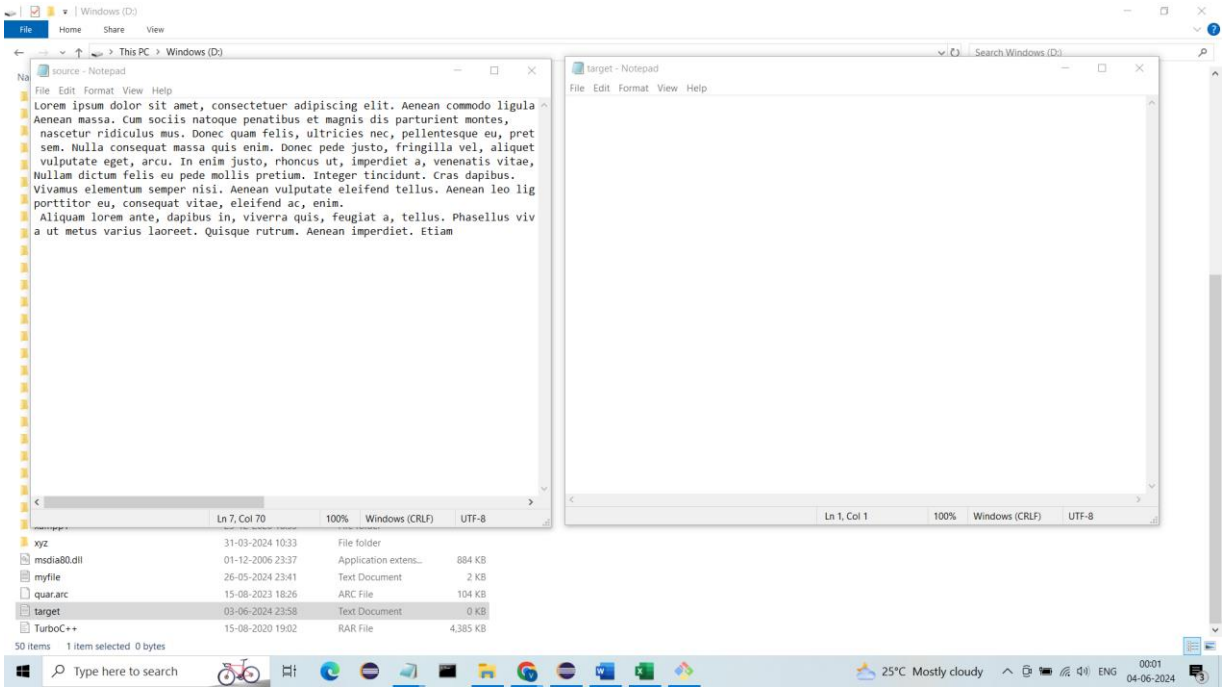
Use NIO Channels and Buffers to read content from a file and write to another file.

Program: -

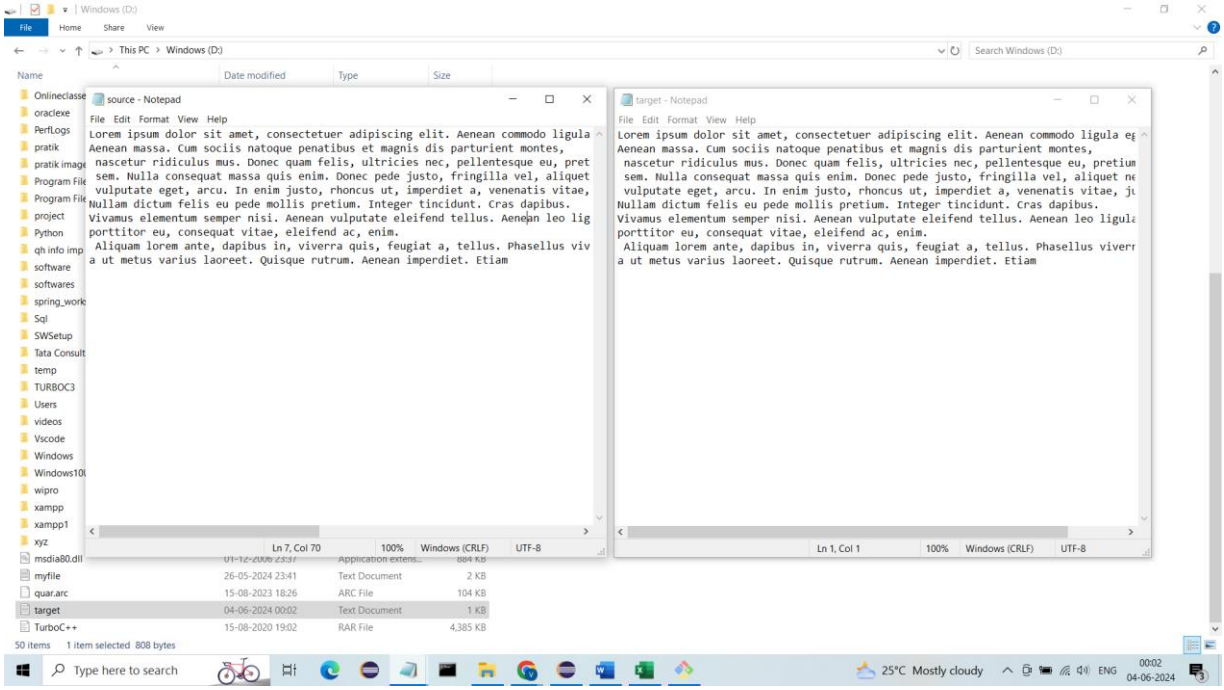
```java
package com.wipro.assignment;
import java.io.IOException;
public class NIOFileCopy {
    public static void main(String[] args) {
        Path sourcePath = Paths.get("C:/source.txt");
        Path targetPath = Paths.get("D:/target.txt");
        try (FileChannel sourceChannel = FileChannel.open(sourcePath, StandardOpenOption.READ);
                FileChannel targetChannel = FileChannel.open(targetPath, StandardOpenOption.CREATE,
                        StandardOpenOption.WRITE)) {
            ByteBuffer buffer = ByteBuffer.allocate(1024);
            while (sourceChannel.read(buffer) != -1) {
                buffer.flip();
                while (buffer.hasRemaining()) {
                    targetChannel.write(buffer);
                }
                buffer.clear();
            }
            System.out.println("File copied successfully.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output: -

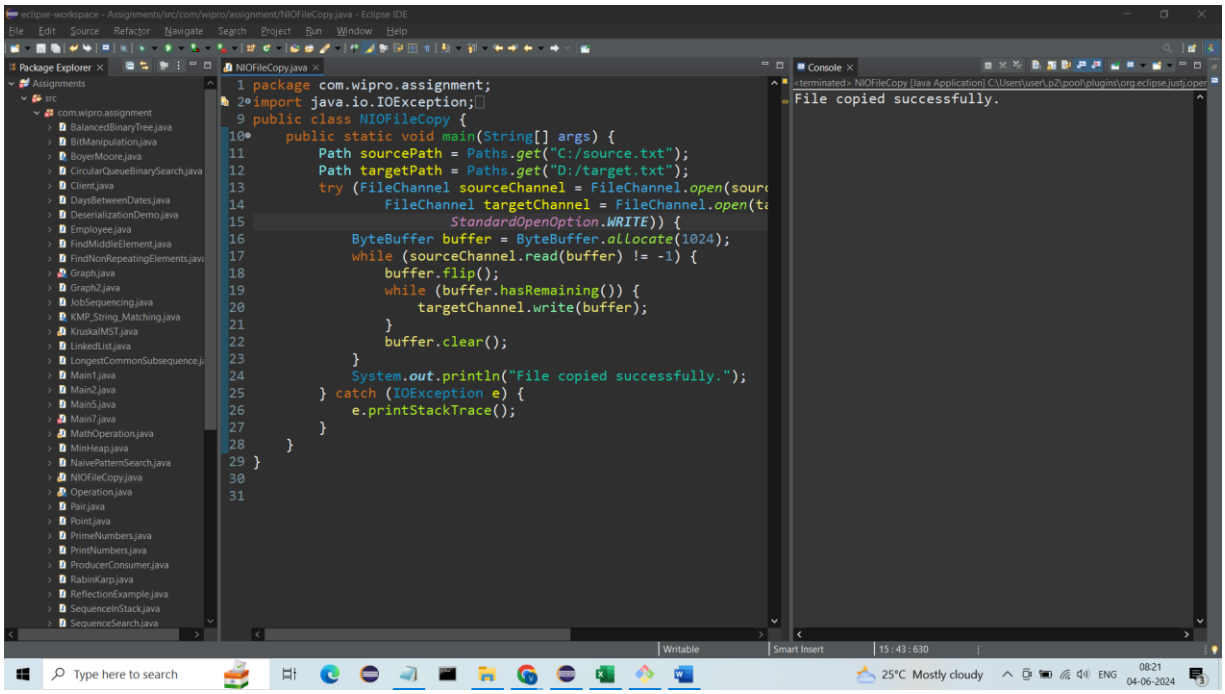
Before Copying: -



After Copying: -



Program Output: -



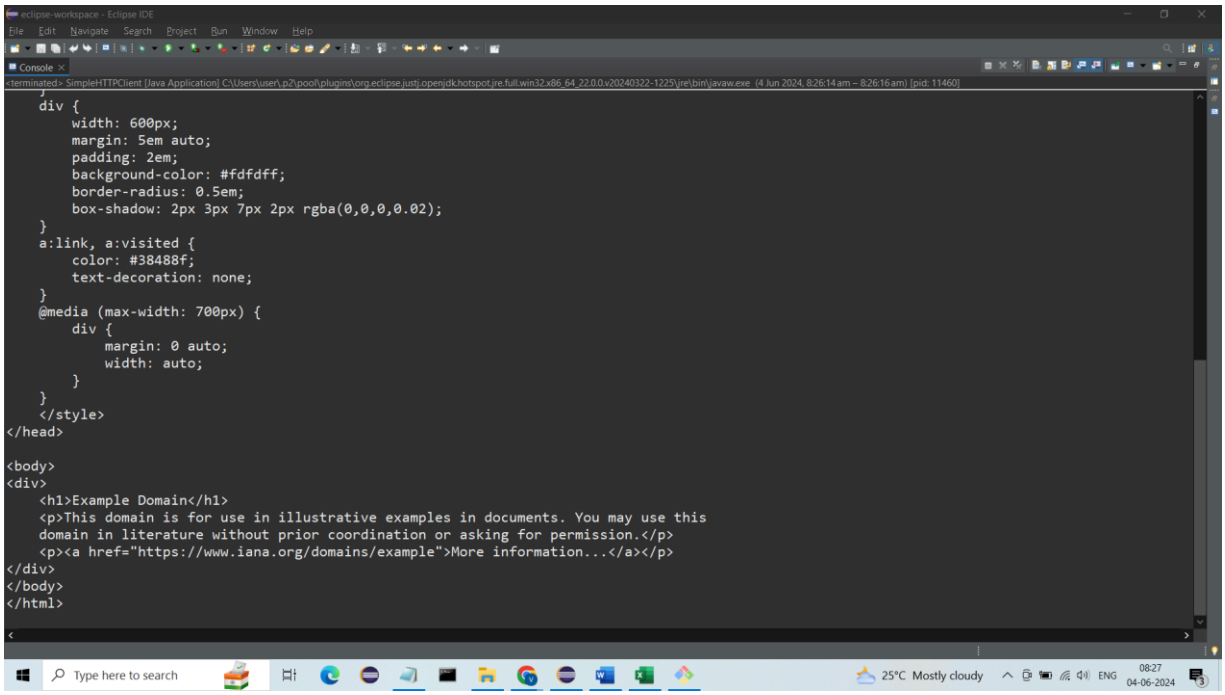
Task 4: Java Networking

Write a simple HTTP client that connects to a URL, sends a request, and displays the response headers and body.

```java
public class SimpleHTTPClient {
    public static void main(String[] args) {
        String urlString = "https://www.example.com"; try {
            URL url = new URL(urlString);
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("GET");
            int responseCode = connection.getResponseCode();
            System.out.println("Response Code: " + responseCode);
            System.out.println("Response Headers:");
            connection.getHeaderFields().forEach((key, value) -> {
                if (key != null) {
                    System.out.println(key + ": " + value);
                }
            });
            System.out.println("\nResponse Body:");
            try (BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()))) {
                String line;
                while ((line = reader.readLine()) != null) {
                    System.out.println(line);
                }
            }
            connection.disconnect();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output: -

```
Response Code: 200
Response Headers:
X-Cache: [HIT]
Server: [ECAcc (dcd/7D4F)]
Last-Modified: [Thu, 17 Oct 2019 07:18:26 GMT]
Date: [Tue, 04 Jun 2024 02:56:15 GMT]
Accept-Ranges: [bytes]
Cache-Control: [max-age=604800]
Etag: ["3147526947+gzip"]
Vary: [Accept-Encoding]
Expires: [Tue, 11 Jun 2024 02:56:15 GMT]
Content-Length: [1256]
Age: [375251]
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" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
```



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

Request class: -

Request.java × Server.java Response.java

```java
package com.wipro.assignment;

import java.io.Serializable;

@SuppressWarnings("serial")
class Request implements Serializable {
    private int num1;
    private int num2;
    private String operation;

    public Request(int num1, int num2, String operation) {
        this.num1 = num1;
        this.num2 = num2;
        this.operation = operation;
    }

    public int getNum1() {
        return num1;
    }

    public int getNum2() {
        return num2;
    }

    public String getOperation() {
        return operation;
    }

}
```

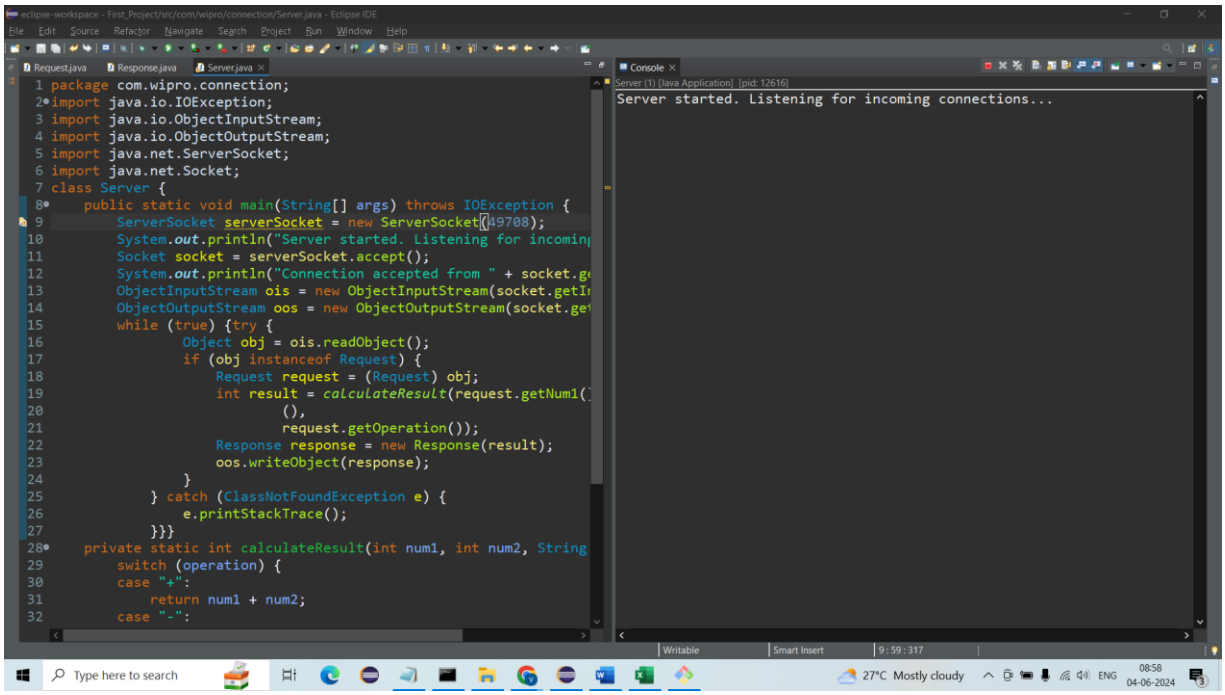
Response Class: -

```java
package com.wipro.assignment;

import java.io.Serializable;

class Response implements Serializable {
private int result;

public Response(int result) {
    this.result = result;
}

public int getResult() {
    return result;
}

}
```

Server Class: -

```java
class Server {
    public static void main(String[] args) throws IOException {
        ServerSocket serverSocket = new ServerSocket(49704);
        System.out.println("Server started. Listening for incoming connections...");
        Socket socket = serverSocket.accept();
        System.out.println("Connection accepted from " + socket.getInetAddress());
        ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());
        ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());
        while (true) {try {
                Object obj = ois.readObject();
                if (obj instanceof Request) {
                    Request request = (Request) obj;
                    int result = calculateResult(request.getNum1(), request.getNum2(), request.getOperation());
                    Response response = new Response(result);
                    oos.writeObject(response);
                }
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }}}
    private static int calculateResult(int num1, int num2, String operation) {
        switch (operation) {
        case "+":
            return num1 + num2;
        case "-":
            return num1 - num2;
        case "*":
            return num1 * num2;
        case "/":
            return num1 / num2;
        default:
            return 0;
        }
```

Output: -



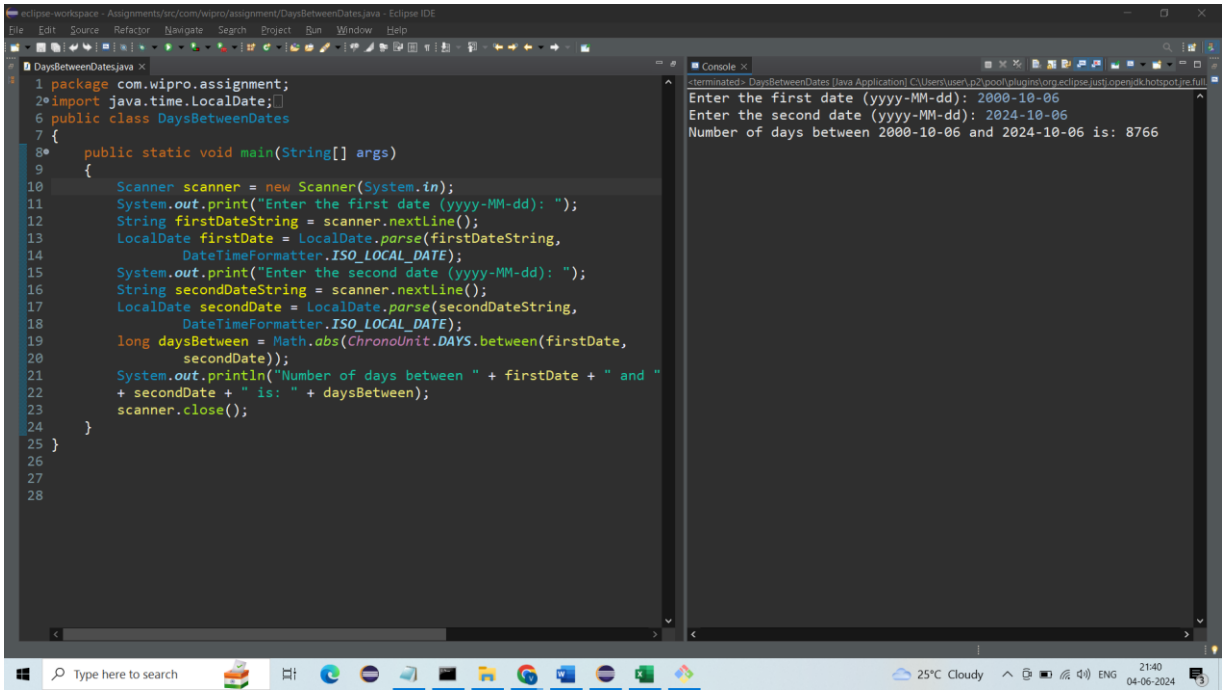
Task 6: Java 8 Date and Time API

Write a program that calculates the number of days between two dates input by the user.

Program: -

```java
DaysBetweenDates.java
package com.wipro.assignment;
import java.time.LocalDate;
public class DaysBetweenDates
{
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the first date (yyyy-MM-dd): ");
        String firstDateString = scanner.nextLine();
        LocalDate firstDate = LocalDate.parse(firstDateString,
                DateTimeFormatter.ISO_LOCAL_DATE);
        System.out.print("Enter the second date (yyyy-MM-dd): ");
        String secondDateString = scanner.nextLine();
        LocalDate secondDate = LocalDate.parse(secondDateString,
                DateTimeFormatter.ISO_LOCAL_DATE);
        long daysBetween = Math.abs(ChronoUnit.DAYS.between(firstDate,
                secondDate));
        System.out.println("Number of days between " + firstDate + " and "
        + secondDate + " is: " + daysBetween);
        scanner.close();
    }
}
```

Output: -



Task 7: Timezone

Create a timezone converter that takes a time in one timezone and converts it to another timezone.

Function: -

```
}
public static String convertTimezone(String time, String fromTimezone, String toTimezone)
{
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
    LocalDateTime localDateTime = LocalDateTime.parse(time, formatter);
    ZoneId fromZone = ZoneId.of(fromTimezone);
    ZoneId toZone = ZoneId.of(toTimezone);
    ZonedDateTime fromZonedDateTime = ZonedDateTime.of(localDateTime, fromZone);
    ZonedDateTime toZonedDateTime = fromZonedDateTime.withZoneSameInstant(toZone);
    return toZonedDateTime.format(formatter);
}
```

Output: -