Task 1: Java IO Basics

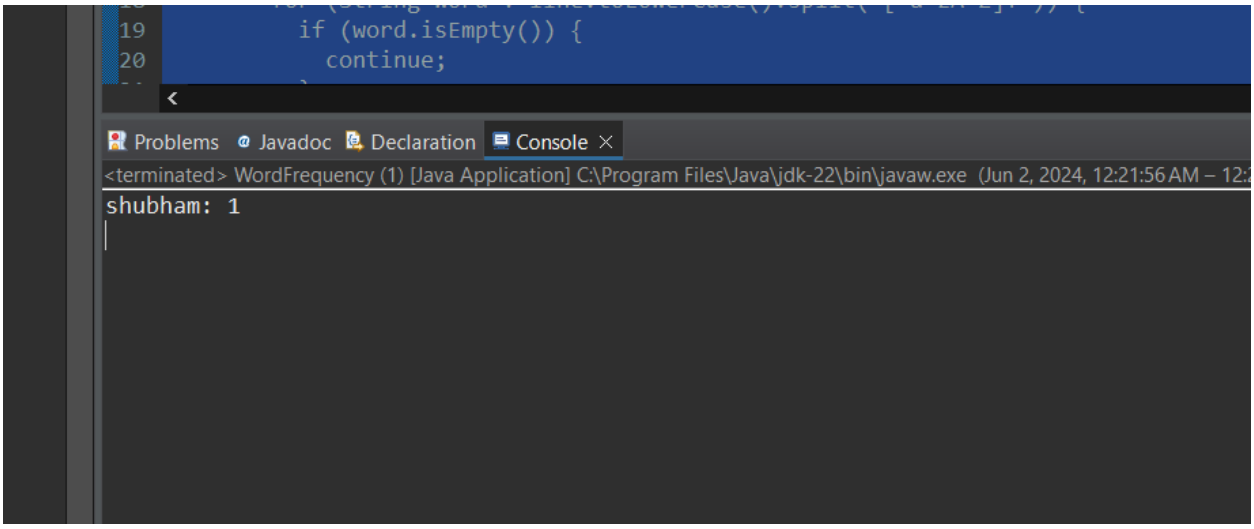
Write a program that reads a text file and counts the frequency of each word using FileReader and FileWriter.

Ans:

```
package Practice1;


import java.io.BufferedReader;

import java.io.FileReader;

import java.util.HashMap;

import java.util.Map;


public class WordFrequency {


  public static void main(String[] args) throws Exception {

    String fileName = "one.txt";

    Map<String, Integer> wordCounts = new HashMap<>();


    try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {

      String line;

      while ((line = reader.readLine()) != null) {


        for (String word : line.toLowerCase().split("[^a-zA-Z]+")) {

          if (word.isEmpty()) {

            continue;

          }

          wordCounts.put(word, wordCounts.getOrDefault(word, 0) + 1);

        }

      }

    }
```

```
    for (Map.Entry<String, Integer> entry : wordCounts.entrySet()) {
      System.out.println(entry.getKey() + ": " + entry.getValue());
    }
  }
}
```

OUTPUT:



Task 2: Serialization and Deserialization

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

Ans.

```
package Seralization;

import java.io.*;

public class SerializationDemo {

    public static void main(String[] args) {

        Person person = new Person("Shubham", 24);
```

```
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("person.dat"))) {
            oos.writeObject(person);
            System.out.println("Object serialized successfully!");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        Person deserializedPerson = null;
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream("person.dat"))) {
            while (true) {
                deserializedPerson = (Person) ois.readObject();
                System.out.println("Object deserialized successfully!");
                System.out.println(deserializedPerson);
                break;
            }
        } catch (EOFException e) {

            System.out.println("Reached end of file.");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}

class Person implements Serializable {
    private String name;
    private int age;

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


    @Override
    public String toString() {
        return "Person [name=" + name + ", age=" + age + "]";
```

```
    }
}
```

Outpt:

Task 3: New IO (NIO)

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

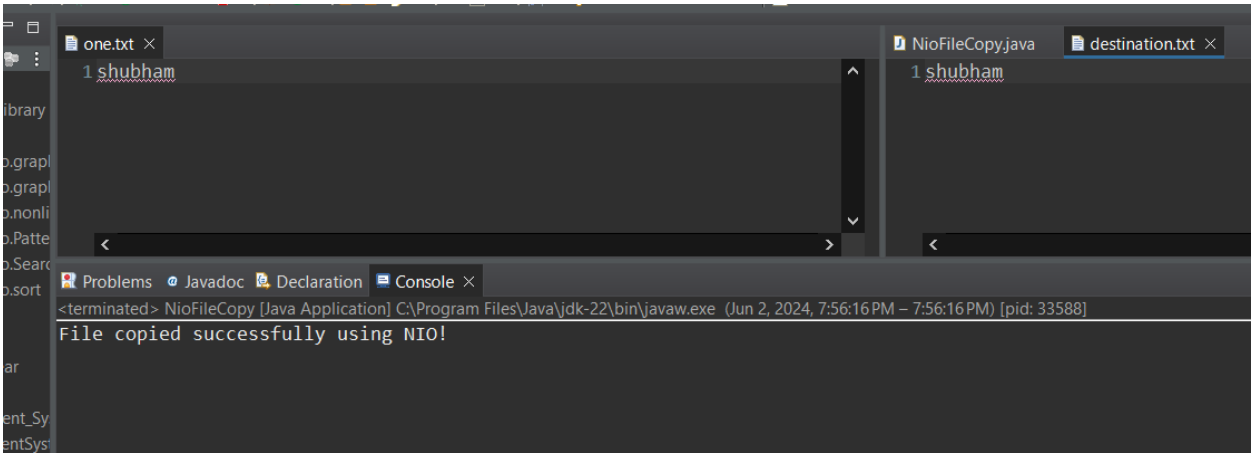
Ans.

```
package Practice1;


import java.io.IOException;

import java.nio.ByteBuffer;

import java.nio.channels.FileChannel;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.StandardOpenOption;


public class NioFileCopy {


    public static void main(String[] args) throws IOException {
```

```
    Path sourcePath = Path.of("one.txt");
    Path destinationPath = Path.of("destination.txt");

    try (FileChannel sourceChannel = (FileChannel) Files.newByteChannel(sourcePath,
StandardOpenOption.READ);
         FileChannel destinationChannel = (FileChannel) Files.newByteChannel(destinationPath,
StandardOpenOption.CREATE, StandardOpenOption.WRITE)) {

        ByteBuffer buffer = ByteBuffer.allocate(1024); // Adjust buffer size as needed

        while (true) {

            int bytesRead = sourceChannel.read(buffer);

            if (bytesRead == -1) {

                break;
            }

            buffer.flip();

            while (buffer.hasRemaining()) {
                destinationChannel.write(buffer);
            }

            buffer.clear();
        }
        System.out.println("File copied successfully using NIO!");
    }
```

```
    }
}
```

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.

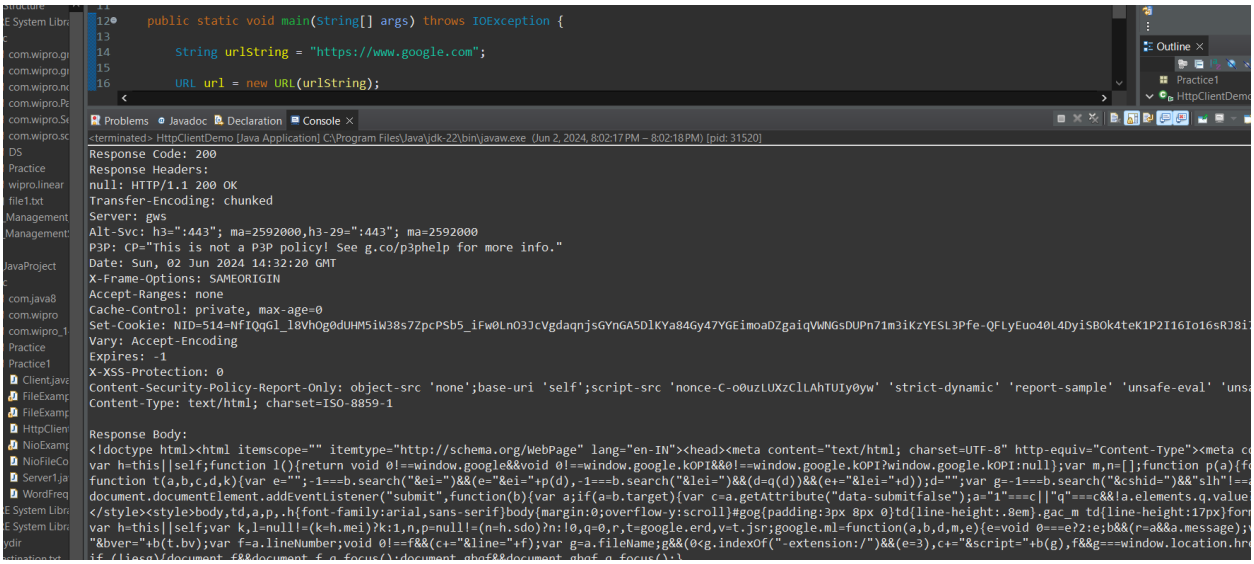
Ans:

```
package Practice1;


import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.net.HttpURLConnection;

import java.net.URL;


public class HttpClientDemo {


    public static void main(String[] args) throws IOException {
```

```
String urlString = "https://www.google.com";

URL url = new URL(urlString);

HttpURLConnection connection = (HttpURLConnection) url.openConnection();

connection.setRequestMethod("GET");

connection.connect();

int responseCode = connection.getResponseCode();

if (responseCode == HttpURLConnection.HTTP_OK) {
   System.out.println("Response Code: " + responseCode);

   System.out.println("Response Headers:");
   for (String headerName : connection.getHeaderFields().keySet()) {
      System.out.println(headerName + ": " + connection.getHeaderField(headerName));
   }

   System.out.println("\nResponse Body:");
   try (InputStream inputStream = connection.getInputStream();
        BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream))) {
      String line;
      while ((line = reader.readLine()) != null) {
         System.out.println(line);
      }
   }
} else {
```

```
            System.out.println("Error: HTTP response code " + responseCode);
        }


        connection.disconnect();
    }
}
```

Output:



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

Ans:

```
package Practice2;

import java.io.Serializable;

public class OperationRequest implements Serializable {
    private static final long serialVersionUID = 1L;


    private double number1;

    private double number2;
```

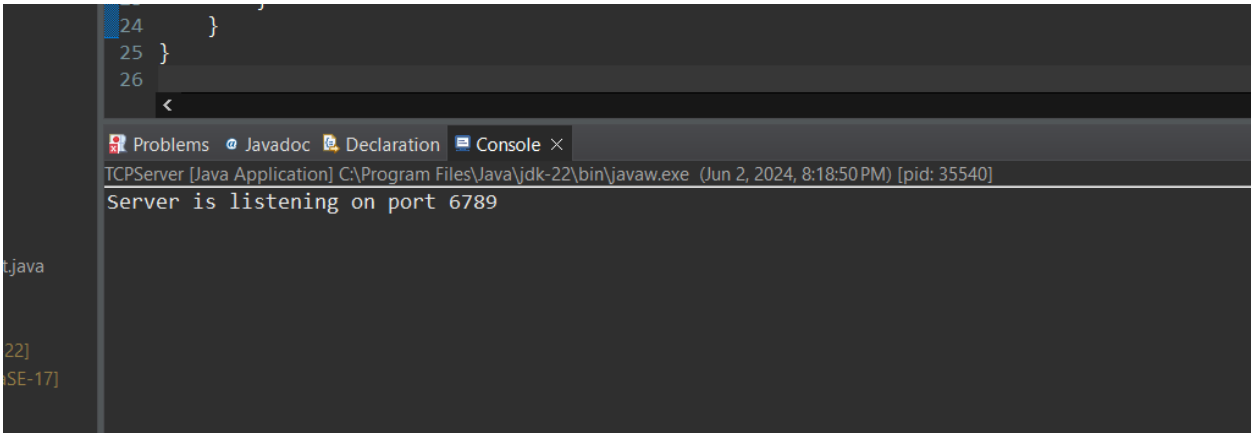
```
    private String operation;

    public OperationRequest(double number1, double number2, String operation) {
        this.number1 = number1;
        this.number2 = number2;
        this.operation = operation;
    }

    public double getNumber1() {
        return number1;
    }

    public double getNumber2() {
        return number2;
    }

    public String getOperation() {
        return operation;
    }
}

package Practice2;

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

public class TCPServer {
    public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(6789)) {
```

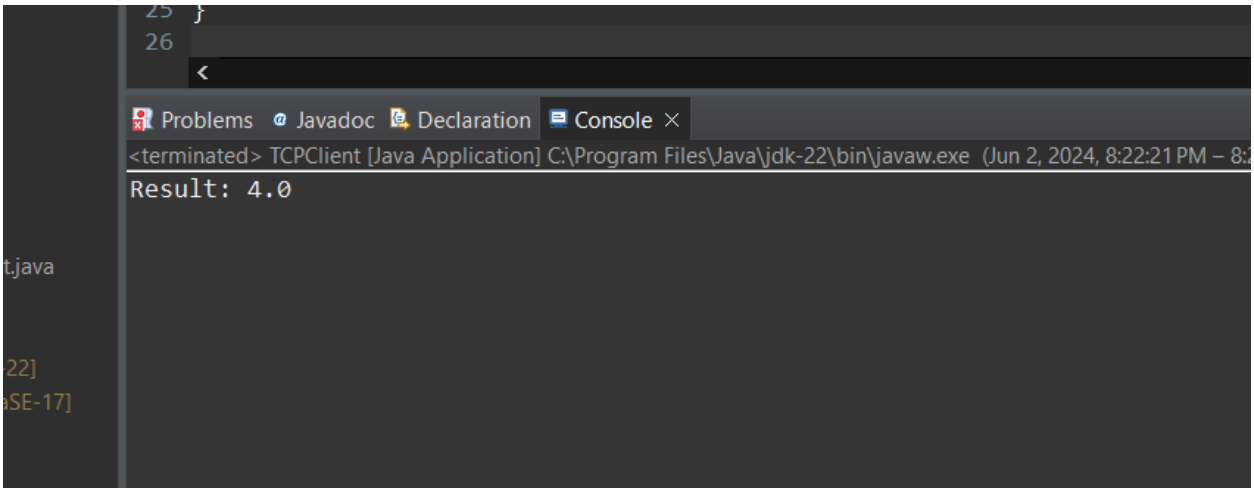
```
System.out.println("Server is listening on port 6789");

        while (true) {
            try (Socket socket = serverSocket.accept();
                 ObjectInputStream in = new ObjectInputStream(socket.getInputStream());
                 ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream())) {

                OperationRequest request = (OperationRequest) in.readObject();
                double result = performOperation(request);

                out.writeObject(result);
                out.flush();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private static double performOperation(OperationRequest request) {
    double number1 = request.getNumber1();
    double number2 = request.getNumber2();
    String operation = request.getOperation();

    switch (operation) {
        case "+":
            return number1 + number2;
```

```
            case "-":
                return number1 - number2;
            case "*":
                return number1 * number2;
            case "/":
                if (number2 != 0) {
                    return number1 / number2;
                } else {
                    throw new ArithmeticException("Division by zero");
                }
            default:
                throw new UnsupportedOperationException("Unknown operation: " + operation);
        }
    }
}

package Practice2;

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

public class TCPClient {
    public static void main(String[] args) {
        String serverAddress = "localhost";
        int serverPort = 6789;

        try (Socket socket = new Socket(serverAddress, serverPort);
             ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());
             ObjectInputStream in = new ObjectInputStream(socket.getInputStream())) {
```

```
            OperationRequest request = new OperationRequest(2, 2, "+");
            out.writeObject(request);
            out.flush();

            double result = (double) in.readObject();
            System.out.println("Result: " + result);
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT:





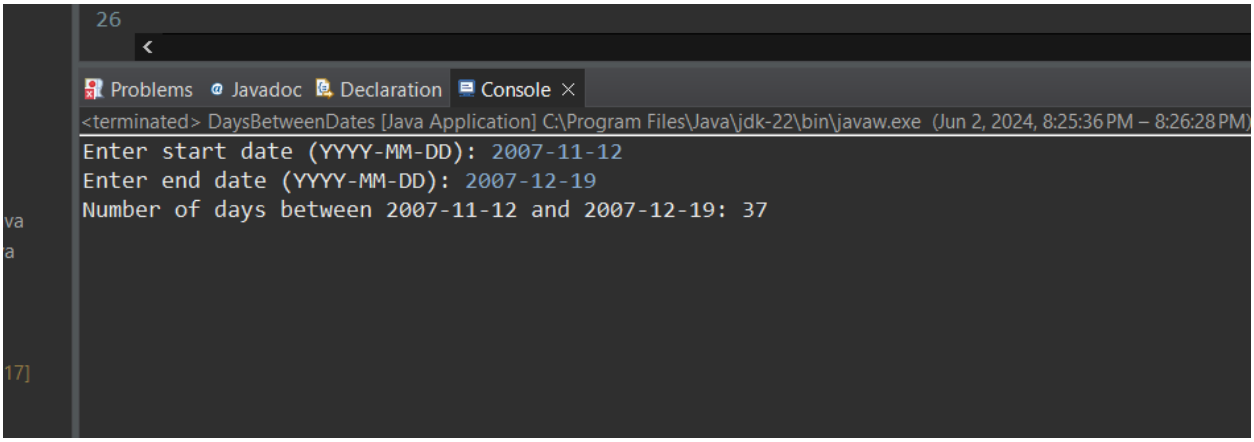
Task 6: Java 8 Date and Time API

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

Ans:

```
package Practice2;


import java.util.*;
import java.time.LocalDate;


import java.time.temporal.ChronoUnit;


public class DaysBetweenDates {


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


      System.out.print("Enter start date (YYYY-MM-DD): ");
      String startDateStr = scanner.nextLine();
      LocalDate startDate = LocalDate.parse(startDateStr);


      System.out.print("Enter end date (YYYY-MM-DD): ");
      String endDateStr = scanner.nextLine();
      LocalDate endDate = LocalDate.parse(endDateStr);



      long daysBetween = ChronoUnit.DAYS.between(startDate, endDate);
      System.out.println("Number of days between " + startDate + " and " + endDate + ": " +
daysBetween);
   }
}
```

Output:



Task 7: Timezone

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

Ans:

```
package Practice2;


import java.time.Instant;

import java.time.LocalDateTime;

import java.time.ZoneId;

import java.time.ZonedDateTime;

import java.time.format.DateTimeFormatter;

import java.util.Scanner;


public class TimezoneConverter {


    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);


        System.out.print("Enter the time (format: YYYY-MM-DD HH:mm): ");

        String timeStr = scanner.nextLine();
```

```
        System.out.print("Enter source timezone (e.g., America/Los_Angeles): ");
        String sourceZoneId = scanner.nextLine();

        System.out.print("Enter target timezone (e.g., Asia/Kolkata): ");
        String targetZoneId = scanner.nextLine();

        LocalDateTime localDateTime = LocalDateTime.parse(timeStr, DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm"));
        ZonedDateTime sourceDateTime = localDateTime.atZone(ZoneId.of(sourceZoneId));

        ZonedDateTime targetDateTime = sourceDateTime.withZoneSameInstant(ZoneId.of(targetZoneId));

        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm z");
        String convertedTime = targetDateTime.format(formatter);
        System.out.println("Converted time in " + targetZoneId + ": " + convertedTime);
    }
}
```

OutPut: