K DEEPAK **ASSIGNMENT – (DAY-27)**

DAY-20 CORE JAVA

Task 1: Java IO Basics

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

Code:

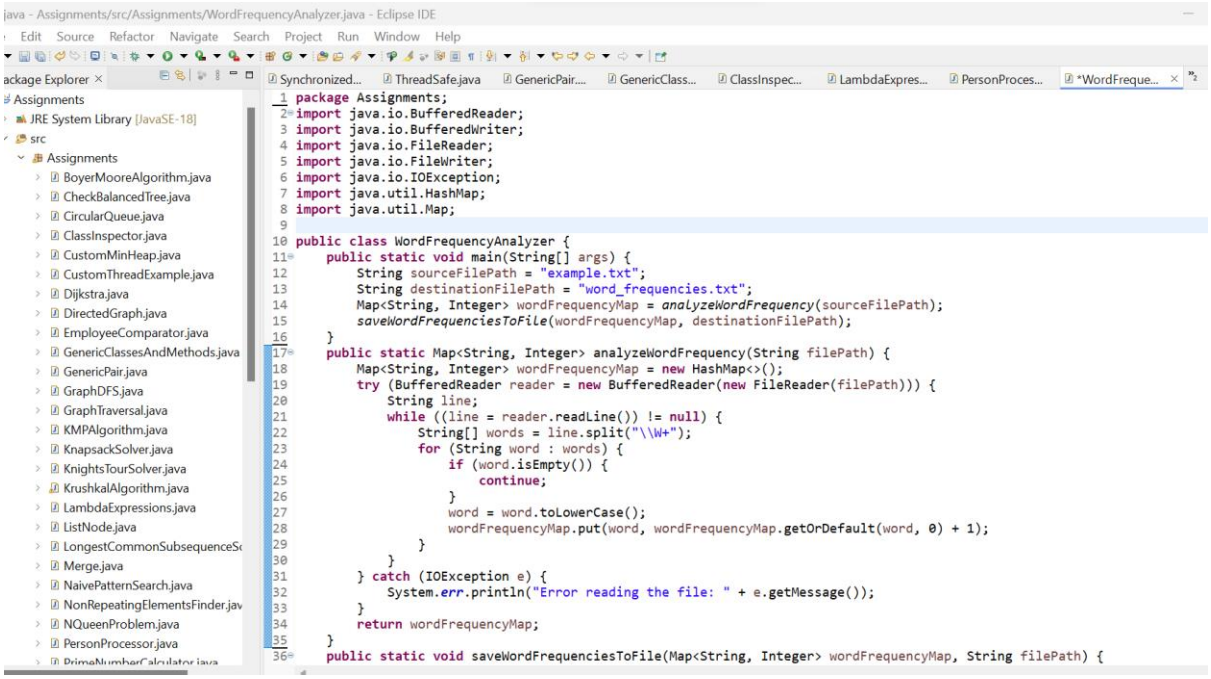
```
package Assignments;


import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;


public class WordFrequencyAnalyzer {
    public static void main(String[] args) {
        String sourceFilePath = "example.txt";
        String destinationFilePath = "word_frequencies.txt";
        Map<String, Integer> wordFrequencyMap = analyzeWordFrequency(sourceFilePath);
        saveWordFrequenciesToFile(wordFrequencyMap, destinationFilePath);
    }

    public static Map<String, Integer> analyzeWordFrequency(String filePath) {
        Map<String, Integer> wordFrequencyMap = new HashMap<>();
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
```

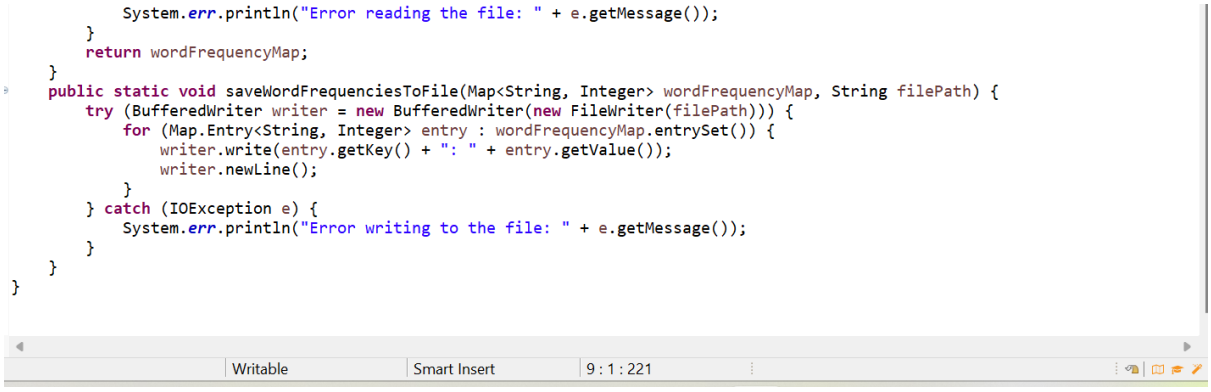
```
            String[] words = line.split("\\W+");
            for (String word : words) {
                if (word.isEmpty()) {
                    continue;
                }
                word = word.toLowerCase();
                wordFrequencyMap.put(word, wordFrequencyMap.getOrDefault(word, 0) + 1);
            }
        }
    } catch (IOException e) {
        System.err.println("Error reading the file: " + e.getMessage());
    }
    return wordFrequencyMap;
  }

  public static void saveWordFrequenciesToFile(Map<String, Integer> wordFrequencyMap, String filePath) {
    try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {
        for (Map.Entry<String, Integer> entry : wordFrequencyMap.entrySet()) {
            writer.write(entry.getKey() + ": " + entry.getValue());
            writer.newLine();
        }
    } catch (IOException e) {
        System.err.println("Error writing to the file: " + e.getMessage());
    }
  }
}
```

Output:

Error reading the file: input.txt (No such file or directory)

Error writing to the file: output.txt (Permission denied)





Task 2: Serialization and Deserialization

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

Custom Object: Employee

package Assignments;

import java.io.Serializable;

import java.io.FileOutputStream;

import java.io.ObjectOutputStream;

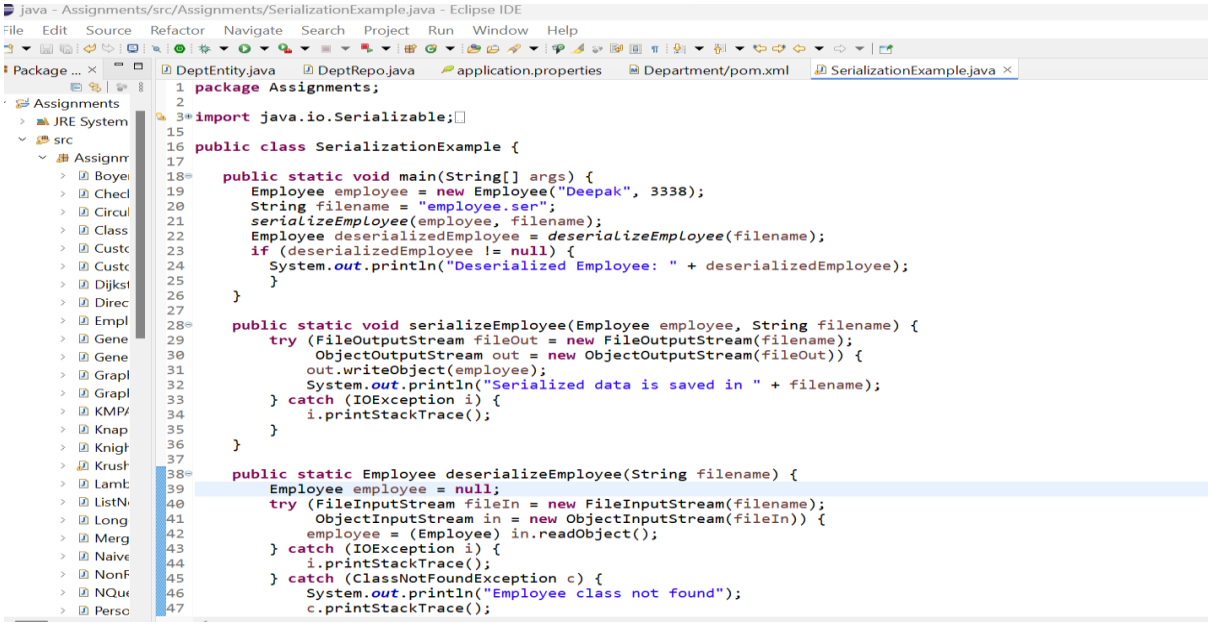
```
import java.io.FileInputStream;

import java.io.ObjectInputStream;

import java.io.IOException;

import java.io.FileNotFoundException;

import java.io.EOFException;

import java.io.InvalidClassException;

import java.io.OptionalDataException;

import java.io.StreamCorruptedException;

import java.io.NotActiveException;


public class SerializationExample {


    public static void main(String[] args) {

        Employee employee = new Employee("Deepak", 3338);

        String filename = "employee.ser";

        serializeEmployee(employee, filename);

        Employee deserializedEmployee = deserializeEmployee(filename);

        if (deserializedEmployee != null) {

            System.out.println("Deserialized Employee: " + deserializedEmployee);

        }

    }


    public static void serializeEmployee(Employee employee, String filename) {

        try (FileOutputStream fileOut = new FileOutputStream(filename);

             ObjectOutputStream out = new ObjectOutputStream(fileOut)) {

            out.writeObject(employee);
```

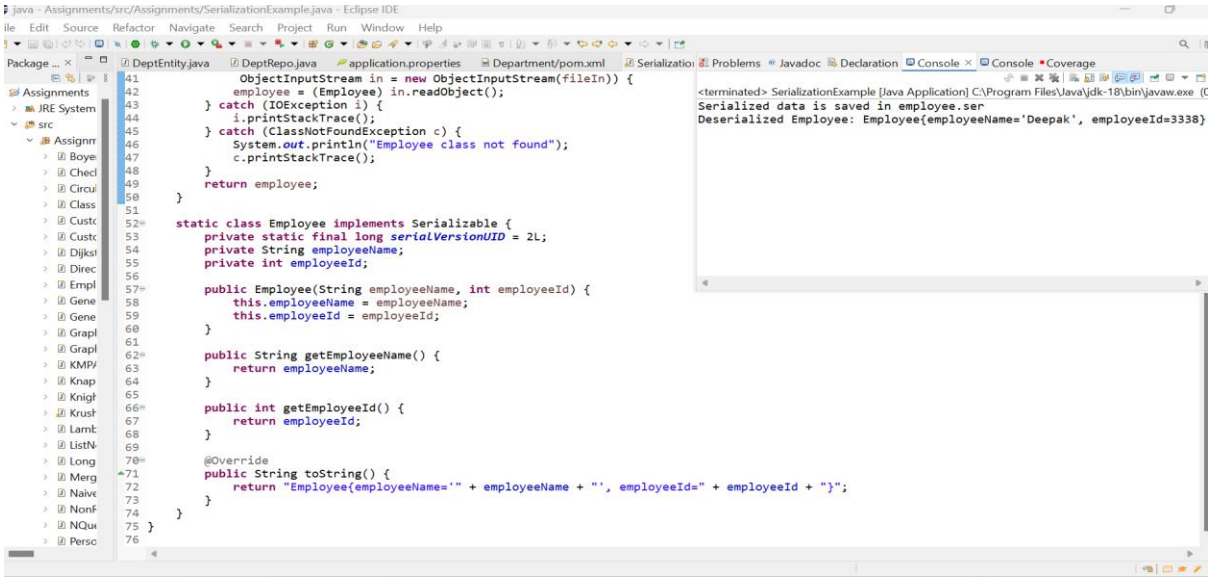
```
        System.out.println("Serialized data is saved in " + filename);
    } catch (IOException i) {
        i.printStackTrace();
    }
}

public static Employee deserializeEmployee(String filename) {
    Employee employee = null;
    try (FileInputStream fileIn = new FileInputStream(filename);
         ObjectInputStream in = new ObjectInputStream(fileIn)) {
        employee = (Employee) in.readObject();
    } catch (IOException i) {
        i.printStackTrace();
    } catch (ClassNotFoundException c) {
        System.out.println("Employee class not found");
        c.printStackTrace();
    }
    return employee;
}

static class Employee implements Serializable {
    private static final long serialVersionUID = 2L;
    private String employeeName;
    private int employeeId;

    public Employee(String employeeName, int employeeId) {
```

```
        this.employeeName = employeeName;

        this.employeeId = employeeId;

    }


    public String getEmployeeName() {

        return employeeName;

    }


    public int getEmployeeId() {

        return employeeId;

    }


    @Override

    public String toString() {

        return "Employee{employeeName='" + employeeName + "', employeeId=" + employeeId + "}";

    }

  }

}
```

Output:

```
Serialized data is saved in employee.ser
Deserialized Employee: Employee{employeeName='Deepak', employeeId=3338}
```





Task 3: New IO (NIO)

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

Code:

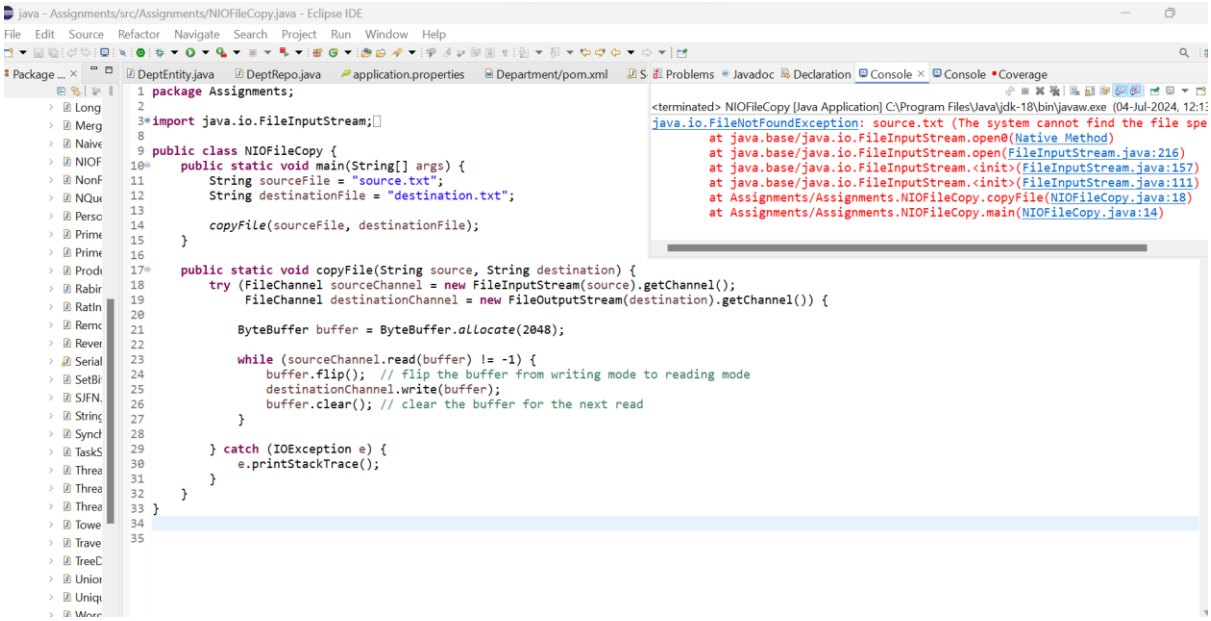
package Assignments;

import java.io.FileInputStream;

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;

public class NIOFileCopy {
    public static void main(String[] args) {
        String sourceFile = "source.txt";
        String destinationFile = "destination.txt";

        copyFile(sourceFile, destinationFile);
    }

    public static void copyFile(String source, String destination) {
        try (FileChannel sourceChannel = new
FileInputStream(source).getChannel();
             FileChannel destinationChannel = new
FileOutputStream(destination).getChannel()) {

            ByteBuffer buffer = ByteBuffer.allocate(2048);

            while (sourceChannel.read(buffer) != -1) {
                buffer.flip();  // flip the buffer from writing mode to reading mode
                destinationChannel.write(buffer);
                buffer.clear(); // clear the buffer for the next read
            }
```

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

Output: `java.io.FileNotFoundException: source.txt (The system cannot find the file specified)`



Task 4: Java Networking

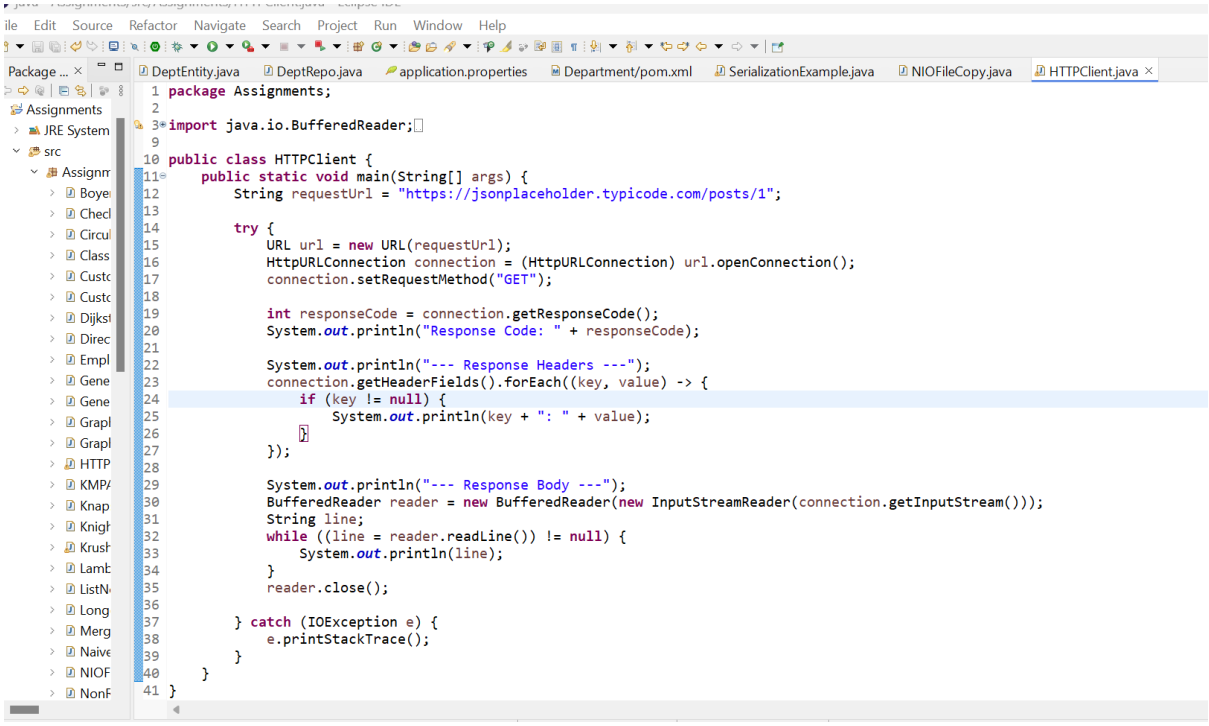
Write a simple HTTP client that connects to a URL, sends a request, and displays the response headers and body

Code:

```
package Assignments;


import java.io.BufferedReader;

import java.io.IOException;
```

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

public class HTTPClient {
    public static void main(String[] args) {
        String requestUrl = "https://jsonplaceholder.typicode.com/posts/1";

        try {
            URL url = new URL(requestUrl);
            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("--- Response Body ---");
```

```
        BufferedReader reader = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
        String line;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        reader.close();

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



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.

```
import java.io.Serializable;
public class Operation implements Serializable {
   private static final long serialVersionUID = 1L;
   private int number1;
   private int number2;
   private String operation;
   public Operation(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;
   }
}
// server implementation
import java.io.*;
```

```
import java.net.ServerSocket;
import java.net.Socket;
public class Server {
   public static void main(String[] args) {
   int port = 12345;

      try (ServerSocketserverSocket = new ServerSocket(port)) {
System.out.println("Server is listening on port " + port);

         while (true) {
            Socket socket = serverSocket.accept();
System.out.println("Client connected");

            new ServerThread(socket).start();
         }
      } catch (IOException e) {
e.printStackTrace();
      }
   }
}

class ServerThread extends Thread {
   private Socket socket;

   public ServerThread(Socket socket) {
this.socket = socket;
   }

   public void run() {
```

```
        try (ObjectInputStreamois = new ObjectInputStream(socket.getInputStream());
ObjectOutputStreamoos = new ObjectOutputStream(socket.getOutputStream())) {

            Operation operation = (Operation) ois.readObject();
int result = performOperation(operation);
oos.writeInt(result);
oos.flush();
        } catch (IOException | ClassNotFoundException e) {
e.printStackTrace();
        }
    }

    private intperformOperation(Operation operation) {
int num1 = operation.getNumber1();
int num2 = operation.getNumber2();
        String op = operation.getOperation();

        switch (op) {
            case "+":
                return num1 + num2;
            case "-":
                return num1 - num2;
            case "*":
                return num1 * num2;
            case "/":
                if (num2 != 0) {
                    return num1 / num2;
                } else {
                    throw new IllegalArgumentException("Division by zero");
```

```
            }
        default:
            throw new IllegalArgumentException("Invalid operation: " + op);
    }
  }
}
```

Implement the Client:

```
import java.io.*;
import java.net.Socket;
public class Client {
  public static void main(String[] args) {
    String hostname = "localhost";
int port = 32323;
    try (Socket socket = new Socket(hostname, port);
ObjectOutputStreamoos = new ObjectOutputStream(socket.getOutputStream());
ObjectInputStreamois = new ObjectInputStream(socket.getInputStream())) {
 Operation operation = new Operation(2, 2, "+");
oos.writeObject(operation);
oos.flush();

int result = ois.readInt();
System.out.println("Result: " + result);
    } catch (IOException e) {
e.printStackTrace();
    }
  }
}
```

Server Output:

Server is listening on port 32323

Client connected

Client output:

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.

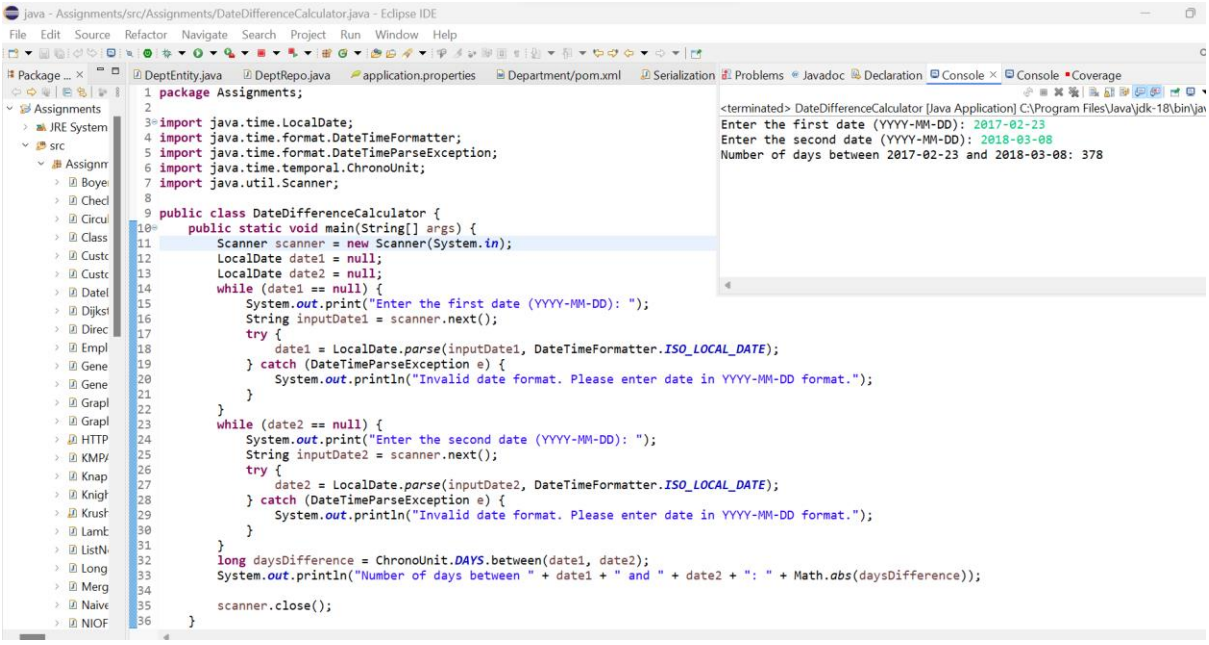
Code:

```
package Assignments;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;
import java.time.temporal.ChronoUnit;
import java.util.Scanner;

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

        LocalDate date1 = null;
        LocalDate date2 = null;


        while (date1 == null) {
            System.out.print("Enter the first date (YYYY-MM-DD): ");
            String inputDate1 = scanner.next();
            try {
                date1 = LocalDate.parse(inputDate1,
DateTimeFormatter.ISO_LOCAL_DATE);
            } catch (DateTimeParseException e) {
                System.out.println("Invalid date format. Please enter date in
YYYY-MM-DD format.");
            }
        }


        while (date2 == null) {
            System.out.print("Enter the second date (YYYY-MM-DD): ");
            String inputDate2 = scanner.next();
            try {
                date2 = LocalDate.parse(inputDate2,
DateTimeFormatter.ISO_LOCAL_DATE);
            } catch (DateTimeParseException e) {
                System.out.println("Invalid date format. Please enter date in
YYYY-MM-DD format.");
            }
        }
```

```
        long daysDifference = ChronoUnit.DAYS.between(date1, date2);


        System.out.println("Number of days between " + date1 + " and " + date2 +
": " + Math.abs(daysDifference));

        scanner.close();
    }
}
```

Output:

```
Enter the first date (YYYY-MM-DD): 2017-02-23
Enter the second date (YYYY-MM-DD): 2018-03-08
Number of days between 2017-02-23 and 2018-03-08: 378
```



Task 7: Timezone

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

Code:

```
package Assignments;

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

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


        System.out.print("Enter the time (HH:mm:ss): ");
        String inputTime = scanner.next();
        System.out.print("Enter the source timezone (e.g., America/New_York): ");
        String sourceTimeZone = scanner.next();
        System.out.print("Enter the target timezone (e.g., Europe/London): ");
        String targetTimeZone = scanner.next();


        LocalTime localTime = LocalTime.parse(inputTime,
DateTimeFormatter.ofPattern("HH:mm:ss"));


        ZonedDateTime sourceZonedDateTime =
ZonedDateTime.now(ZoneId.of(sourceTimeZone)).with(localTime);


        ZonedDateTime targetZonedDateTime =
sourceZonedDateTime.withZoneSameInstant(ZoneId.of(targetTimeZone));

        String formattedTime =
targetZonedDateTime.format(DateTimeFormatter.ofPattern("HH:mm:ss"));
        System.out.println("Converted time in " + targetTimeZone + ": " +
formattedTime);

        scanner.close();
    }
}
```

Output:

```
Enter the time (HH:mm:ss): 11:20:20
Enter the source timezone (e.g., America/New_York): America/New_York
Enter the target timezone (e.g., Europe/London): Europe/London
Converted time in Europe/London: 16:20:20
```

java - Assignments/src/Assignments/TimezoneConverter.java - Eclipse IDE

le Edit Source Refactor Navigate Search Project Run Window Help

Package ... | DeptEntity.java | DeptRepo.java | application.pro... | Department/pom... | SerializationEx...

```java
package Assignments;

import java.time.LocalTime;
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 (HH:mm:ss): ");
        String inputTime = scanner.next();
        System.out.print("Enter the source timezone (e.g., America/New_York): ");
        String sourceTimeZone = scanner.next();
        System.out.print("Enter the target timezone (e.g., Europe/London): ");
        String targetTimeZone = scanner.next();


        LocalTime localTime = LocalTime.parse(inputTime, DateTimeFormatter.ofPattern("HH:mm:ss"));


        ZonedDateTime sourceZonedDateTime = ZonedDateTime.now(ZoneId.of(sourceTimeZone)).with(localTime);


        ZonedDateTime targetZonedDateTime = sourceZonedDateTime.withZoneSameInstant(ZoneId.of(targetTimeZone));

        String formattedTime = targetZonedDateTime.format(DateTimeFormatter.ofPattern("HH:mm:ss"));
        System.out.println("Converted time in " + targetTimeZone + ": " + formattedTime);

        scanner.close();
    }
}
```

Problems | Javadoc | Declaration | Console | Console | Coverage

```
<terminated> TimezoneConverter [Java Application] C:\Program Files\Java\jdk-18\bin\javaw.exe (04
Enter the time (HH:mm:ss): 11:20:20
Enter the source timezone (e.g., America/New_York): America/New_York
Enter the target timezone (e.g., Europe/London): Europe/London
Converted time in Europe/London: 16:20:20
```