

## 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.**

```
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 WordFrequencyCounter {

    public static void main(String[] args) {

        String inputFile = "input.txt";
        String outputFile = "output.txt";

        Map<String, Integer> wordFreqMap = new HashMap<>();
        try (BufferedReader reader = new BufferedReader(new FileReader(inputFile))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] words = line.split("\\s+");
                for (String word : words) {
                    word = word.toLowerCase();
                    wordFreqMap.put(word, wordFreqMap.getOrDefault(word, 0) + 1);
                }
            }
        } catch (IOException e) {
```

```

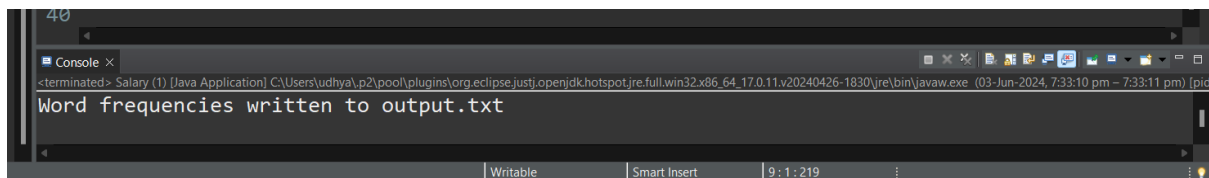
        e.printStackTrace();
    }

    try (BufferedWriter writer = new BufferedWriter(new FileWriter(outputFile))) {
        for (Map.Entry<String, Integer> entry : wordFreqMap.entrySet()) {
            writer.write(entry.getKey() + ": " + entry.getValue() + "\n");
        }
    } catch (IOException e) {
        e.printStackTrace();
    }

    System.out.println("Word frequencies written to " + outputFile);
}
}

```

## OUTPUT:



## Task 2: Serialization and Deserialization

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

```
import java.io.*
```

```

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

    public CustomObject(String name, int age) {

```

```

        this.name = name;

        this.age = age;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }
}

public class SerializationExample {

    public static void main(String[] args) {
        CustomObject obj = new CustomObject("John", 30);

        try (ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream("object.ser"))) {
            outputStream.writeObject(obj);

            System.out.println("Object serialized successfully.");
        } catch (IOException e) {
            e.printStackTrace();
        }

        try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream("object.ser"))) {
            CustomObject deserializedObj = (CustomObject) inputStream.readObject();

            System.out.println("Object deserialized successfully.");

            System.out.println("Name: " + deserializedObj.getName());

            System.out.println("Age: " + deserializedObj.getAge());
        }
    }
}

```

```

    } catch (IOException | ClassNotFoundException e) {

        e.printStackTrace();

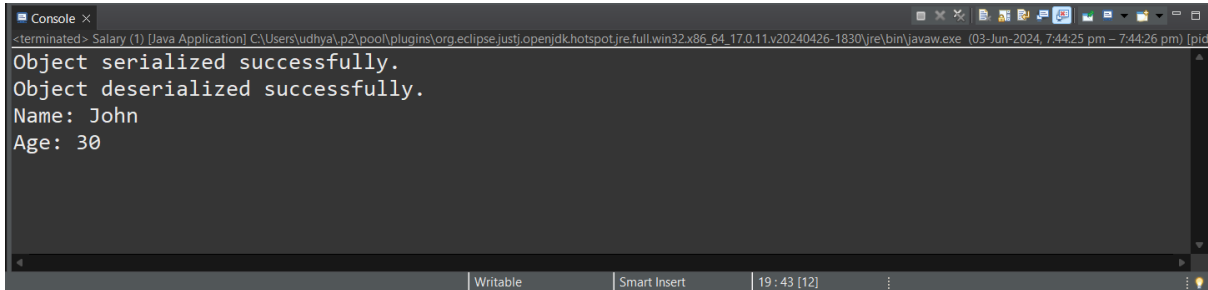
    }

}

}

```

## OUTPUT:



```

<terminated> Salary (1) [Java Application] C:\Users\uudhya\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.11.v20240426-1830\jre\bin\javaw.exe (03-Jun-2024, 7:44:25 pm - 7:44:26 pm) [pid
Object serialized successfully.
Object deserialized successfully.
Name: John
Age: 30

```

## Task 3: New IO (NIO)

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

```

package com.wipro;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.StandardOpenOption;

import java.util.Iterator;

import java.util.List;

public class Mynio {

    String fileName = "mydir/rhymes.txt";

    public void createDirectory() {

        Path p = Paths.get("mydir");

        try {

            if (Files.exists(p)) {

```

```

        System.out.println("Directory already exists");
    } else {
        Path cPath = Files.createDirectories(p);
        System.out.println("Directory created at " + cPath.toString());
    }
} catch (Exception e) {
    e.printStackTrace();
}
}

public void createFile(String fileName) {
    Path f = Paths.get(fileName);
    try {
        if (Files.exists(f)) {
            System.out.println("File already exists");
        } else {
            Path cFile = Files.createFile(f);
            System.out.println("Directory created at " + cFile.toString());
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void readFile() {
    Path f = Paths.get(fileName);
    try {
        List<String> data = Files.readAllLines(f);
        for (String str : data) {
            System.out.println(str);
        }
    }
}

```

```

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

    public void writeFile(String fileName) {
        Path f = Paths.get(fileName);
        try {
            String content = " Johny Johny , Yes Papa,\n Eating sugar ? No Papa";
            Files.write(f, content.getBytes());
            System.out.println("Data Written Successfully");
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public void appendFile(String fileName) {
        Path f = Paths.get(fileName);
        try {
            String content = "\n Telling Lies ? No Papa,\n Open your Mouth, Ha Ha
Ha :)";
            Files.write(f, content.getBytes(), StandardOpenOption.APPEND);
            System.out.println("Data Appended Successfully");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {

```

```
Mynio mn = new Mynio();

        mn.createDirectory();

mn.createFile("mydir/rhymes.txt");

        mn.readFile();
        mn.writeFile(mn.fileName);


mn.readFile();

mn.appendFile(mn.fileName);

mn.readFile();
    }

}
```

**OUTPUT:**

```
Console x
<terminated> Mymio [Java Application] C:\Users\udhya\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.11.v20240426-1830\jre\bin\javaw.exe (03-Jun-2024, 7:41:05 pm - 7:41:06 pm) [pid: 24]
Directory already exists
File already exists
Johnny Johnny , Yes Papa,
Eating sugar ? No Papa
Telling Lies ? No Papa,
Open your Mouth, Ha Ha Ha :)
Data Written Successfully
Johnny Johnny , Yes Papa,
Eating sugar ? No Papa
Data Appended Successfully
Johnny Johnny , Yes Papa,
Eating sugar ? No Papa
Telling Lies ? No Papa,
Open your Mouth, Ha Ha Ha :)

Writable | Smart Insert | 29 : 14 : 731
```

## Task 4: Java Networking

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

### CLIENT

```
package com.java8;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.InetAddress;
import java.net.Socket;
import java.net.UnknownHostException;
```



```

public class Clientpgm {

    public static void main(String[] args) {
        try {
            InetAddress addr = InetAddress.getByName("localhost");
            Socket soc = new Socket(addr, Serverpgm.port);
            System.out.println(soc);

            BufferedReader br = new BufferedReader(new
InputStreamReader(soc.getInputStream()));

            PrintWriter out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(soc.getOutputStream()),true);

            for(int i =1;i<=10;i++) {
                out.println("Hello Server take " + i);
            }

            out.println("End");
            System.out.println(br.readLine());
        } catch (UnknownHostException e) {

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

            e.printStackTrace();
        }
    }
}

```

```
}
```

## **SERVER**

```
package com.java8;
```

```
import java.io.BufferedReader;
```

```
import java.io.BufferedWriter;
```

```
import java.io.IOException;
```

```
import java.io.InputStreamReader;
```

```
import java.io.OutputStreamWriter;
```

```
import java.io.PrintWriter;
```

```
import java.net.ServerSocket;
```

```
import java.net.Socket;
```

```
public class Serverpgm {
```

```
    static int port=2024;
```

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

```
        try {
```

```
            ServerSocket s = new ServerSocket(port);
```

```
            System.out.println("The server socket is created " +s);
```

```
            Socket soc=s.accept();
```

```
            System.out.println("The server accepted the client");
```

```
            BufferedReader br = new BufferedReader(new  
InputStreamReader(soc.getInputStream()));
```

```
            PrintWriter out = new PrintWriter(new BufferedWriter(new  
OutputStreamWriter(soc.getOutputStream()),true);
```

```

        while(true) {
            String str = br.readLine();
            if(str.equals("End")) {
                break;
            }
            System.out.println(str);
            out.println("Ok Received. Thank you !!!");
        }

    } 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**

### Server Code

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

```

import java.net.*;

public class TCPServer {

    public static void main(String[] args) {

        int portNumber = 5555;

        try (ServerSocket serverSocket = new ServerSocket(portNumber)) {

            System.out.println("Server is running...");

            while (true) {

                try (

                    Socket clientSocket = serverSocket.accept();

                    ObjectInputStream in = new ObjectInputStream(clientSocket.getInputStream());

                    ObjectOutputStream out = new
ObjectOutputStream(clientSocket.getOutputStream());

                ) {

                    CalculationRequest request = (CalculationRequest) in.readObject();

                    int result = calculate(request);

                    out.writeObject(result);

                } catch (IOException | ClassNotFoundException e) {

                    e.printStackTrace();

                }

            } catch (IOException e) {

                e.printStackTrace();

            }

        }
    }
}

```

```

private static int calculate(CalculationRequest request) {

    int num1 = request.getNum1();

    int num2 = request.getNum2();

    char operation = request.getOperation();


    switch (operation) {

        case '+':

            return num1 + num2;

        case '-':

            return num1 - num2;

        case '*':

            return num1 * num2;

        case '/':

            if (num2 != 0)

                return num1 / num2;

            else

                throw new ArithmeticException("Division by zero!");

        default:

            throw new IllegalArgumentException("Invalid operation: " + operation);

    }

}

```

## Client Code

```

import java.io.*;

import java.net.*;


public class TCPClient {

    public static void main(String[] args) {

        String hostName = "localhost";

```

```

int portNumber = 5555;

try (
    Socket socket = new Socket(hostName, portNumber);
    ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());
    ObjectInputStream in = new ObjectInputStream(socket.getInputStream());
) {

    CalculationRequest request = new CalculationRequest(2, 2, '+');
    out.writeObject(request);

    int result = (int) in.readObject();

    System.out.println("Result from server: " + result);
} catch (IOException | ClassNotFoundException e) {
    e.printStackTrace();
}
}
}

```

## Task 6: Java 8 Date and Time API

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

```

package com.wipro;

import java.time.DayOfWeek;
import java.time.Duration;
import java.time.LocalDate;
import java.time.LocalDateTime;

```

```
import java.time.LocalDateTime;
import java.time.Month;
import java.time.Period;
import java.time.Year;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.TemporalAdjusters;

public class DateTimeEg {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        LocalDate localDate = LocalDate.now();
        System.out.println("Local Date : " + localDate);

        LocalDate cDate = LocalDate.of(2024 , Month.MAY,21);
        System.out.println("Custom Date : " + cDate);

        LocalDateTime localTime = LocalDateTime.now();
        System.out.println("Local Time : " + localTime);

        LocalDateTime cTime = LocalDateTime.of(15,29,45);
        System.out.println("Custom Time : " + cTime);

        LocalDateTime ldt = LocalDateTime.now();

        System.out.println("Local Date and Time : " + ldt);
```

```
LocalDateTime cldt = LocalDateTime.of (2024, Month.MAY, 21, 16, 29, 45);  
System.out.println("Local Date and Time :" + cldt);
```

```
ZonedDateTime zdt = ZonedDateTime.now();
```

```
System.out.println("Zoned Date and Time:" + zdt);
```

```
ZonedDateTime zdt1 = ZonedDateTime.of(cldt, ZoneId.of("Asia/Kolkata"));
```

```
System.out.println("Zoned Date and Time:" + zdt1);
```

```
Period period = Period.between(localDate, cDate);
```

```
System.out.println(period.getDays()+ " " +period.getMonths() + " " +  
period.getYears());
```

```
Duration duration = Duration.between(localTime, cTime);
```

```
System.out.println(duration.toHours()+ " " +duration.toMinutes() + " " +  
duration.toNanos());
```

```
DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd-MM-yyyy  
h:m:s");
```

```
String fDate = ldt.format(dtf);
```

```
System.out.println("Formatted Date : " + fDate);
```

```
checkDate(cDate);
```

```
int year =2024;
```

```
System.out.println(Year.isLeap(year));
```

```
System.out.println(localDate.minusMonths(2));
```



```
        System.out.println("firstDayOfMonth : "+  
localDate.with(TemporalAdjusters.firstDayOfMonth()));
```

```
        System.out.println("firstDayOfNextMonth :  
"+localDate.with(TemporalAdjusters.firstDayOfNextMonth()));
```

```
        System.out.println("firstDayOfNextYear :  
"+localDate.with(TemporalAdjusters.firstDayOfNextYear()));
```

```
        System.out.println("lastInMonth :"  
+localDate.with(TemporalAdjusters.lastInMonth(DayOfWeek.TUESDAY)));
```

```
        System.out.println("previous: "+localDate.with(TemporalAdjusters.previous  
(DayOfWeek.TUESDAY)));
```

```
    }
```

```
    public static void checkDate (LocalDate ld) {
```

```
        LocalDate today = LocalDate.now();
```

```
        if(ld.isBefore(today)) {
```

```
            System.out.println(ld + " is before today");
```

```
        }else if(ld.isAfter (today)) {
```

```
            System.out.println(ld + "is after today");
```

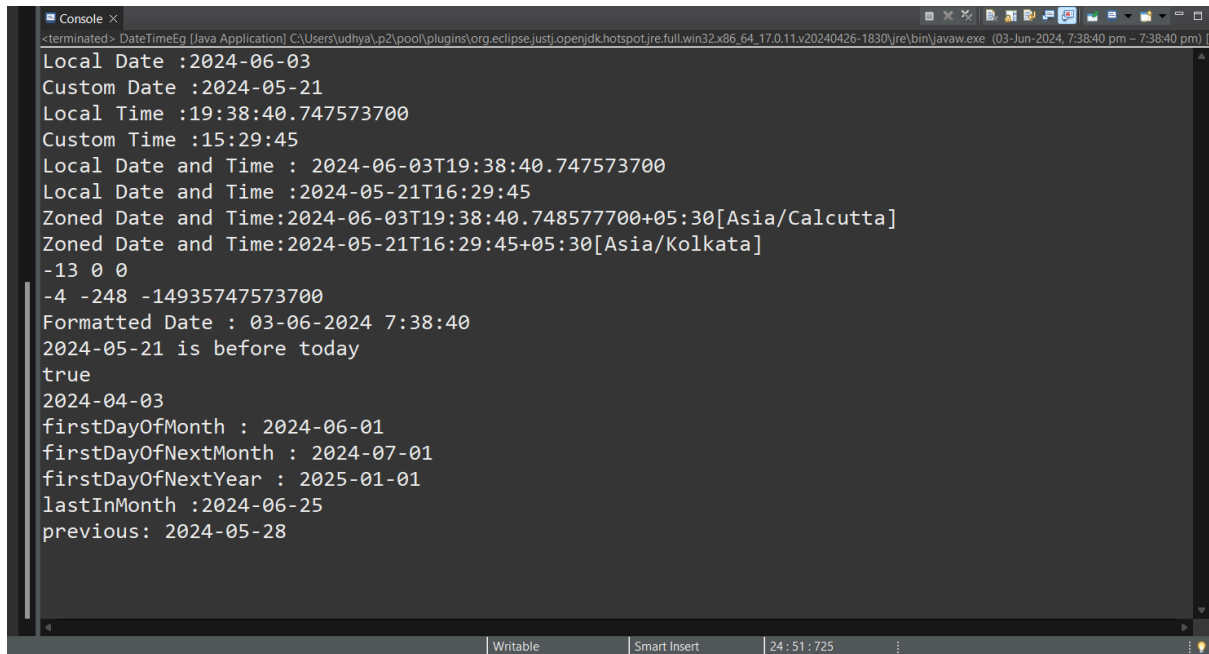
```
        }else {
```

```
            System.out.println(ld + " is today");
```

```
    }
```

}}

## OUTPUT:

A screenshot of a Java console window titled "Console x". The window shows the output of a Java application. The output includes local and custom dates and times, zoned dates and times for Asia/Calcutta and Asia/Kolkata, and various date-related methods like firstDayOfMonth, firstDayOfNextMonth, firstDayOfNextYear, lastInMonth, and previous. The console window has a dark background and a light-colored text. The status bar at the bottom shows "Writable", "Smart Insert", and "24:51:725".

```
<terminated> DateTimeEg [Java Application] C:\Users\udhya\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.11.v20240426-1830\jre\bin\javaw.exe (03-Jun-2024, 7:38:40 pm - 7:38:40 pm) [
Local Date :2024-06-03
Custom Date :2024-05-21
Local Time :19:38:40.747573700
Custom Time :15:29:45
Local Date and Time : 2024-06-03T19:38:40.747573700
Local Date and Time :2024-05-21T16:29:45
Zoned Date and Time:2024-06-03T19:38:40.748577700+05:30[Asia/Calcutta]
Zoned Date and Time:2024-05-21T16:29:45+05:30[Asia/Kolkata]
-13 0 0
-4 -248 -14935747573700
Formatted Date : 03-06-2024 7:38:40
2024-05-21 is before today
true
2024-04-03
firstDayOfMonth : 2024-06-01
firstDayOfNextMonth : 2024-07-01
firstDayOfNextYear : 2025-01-01
lastInMonth :2024-06-25
previous: 2024-05-28
```

## Task 7: Timezone

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

```
import java.time.*;
```

```
import java.time.format.DateTimeFormatter;
```

```
public class TimezoneConverter {
```

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

```
        LocalDateTime time = LocalDateTime.of(2024, 6, 1, 12, 0); // Assuming time is in UTC
```

```
        String fromTimezone = "UTC";
```

```
        String toTimezone = "America/New_York";
```

```
        LocalDateTime convertedTime = convertTimezone(time, fromTimezone, toTimezone);
```

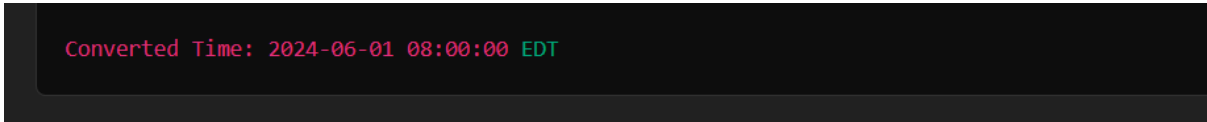
```
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss z");
System.out.println("Converted Time: " + formatter.format(convertedTime));
}

public static LocalDateTime convertTimezone(LocalDateTime time, String fromTimezone, String
toTimezone) {
    ZoneId fromZoneId = ZoneId.of(fromTimezone);
    ZoneId toZoneId = ZoneId.of(toTimezone);

    ZonedDateTime zonedDateTime = ZonedDateTime.of(time, ZoneOffset.UTC);
    ZonedDateTime convertedZonedDateTime = zonedDateTime.withZoneSameInstant(toZoneId);

    return convertedZonedDateTime.toLocalDateTime();
}
}
```

## OUTPUT:



```
Converted Time: 2024-06-01 08:00:00 EDT
```