Day 27 Assignment

Name: Mehul Anjikhane   Email: mehulanjikhane13@gmail.com

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

```
package assignments;

import java.io.*;
import java.util.*;

public class WordFrequency {
	public static void main(String[] args) {
		String inputFile = "input.txt";
		String outputFile = "output.txt";

		try {
			FileReader fileReader = new FileReader(inputFile);
			BufferedReader bufferedReader = new BufferedReader(fileReader);

			Map<String, Integer> wordCount = new HashMap<>();
			String line;
			while ((line = bufferedReader.readLine()) != null) {
				String[] words = line.split("\\W+");
				for (String word : words) {
					word = word.toLowerCase();
					wordCount.put(word, wordCount.getOrDefault(word, 0)
+ 1);
				}
			}
			bufferedReader.close();

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

Input.txt

```
This is a sample text file. This file is for testing word frequency.
```

output.txt

```
a: 1
file: 2
testing: 1
this: 2
for: 1
is: 2
text: 1
sample: 1
word: 1
frequency: 1
```

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

```
package assignments;

import java.io.*;

class Persons implements Serializable {
	private static final long serialVersionUID = 1L;
	String name;
	int age;

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

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

public class SerializationExample {
	public static void main(String[] args) {
		Persons person = new Persons("John Doe", 30);
		String filename = "person.ser";

		// Serialization
		try (FileOutputStream fileOut = new FileOutputStream(filename);
				ObjectOutputStream out = new ObjectOutputStream(fileOut))
{
			out.writeObject(person);
		} catch (IOException i) {
			i.printStackTrace();
		}

		// Deserialization
		Persons deserializedPerson = null;
		try (FileInputStream fileIn = new FileInputStream(filename);
				ObjectInputStream in = new ObjectInputStream(fileIn)) {
```

```
                deserializedPerson = (Persons) in.readObject();
            } catch (IOException | ClassNotFoundException i) {
                i.printStackTrace();
            }

            System.out.println("Deserialized Person: " + deserializedPerson);
        }
}
```

Output:

```
Deserialized Person: Person{name='John Doe', age=30}
```

Task 3: New IO (NIO)

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

```
package assignments;

import java.io.*;
import java.nio.*;
import java.nio.channels.*;
import java.nio.file.*;

public class NIOExample {
        public static void main(String[] args) {
            String inputFile = "input.txt";
            String outputFile = "output.txt";

            try (FileChannel inputChannel = FileChannel.open(Paths.get(inputFile),
StandardOpenOption.READ);
                    FileChannel outputChannel =
FileChannel.open(Paths.get(outputFile), StandardOpenOption.WRITE,
                                        StandardOpenOption.CREATE)) {

                ByteBuffer buffer = ByteBuffer.allocate(1024);
                while (inputChannel.read(buffer) > 0) {
                    buffer.flip();
                    outputChannel.write(buffer);
                    buffer.clear();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
}
```

input.txt

```
This is a sample text file. This file is for NIO Example Class.
```

output.txt

```
This is a sample text file. This file is for NIO Example Class.
```

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

```
package assignments;

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

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

                try {
                        URL url = new ~~URL~~(urlString);
                        HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
                        connection.setRequestMethod("GET");

                        // Get response headers
                        Map<String, List<String>> headers = connection.getHeaderFields();
                        for (Map.Entry<String, List<String>> entry : headers.entrySet())
{
                                System.out.println(entry.getKey() + ": " +
entry.getValue());
                        }

                        // Get response body
                        BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
                        String inputLine;
                        StringBuilder response = new StringBuilder();
                        while ((inputLine = in.readLine()) != null) {
                                response.append(inputLine);
                        }
                        in.close();

                        System.out.println("Response body: " + response.toString());
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }
}
```

Output:

```
null: [HTTP/1.1 200 OK]
X-Cache: [HIT]
Server: [ECAcc (sac/255D)]
Last-Modified: [Thu, 17 Oct 2019 07:18:26 GMT]
Date: [Sat, 08 Jun 2024 11:43:56 GMT]
Accept-Ranges: [bytes]
Cache-Control: [max-age=604800]
```

```
Etag: ["3147526947"]
Vary: [Accept-Encoding]
Expires: [Sat, 15 Jun 2024 11:43:56 GMT]
Content-Length: [1256]
Age: [432631]
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;            }    div {        width: 600px;        margin: 5em auto;        padding: 2em;        background-color: #fdfdff;        border-radius: 0.5em;        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);    }    a:link, a:visited {        color: #38488f;        text-decoration: none;    }    @media (max-width: 700px) {        div {            margin: 0 auto;            width: auto;        }    }    </style></head><body><div>    <h1>Example Domain</h1>    <p>This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.</p>    <p><a href="https://www.iana.org/domains/example">More information...</a></p></div></body></html>
```

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

```java
package assignments;

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

class CalculationRequest implements Serializable {
	private static final long serialVersionUID = 1L;
	int number1;
	int number2;
	String operation;

	CalculationRequest(int number1, int number2, String operation) {
		this.number1 = number1;
		this.number2 = number2;
		this.operation = operation;
	}
}

public class CalculationServer {
	public static void main(String[] args) {
		try (ServerSocket serverSocket = new ServerSocket(9090)) {
			System.out.println("Server is listening on port 9090");
```

```
                while (true) {
                    Socket socket = serverSocket.accept();
                    new CalculationHandler(socket).start();
                }
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
}

class CalculationHandler extends Thread {
        private Socket socket;

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

        public void run() {
            try (ObjectInputStream in = new
ObjectInputStream(socket.getInputStream());
                    ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream())) {

                CalculationRequest request = (CalculationRequest)
in.readObject();
                int result = 0;
                switch (request.operation) {
                case "+":
                    result = request.number1 + request.number2;
                    break;
                case "-":
                    result = request.number1 - request.number2;
                    break;
                case "*":
                    result = request.number1 * request.number2;
                    break;
                case "/":
                    if (request.number2 != 0) {
                        result = request.number1 / request.number2;
                    } else {
                        out.writeObject("Error: Division by zero");
                        return;
                    }
                    break;
                default:
                    out.writeObject("Error: Invalid operation");
                    return;
                }

                out.writeObject(result);
            } catch (IOException | ClassNotFoundException ex) {
                ex.printStackTrace();
            }
        }
}
```

Output:

```
Server is listening on port 9090
```

Client Code

```
package assignments;

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

public class CalculationClient {
	public static void main(String[] args) {
		String hostname = "localhost";
		int port = 1234;

		CalculationRequest request = new CalculationRequest(2, 2, "+");

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

			out.writeObject(request);
			Object response = in.readObject();

			System.out.println("Result: " + response);
		} catch (IOException | ClassNotFoundException ex) {
			ex.printStackTrace();
		}
	}
}
```

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."

```
package assignments;

import java.time.*;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;

public class DaysBetweenDates {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");

		System.out.println("Enter the first date (yyyy-MM-dd): ");
		String firstDateInput = scanner.nextLine();
		LocalDate firstDate = LocalDate.parse(firstDateInput, formatter);
		System.out.println("Enter the second date (yyyy-MM-dd): ");
		String secondDateInput = scanner.nextLine();
		LocalDate secondDate = LocalDate.parse(secondDateInput, formatter);

		long daysBetween = Duration.between(firstDate.atStartOfDay(), 
secondDate.atStartOfDay()).toDays();

		System.out.println("Number of days between: " + daysBetween);
		scanner.close();
	}

}
```

Output:

```
Enter the first date (yyyy-MM-dd): 
1947-08-15
Enter the second date (yyyy-MM-dd): 
2014-05-16
Number of days between: 24381
```

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

```
package assignments;

import java.time.*;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;

public class TimezoneConverter {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd 
HH:mm");

		System.out.println("Enter the date and time (yyyy-MM-dd HH:mm): ");
		String dateTimeInput = scanner.nextLine();
		LocalDateTime dateTime = LocalDateTime.parse(dateTimeInput, formatter);

		System.out.println("Enter the source timezone (e.g., Asia/Kolkata): ");
		String sourceZone = scanner.nextLine();
		ZoneId sourceZoneId = ZoneId.of(sourceZone);

		System.out.println("Enter the target timezone (e.g., America/New_York): 
");
		String targetZone = scanner.nextLine();
		ZoneId targetZoneId = ZoneId.of(targetZone);

		ZonedDateTime sourceZonedDateTime = ZonedDateTime.of(dateTime, 
sourceZoneId);
```

```
              ZonedDateTime targetZonedDateTime = 
sourceZonedDateTime.withZoneSameInstant(targetZoneId);

              System.out.println("Converted time: " + 
targetZonedDateTime.format(formatter));
              scanner.close();
       }

}
```

Output:

```
Enter the date and time (yyyy-MM-dd HH:mm):
2024-06-08 17:27
Enter the source timezone (e.g., Asia/Kolkata):
Asia/Kolkata
Enter the target timezone (e.g., America/New_York):
America/New_York
Converted time: 2024-06-08 07:57
```