Assignment – 21

Maddula Rupa Sri Manohar
maddularupasrimanohar2001@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.

Program:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
public class CountsOfFrequency {
	public static void main(String[] args) {
		FileReader reader = null;
		BufferedReader bReader = null;
		FileWriter writer = null;
		Map<String, Integer> frequencyOfWords = new HashMap<>();
		try {
			reader = new FileReader("manohar.txt");
			bReader = new BufferedReader(reader);
			writer = new FileWriter("outmanohar.txt");
			String n;
			while ((n = bReader.readLine()) != null) {
				String[] words = n.split("\\s+");
				for (String word : words) {
					word = word.toLowerCase();
					frequencyOfWords.put(word,
				 frequencyOfWords.getOrDefault(word, 0) + 1);
				}
			}for (Map.Entry<String, Integer> word :
					 frequencyOfWords.entrySet()) {
				writer.write(word.getKey() + ": "
							+ word.getValue() + "\n");
				writer.flush();
			}
```

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

Output:

Input File:

manohar
ram
pavan
manohar
charan
ram
feroz

Output File:

manohar: 2
feroz: 1
pavan: 1
charan: 1
ram: 2

Task 2:

Serialization and Deserialization:

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

Program:

Customer Class:

```
import java.io.Serializable;

public class Customer implements Serializable {
        private int cid;
        private String cname;
        private int sal;
        public Customer(int cid, String cname, int sal) {
                super();
                this.cid = cid;
                this.cname = cname;
                this.sal = sal;
        }
        public int getCid() {
```

```
		return cid;
	}
	public void setCid(int cid) {
		this.cid = cid;
	}
	public String getCname() {
		return cname;
	}
	public void setCname(String cname) {
		this.cname = cname;
	}
	public int getSal() {
		return sal;
	}
	public void setSal(int sal) {
		this.sal = sal;
	}
	@Override
	public String toString() {
		return "Customer [cid=" + cid + ", cname=" + cname + ",
                                        sal=" + sal + "]";
	}
}
```

Main Class:

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
public class SerializeAndDeserialize {
	public static void main(String[] args) {
		Customer customer = new Customer(101, "manohar", 60000);
		try {
			FileOutputStream fileOutputStream =
                                new FileOutputStream("emp.ser");
```

```
                ObjectOutputStream objectOutputStream =
                        new ObjectOutputStream(fileOutputStream);
                objectOutputStream.writeObject(customer);
                System.out.println("Customer Object Serialized..");
                FileInputStream fileInputStream =
                                new FileInputStream("emp.ser");
                ObjectInputStream objectInputStream =
                        new ObjectInputStream(fileInputStream);
                Object object = objectInputStream.readObject();
                Customer customer2 = (Customer) object;
                System.out.println("Customer object
                                        Deserialized..");
                System.out.println("Data in Customer is: "
                                        + customer2);
            } catch (FileNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
}
```

Output:

```
Customer Object Serialized..
Customer object Deserialized..
Data in Customer is: Customer [cid=101, cname=manohar, sal=60000]
```

Task 3:

New IO (NIO):

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

Program:

```
import java.io.IOException;
import java.net.InetSocketAddress;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.channels.ServerSocketChannel;
import java.nio.channels.SocketChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import java.util.EnumSet;
public class SocketChannelServerDemo {
        public static void main(String[] args) {
                ServerSocketChannel serverSocket = null;
            SocketChannel client = null;
            FileChannel fileChannel =null;
            try {
                        serverSocket = ServerSocketChannel.open();
                        serverSocket.socket().bind(new InetSocketAddress(9012));
                    client = serverSocket.accept();
                    System.out.println("Connection Set:  " +
client.getRemoteAddress());
                    Path path = Paths.get("D:/Assignments/temp1.txt");
                    fileChannel = FileChannel.open(path,
                      EnumSet.of(StandardOpenOption.CREATE,
                        StandardOpenOption.TRUNCATE_EXISTING,
                        StandardOpenOption.WRITE)
                      );
                    ByteBuffer buffer = ByteBuffer.allocate(1024);
                    while(client.read(buffer) > 0) {
                      buffer.flip();
                      fileChannel.write(buffer);
                      buffer.clear();
                    }
                } catch (IOException e) {
                        e.printStackTrace();
```

```
            }
        finally {
            try {
                    fileChannel.close();
                    System.out.println("File Received");
                client.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
        }
    }
}
```

Output:

File Receive

Task 4:

Java Networking:

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

Program:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

public class NetworkingHTTPClient {
    public static void main(String[] args) {
        String weburl =
            "https://jsonplaceholder.typicode.com/posts/12";
        URL url;
        HttpURLConnection con = null;
        try {
            url = new URL(weburl);

            con = (HttpURLConnection) url.openConnection();

            con.setRequestMethod("GET");

            int responseCode = con.getResponseCode();
```

```
                System.out.println("Sending 'GET' request to URL:"
                                        +weburl);
                System.out.println("Response Code: "+ responseCode);

                BufferedReader in = new BufferedReader(
                        new InputStreamReader(con.getInputStream()));
                String inputLine;
                StringBuilder response = new StringBuilder();

                while ((inputLine = in.readLine()) != null) {
                        response.append(inputLine);
                }
                in.close();

                System.out.println("\nResponse Headers:");
                con.getHeaderFields().forEach((key, value) -> {
                        if (key != null) {
                                System.out.println(key + ": " + value);
                        } else {
                                System.out.println(value.get(0));
                        }
                });

                System.out.println("\nResponse Body:");
                System.out.println(response.toString());
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }

            con.disconnect();
        }
}
```

Output:

```
Sending 'GET' request to URL: https://jsonplaceholder.typicode.com/posts/12
Response Code: 200

Response Headers:
HTTP/1.1 200 OK
```

Server: [cloudflare]
X-Ratelimit-Reset: [1718536349]
Etag: [W/"fa-Kb6iNtCKEzLRQ8QyZ5zRPuKRNgs"]
Report-To: [{"group":"heroku-nel","max_age":3600,"endpoints":[{"url":"https://nel.heroku.com/reports?ts=1718536341&sid=e11707d5-02a7-43ef-b45e-2cf4d2036f7d&s=5VTnJw8ju%2B2mfpndBmbGoxNRFvKk%2B4pyUSKFC0vbML0%3D"}]}]
Access-Control-Allow-Credentials: [true]
Content-Length: [250]
Content-Type: [application/json; charset=utf-8]
X-Powered-By: [Express]
CF-RAY: [894a5d432c0a3750-MXP]
X-Ratelimit-Remaining: [997]
X-Content-Type-Options: [nosniff]
Connection: [keep-alive]
Reporting-Endpoints: [heroku-nel=https://nel.heroku.com/reports?ts=1718536341&sid=e11707d5-02a7-43ef-b45e-2cf4d2036f7d&s=5VTnJw8ju%2B2mfpndBmbGoxNRFvKk%2B4pyUSKFC0vbML0%3D]
Pragma: [no-cache]
Date: [Sun, 16 Jun 2024 11:12:21 GMT]
Via: [1.1 vegur]
Accept-Ranges: [bytes]
X-Ratelimit-Limit: [1000]
CF-Cache-Status: [MISS]
Nel: [{"report_to":"heroku-nel","max_age":3600,"success_fraction":0.005,"failure_fraction":0.05,"response_headers":["Via"]}]
Cache-Control: [max-age=43200]
Vary: [Origin, Accept-Encoding]
Expires: [-1]
alt-svc: [h3=":443"; ma=86400]
Response Body:
{

"userId": 2,
"id": 12,
"title": "in quibusdam tempore odit est dolorem",

"body": "itaque id aut magnam ……"

}

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.

Program:

```
import java.io.Serializable;

public class RequestHolder implements Serializable {
	private static final int serialVersionUID = 1;
	private double num1;
	private double num2;
	private String operation;
	public RequestHolder(double num1, double num2,
					String operation) {
		this.num1 = num1;
		this.num2 = num2;
		this.operation = operation;
	}
	public double getNum1() {
		return num1;
	}
	public double getNum2() {
		return num2;
	}
	public String getOperation() {
		return operation;
	}
	@Override
	public String toString() {
		return "RequestHolder [number1=" + num1 + ", number2="
				+ num2 + ", operation=" + operation + "]";
	}
}
```

Server:

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

```
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class TCPServer {
        public static void main(String[] args) {
                final int portNumber = 19056;
                ObjectInputStream inputStream=null;
                ObjectOutputStream outputStream=null;
                Socket clientSocket=null;
      try {
         ServerSocket serverSocket = new ServerSocket
                                        (portNumber);
         System.out.println("Server is listening on port "
                                        + portNumber);

         while (true) {
            clientSocket = serverSocket.accept();
            System.out.println("Accepted connection from client:
                                                  " + clientSocket);

            inputStream = new ObjectInputStream
                                           (clientSocket.getInputStream());
            outputStream = new ObjectOutputStream
                                           (clientSocket.getOutputStream());

               RequestHolder request = (RequestHolder)
                                                    inputStream.readObject();
               System.out.println("Received from client: "
                                                           + request);

               double result = calculateResult(request);

               outputStream.writeDouble(result);
               outputStream.flush();
               System.out.println("Sent result to client: "
                                                           + result);
         }
      } catch (IOException e) {
```

```
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }finally {
            try {
                                    inputStream.close();
                                    outputStream.close();
                    clientSocket.close();
                            } catch (IOException e) {
                                    e.printStackTrace();
                            }
        }
          }

          private static double calculateResult(RequestHolder request) {
        double result = 0;
        double num1 = request.getNum1();
        double num2 = request.getNum2();
        String operation = request.getOperation();

        switch (operation) {
            case "+":
                result = num1 + num2;
                break;
            case "-":
                result = num1 - num2;
                break;
            case "*":
                result = num1 * num2;
                break;
            case "/":
                result = num1 / num2;
                break;
            default:
                System.out.println("Unsupported operation: " + operation);
        }

        return result;
    }
}
```

Client:

```
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;

public class TCPClient {
        public static void main(String[] args) {
                final String serverHost = "localhost";
      final int serverPort = 19056;
      try {
         Socket socket = new Socket(serverHost, serverPort);
         System.out.println("Connected to server on port "
                                                    + serverPort);
         ObjectOutputStream outputStream = new ObjectOutputStream
                                            (socket.getOutputStream());
         ObjectInputStream inputStream = new ObjectInputStream
                                            (socket.getInputStream());

         try {
            RequestHolder request =new RequestHolder(2, 2, "+");

            outputStream.writeObject(request);
            outputStream.flush();
            System.out.println("Sent to server: " + request);

            double result = inputStream.readDouble();
            System.out.println("Received result from server: "
                                                    + result);
         } finally {
            outputStream.close();
            inputStream.close();
            socket.close();
         }
      } catch (IOException e) {
         e.printStackTrace();
      }
        }
}
```

Output:

Server is listening on port 19056
Connected to server on port 19056
Sent to server: RequestHolder [number1=2.0, number2=2.0, operation=+]
Received result from server: 4.0

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:

```
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;
import java.util.Scanner;

public class DateAndTine {

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter the first date in the formate
					(YYYY-MM-DD): ");
		String firstDate = sc.next();
		LocalDate date1 = LocalDate.parse(firstDate);

		System.out.println("Enter the second date in the formate
					(YYYY-MM-DD): ");
		String secondDate = sc.next();
		LocalDate date2 = LocalDate.parse(secondDate);
		long days = ChronoUnit.DAYS.between(date1, date2);
		System.out.println("First date: " + date1);
		System.out.println("Second date: "+date2);
		System.out.println("number of days: " + days);

		sc.close();
	}
}
```

Output:

Enter the first date in the formate(YYYY-MM-DD):
2023-06-15

Enter the second date in the formate(YYYY-MM-DD):
2024-06-15
First date: 2023-06-15
Second date: 2024-06-15
number of days: 366

Task 7:

Timezone:

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

Program:

```
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.time.ZonedDateTime;

public class TimeZone {

	public static void main(String[] args) {
		LocalDateTime time=LocalDateTime.now();

		ZonedDateTime zdt=ZonedDateTime.of(time,
					ZoneId.systemDefault());
		System.out.println("default time zone: "+zdt);

		ZoneId zi=ZoneId.of("America/New_York");
		ZonedDateTime zdt1=zdt.withZoneSameInstant(zi);
		System.out.println("Changed time zone: "+zdt1);
	}
}
```

Output:

```
default time zone: 2024-06-09T12:57:16.887911900+05:30[Asia/Calcutta]
Changed time zone: 2024-06-09T03:27:16.887911900-
04:00[America/New_York]
```