ASSIGNMENT: 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.

ANS:

```
import java.io.*;

import java.util.HashMap;

import java.util.Map;

import java.util.StringTokenizer;


public class JavaIO {


  public static void main(String[] args) {

    String inputFile = "input.txt";

    String outputFile = "output.txt";


    try {

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


      BufferedReader reader = new BufferedReader(new FileReader(inputFile));

      String line;

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

        StringTokenizer tokenizer = new StringTokenizer(line);

        while (tokenizer.hasMoreTokens()) {

          String word = tokenizer.nextToken().toLowerCase();

          wordCount.put(word, wordCount.getOrDefault(word, 0) + 1);
```

```
                }
            }
            reader.close();

            BufferedWriter writer = new BufferedWriter(new FileWriter(outputFile));
            for (Map.Entry<String, Integer> entry : wordCount.entrySet()) {
                writer.write(entry.getKey() + ": " + entry.getValue());
                writer.newLine();
            }
            writer.close();

            System.out.println("Word frequency count has been written to " + outputFile);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
//code_by_RUBY
```

Task 2: Serialization and Deserialization

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

ANS:

```
class Person implements Serializable {
    private static final long serialVersionUID = 1L;
    String name;

    int age;


    Person(String name, int age) {

        this.name = name;

        this.age = age;

    }


    @Override

    public String toString() {

        return "Person{name='" + name + "', age=" + age + '}';

    }

}


public class SerializationDeserialization {


    public static void main(String[] args) {

        Person person = new Person("Alice", 30);

        String filename = "person.ser";




        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(filename))) {

            out.writeObject(person);

            System.out.println("Person has been serialized: " + person);

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



        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(filename))) {
            Person deserializedPerson = (Person) in.readObject();
            System.out.println("Person has been deserialized: " + deserializedPerson);
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
//code_by_RUBY
```

Task 3: New IO (NIO)

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

ANS:

```
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.Paths;
import java.nio.file.StandardOpenOption;
```

```
}

public class NIO {

    public static void main(String[] args) {
        Path inputPath = Paths.get("input.txt");
        Path outputPath = Paths.get("output.txt");

        try (FileChannel inputChannel = FileChannel.open(inputPath, StandardOpenOption.READ);
             FileChannel outputChannel = FileChannel.open(outputPath, StandardOpenOption.CREATE, 
StandardOpenOption.WRITE)) {

            ByteBuffer buffer = ByteBuffer.allocate(1024);

            while (inputChannel.read(buffer) > 0) {
                buffer.flip();
                outputChannel.write(buffer);
                buffer.clear();
            }

            System.out.println("Content has been copied from input.txt to output.txt");

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

//code_by_RUBY

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:

```
import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.HttpURLConnection;

import java.net.URL;


public class JavaNetworking {


    public static void main(String[] args) {

        String urlString = "http://www.example.com";


        try {

            URL url = new URL(urlString);

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

            connection.setRequestMethod("GET");


            System.out.println("Response Code: " + connection.getResponseCode());

            System.out.println("Response Message: " + connection.getResponseMessage());
            System.out.println("\nHeaders:");
            connection.getHeaderFields().forEach((key, value) -> System.out.println(key + ": " + value));


            System.out.println("\nBody:");
            try (BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()))) {
                String inputLine;
                while ((inputLine = in.readLine()) != null) {
                    System.out.println(inputLine);
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
//code_by_RUBY
```

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:

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;



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

    public Operation(int number1, int number2, String operator) {
        this.number1 = number1;
        this.number2 = number2;
        this.operator = operator;
    }

    public int getNumber1() {
        return number1;
    }

    public int getNumber2() {
        return number2;
    }

    public String getOperator() {
        return operator;
    }
}

// Server class to handle client requests
public class JavaNetworking_Serialization {

    public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(12345)) {
            System.out.println("Server is listening on port 12345");
            while (true) {
                try (Socket socket = serverSocket.accept()) {
                    ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());
                    ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());

                    Operation operation = (Operation) ois.readObject();
                    int result = performOperation(operation);

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


    private static int performOperation(Operation operation) {
        int number1 = operation.getNumber1();
        int number2 = operation.getNumber2();
        String operator = operation.getOperator();


        switch (operator) {
            case "+":
                return number1 + number2;
            case "-":
                return number1 - number2;
            case "*":
                return number1 * number2;
            case "/":
                return number2 != 0 ? number1 / number2 : 0;  // Handle division by zero
            default:
                throw new IllegalArgumentException("Invalid operator: " + operator);
        }
    }
}


// Client class to send requests to the server
class OperationClient {
    public static void main(String[] args) {
        try (Socket socket = new Socket("localhost", 12345)) {
            ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());
            ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());

            Scanner scanner = new Scanner(System.in);

            System.out.println("Enter first number: ");
            int number1 = scanner.nextInt();
            System.out.println("Enter second number: ");
            int number2 = scanner.nextInt();
            System.out.println("Enter operator (+, -, *, /): ");
            String operator = scanner.next();

            Operation operation = new Operation(number1, number2, operator);
            oos.writeObject(operation);

            int result = (int) ois.readObject();
            System.out.println("Result: " + result);

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

//code_by_RUBY

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:

```
import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.time.temporal.ChronoUnit;

import java.util.Scanner;


public class Java8DateAndTime {


  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 firstDateString = scanner.nextLine();

    LocalDate firstDate = LocalDate.parse(firstDateString, formatter);


    System.out.println("Enter the second date (yyyy-MM-dd): ");

    String secondDateString = scanner.nextLine();

    LocalDate secondDate = LocalDate.parse(secondDateString, formatter);
```

```
        long daysBetween = ChronoUnit.DAYS.between(firstDate, secondDate);


        System.out.println("Number of days between " + firstDate + " and " + secondDate + " is: " +
daysBetween);
    }
}
//code_by_RUBY
```

Task 7: Timezone

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

ANS:

```
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.println("Enter date and time (yyyy-MM-dd HH:mm): ");
        String dateTimeString = scanner.nextLine();
        System.out.println("Enter source timezone (e.g., America/New_York): ");
        String sourceTimezone = scanner.nextLine();



        System.out.println("Enter target timezone (e.g., Asia/Tokyo): ");
        String targetTimezone = scanner.nextLine();



        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
        LocalDateTime localDateTime = LocalDateTime.parse(dateTimeString, formatter);



        ZonedDateTime sourceZonedDateTime = localDateTime.atZone(ZoneId.of(sourceTimezone));



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



        String formattedTargetDateTime = targetZonedDateTime.format(formatter);
        System.out.println("Converted date and time in target timezone: " + formattedTargetDateTime);
    }
}
//code_by_RUBY
```