1.A Java IO Basics

import java.io.\*;

import java.util.\*;

public class WordFrequencyCounter {

public static void main(String[] args) {

try {

BufferedReader reader = new BufferedReader(new FileReader("input.txt"));

String line;

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

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

String[] words = line.split("\\s+"); // Split by whitespace

for (String word : words) {

word = word.replaceAll("[^a-zA-Z]", "");

word = word.toLowerCase();

if (!word.isEmpty()) {

wordFrequency.put(word, wordFrequency.getOrDefault(word, 0) + 1);

}

}

}

reader.close();

FileWriter writer = new FileWriter("output.txt");

for (Map.Entry<String, Integer> entry : wordFrequency.entrySet()) {

writer.write(entry.getKey() + ": " + entry.getValue() + "\n");

}

writer.close();

System.out.println("Word frequencies written to output.txt.");

} catch (IOException e) {

e.printStackTrace();

}

}

}

2.A Serialization and Deserialization

import java.io.Serializable;

public class CustomObject implements Serializable {

}

try (FileOutputStream fileOut = new FileOutputStream("customObject.ser");

ObjectOutputStream out = new ObjectOutputStream(fileOut)) {

CustomObject obj = new CustomObject();

out.writeObject(obj);

System.out.println("Custom object serialized.");

} catch (IOException e) {

e.printStackTrace();

}

try (FileInputStream fileIn = new FileInputStream("customObject.ser");

ObjectInputStream in = new ObjectInputStream(fileIn)) {

CustomObject deserializedObj = (CustomObject) in.readObject();

System.out.println("Custom object deserialized.")

} catch (IOException | ClassNotFoundException e) {

e.printStackTrace();

}

3.A New IO

try (RandomAccessFile aFile = new RandomAccessFile("input.txt", "r");

FileChannel inChannel = aFile.getChannel()) {

long fileSize = inChannel.size();

ByteBuffer buffer = ByteBuffer.allocate((int) fileSize);

inChannel.read(buffer);

buffer.flip(); // Prepare for reading

// Now you can process the data in 'buffer'

} catch (IOException e) {

e.printStackTrace();

}

RandomAccessFile("largefile.txt", "r");

FileChannel inChannel = aFile.getChannel()) {

ByteBuffer buffer = ByteBuffer.allocate(1024);

while (inChannel.read(buffer) > 0) {

buffer.flip(); // Prepare for reading

// Process the data in 'buffer'

buffer.clear(); // Clear the buffer for the next chunk

}

} catch (IOException e) {

e.printStackTrace();

}

try (RandomAccessFile outFile = new RandomAccessFile("output.txt", "rw");

FileChannel outChannel = outFile.getChannel()) {

// Assuming you have data in 'buffer'

outChannel.write(buffer);

} catch (IOException e) {

e.printStackTrace();

}

try (RandomAccessFile aFile = new RandomAccessFile("largefile.txt", "r");

FileChannel inChannel = aFile.getChannel()) {

MappedByteBuffer buffer = inChannel.map(FileChannel.MapMode.READ\_ONLY, 0, inChannel.size());

// Process data in 'buffer'

} catch (IOException e) {

e.printStackTrace();

}

4.A Java Networking

import java.net.URI;

import java.net.http.HttpClient;

import java.net.http.HttpRequest;

import java.net.http.HttpResponse;

import java.util.concurrent.CompletableFuture;

public class SimpleHttpClientExample {

public static void main(String[] args)

HttpClient client = HttpClient.newHttpClient();

String serviceUrl = "https://example.com"; // Replace with your desired URL

HttpRequest request = HttpRequest.newBuilder()

.uri(URI.create(serviceUrl))

.POST(HttpRequest.BodyPublishers.noBody())

.build();

CompletableFuture<HttpResponse<String>> responseFuture = client.sendAsync(request, HttpResponse.BodyHandlers.ofString())

responseFuture.thenAccept(response -> {

System.out.println("Status Code: " + response.statusCode());

response.headers().map().forEach((key, value) ->

System.out.println(key + ": " + value));

System.out.println("Response Body:\n" + response.body());

});

responseFuture.join();

}

}

5.A Java Networking and Serialization

import java.io.\*;

import java.net.\*;

public class Server {

public static void main(String[] args) throws IOException {

ServerSocket serverSocket = new ServerSocket(9090);

System.out.println("Server is running and waiting for client connection...");

Socket clientSocket = serverSocket.accept();

System.out.println("Client connected!");

BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

String message = in.readLine();

String[] parts = message.split(",");

double num1 = Double.parseDouble(parts[0]);

double num2 = Double.parseDouble(parts[1]);

String operation = parts[2];

double result;

switch (operation) {

case "+":

result = num1 + num2;

break;

case "-":

result = num1 - num2;

break;

case "\*":

result = num1 \* num2;

break;

case "/":

result = num1 / num2;

break;

default:

result = Double.NaN; // Invalid operation

}

// Send the result back to the client

out.println("Result: " + result);

// Close the client socket

clientSocket.close();

// Close the server socket

serverSocket.close();

}

}

Now, let’s create the client-side code:

import java.io.\*;

import java.net.\*;

public class Client {

public static void main(String[] args) throws IOException {

Socket clientSocket = new Socket("localhost", 9090);

System.out.println("Connected to server!");

BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

double num1 = 2.0;

double num2 = 2.0;

String operation = "+";

String serializedObject = num1 + "," + num2 + "," + operation;

out.println(serializedObject);

String response = in.readLine();

System.out.println("Server response: " + response);

clientSocket.close();

}

}

6.A Date and Time API

import java.time.LocalDate;

import java.time.Period;

import java.util.Scanner;

public class DateDifferenceJava8 {

public static void main(String[] args) {

System.out.print("Insert first date (yyyy-mm-dd): ");

Scanner scanner = new Scanner(System.in);

String input1 = scanner.nextLine();

LocalDate date1 = LocalDate.parse(input1);

System.out.print("Insert second date (yyyy-mm-dd): ");

String input2 = scanner.nextLine();

LocalDate date2 = LocalDate.parse(input2);

Period period = Period.between(date1, date2);

int daysBetween = period.getDays();

System.out.println("Days between the two dates: " + daysBetween);

}

}

7.A Timezone

import java.time.LocalDateTime;

import java.time.ZoneId;

import java.time.ZonedDateTime;

import java.time.format.DateTimeFormatter;

public class TimeZoneConverter {

private static final String DATE\_FORMAT = "dd-M-yyyy hh:mm:ss a";

public static void main(String[] args) {

String dateInString = "22-1-2015 10:15:55 AM";

LocalDateTime ldt = LocalDateTime.parse(dateInString, DateTimeFormatter.ofPattern(DATE\_FORMAT));

ZoneId singaporeZoneId = ZoneId.of("Asia/Singapore");

System.out.println("TimeZone: " + singaporeZoneId);

// LocalDateTime + ZoneId = ZonedDateTime

ZonedDateTime asiaZonedDateTime = ldt.atZone(singaporeZoneId);

System.out.println("Date (Singapore): " + asiaZonedDateTime);

ZoneId newYorkZoneId = ZoneId.of("America/New\_York");

System.out.println("TimeZone: " + newYorkZoneId);

ZonedDateTime nyDateTime = asiaZonedDateTime.withZoneSameInstant(newYorkZoneId);

System.out.println("Date (New York): " + nyDateTime);

DateTimeFormatter format = DateTimeFormatter.ofPattern(DATE\_FORMAT);

System.out.println("\n---DateTimeFormatter---");

System.out.println("Date (Singapore): " + format.format(asiaZonedDateTime));

System.out.println("Date (New York): " + format.format(nyDateTime));

}

}