(CNET444) Distributed Systems Final Lab Exam	
Dr. Hussein Zangoti	Hassan Ali Sharahili - 202201918

Simple RPC with Load Balancing Word Counting Program

Task 1: Define the WordCount Interface & Implementation

```java
public interface WordCountService {
    Map<String, Integer> countWords(String textChunk) throws RemoteException;
}
```

```java
package edu.hassan;

import java.util.HashMap;
import java.util.Map;
public class WordCountServiceImpl implements WordCountService {
    @Override
    public Map<String, Integer> countWords(String textChunk) {
        Map<String, Integer> wordCount = new HashMap<>();
        String[] words = textChunk.split("\\W+");
        for (String word : words) {
            if (word.isEmpty()) continue;
            word = word.toLowerCase();
            wordCount.put(word, wordCount.getOrDefault(word, 0) + 1);
        }
        return wordCount;
    }
}
```

Task 2: Implement the RPC Server

```java
package edu.hassan;

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

public class RPC_Server {
    public static void main(String[] args) throws IOException {
        int port = Integer.parseInt(args[0]);
        WordCountServiceImpl service = new WordCountServiceImpl();
        ServerSocket serverSocket = new ServerSocket(port);
        System.out.println("Server started on port " + port);

        while (true) {
            Socket socket = serverSocket.accept();
            new Thread(() -> {
                try (
                        BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
                        PrintWriter out = new PrintWriter(socket.getOutputStream(), true)
                ) {
                    String textChunk = in.readLine();
//
                    System.out.println("[" + java.time.LocalDateTime.now() + "] A Request received from port: "
                            + socket.getInetAddress().getHostAddress() + ":" + socket.getPort());
                    System.out.println("The text request received: " +
                            (textChunk != null ? textChunk.substring(0, Math.min(100, textChunk.length())) : "null"));
                    System.out.println("");

                    Map<String, Integer> result = service.countWords(textChunk);
                    StringBuilder sb = new StringBuilder();
                    for (Map.Entry<String, Integer> entry : result.entrySet()) {
 sb.append(entry.getKey()).append(":").append(entry.getValue()).append(",");
                    }
                    out.println(sb.toString());
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }).start();
        }
    }
}
```

Task 3: Implement the Load Balancer (Listens on port 9000)

```java
package edu.hassan;

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

public class LoadBalancer {
    private static int lastServer = 0;
    private static final int[] SERVER_PORTS = {8080, 8081};

    public static void main(String[] args) throws IOException {
        ServerSocket serverSocket = new ServerSocket(9090);
        System.out.println("Load Balancer started on 9090");

        while (true) {
            Socket clientSocket = serverSocket.accept();
            new Thread(() -> {
                try (
                        BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
                        PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true)
                ) {
                    String textChunk = in.readLine();
                    int serverPort = SERVER_PORTS[lastServer];
                    lastServer = (lastServer + 1) % SERVER_PORTS.length;

                    Socket serverSocketInternal = new Socket("localhost", serverPort);
                    PrintWriter serverOut = new PrintWriter(serverSocketInternal.getOutputStream(), true);
                    BufferedReader serverIn = new BufferedReader(new InputStreamReader(serverSocketInternal.getInputStream()));

                    serverOut.println(textChunk);
                    String response = serverIn.readLine();
                    out.println(response);

                    serverSocketInternal.close();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }).start();
        }
    }
}
```

Task 4: Implement the RPC Client

```java
package edu.hassan;

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

public class RPC_Client {
    private static Map<String, Integer> parseResult(String response) {
        Map<String, Integer> map = new HashMap<>();
        if (response != null && !response.isEmpty()) {
            String[] pairs = response.split(",");
            for (String pair : pairs) {
                if (pair.isEmpty()) continue;
                String[] kv = pair.split(":");
                if (kv.length == 2)
                    map.put(kv[0], Integer.parseInt(kv[1]));
            }
        }
        return map;
    }

    private static Map<String, Integer> mergeMaps(Map<String, Integer> a, Map<String, Integer> b) {
        Map<String, Integer> result = new HashMap<>(a);
        for (Map.Entry<String, Integer> entry : b.entrySet()) {
            result.put(entry.getKey(), result.getOrDefault(entry.getKey(), 0) + entry.getValue());
        }
        return result;
    }

    public static void main(String[] args) throws IOException {
        String filePath = (args.length > 0) ? args[0] : "sample.txt";
        List<String> lines = Files.readAllLines(Paths.get(filePath));

        //Before splitting
        System.out.println("This the input file before splitting");
        System.out.println(lines);
        System.out.println("---------------------------");

        //Here split the input into two half parts
        List<String> firstHalfLines = new ArrayList<>();
        List<String> secondHalfLines = new ArrayList<>();

        for (String line : lines) {
            String[] words = line.trim().split("\\s+");
            int half = words.length / 2;
            String firstHalf = String.join(" ", Arrays.copyOfRange(words, 0, half));
            String secondHalf = String.join(" ", Arrays.copyOfRange(words, half, words.length));
            firstHalfLines.add(firstHalf);
            secondHalfLines.add(secondHalf);
        }

        String part1 = String.join(" ", firstHalfLines);
        String part2 = String.join(" ", secondHalfLines);

//        int mid = lines.size() / 2;
//        String part1 = String.join(" ", lines.subList(0, mid));
//        String part2 = String.join(" ", lines.subList(mid, lines.size()));

        System.out.println("The first Part");
        System.out.println(part1);

        System.out.println("---------------------------");

        System.out.println("The Second Part");
        System.out.println(part2);

        Map<String, Integer> result1 = sendChunk(part1, "localhost", 8080);

        Map<String, Integer> result2 = sendChunk(part2, "localhost", 8081);

        Map<String, Integer> finalResult = mergeMaps(result1, result2);

        System.out.println("Server 1 (Part 1) Results:");
        for (Map.Entry<String, Integer> entry : result1.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }

        System.out.println("Server 2 (Part 2) Results:");
        for (Map.Entry<String, Integer> entry : result2.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }


        System.out.println("All WordCount Results:");
        for (Map.Entry<String, Integer> entry : finalResult.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }
    }

    private static Map<String, Integer> sendChunk(String chunk, String host, int port) throws IOException {
        try (
                Socket socket = new Socket(host, port);
                PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
                BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()))
        ) {
            out.println(chunk);
            String response = in.readLine();
            return parseResult(response);
        }
    }
}
```

Task 5: Prepare the input file (sample.txt)

algorithm algorithm cloud cloud computing GitHub DevOps GitHub DevOps Cloud

```
drn — hassan@hassan-vm: ~/Desktop/FinalLab/RPC_and_WC_Lab — ssh hassan@172.16.186.132 — 120×30
hassan@hassan-vm:~/Desktop/FinalLab/RPC_and_WC_Lab$ cat sample.txt
algorithm algorithm cloud cloud computing GitHub DevOps GitHub DevOps Cloud
hassan@hassan-vm:~/Desktop/FinalLab/RPC_and_WC_Lab$
```

Task 6: Compile & Run the Servers + Load Balancer

$ javac -d . edu/hassan/*.java
$ java edu.hassan.RPC_Server 8080
$ java edu.hassan.RPC_Server 8081

```
drn — hassan@hassan-vm: ~/Desktop/Lab6_Project/RPC_Lab/src/main/java — ssh hassan@172.1...
hassan@hassan-vm:~/Desktop/Lab6_Project/RPC_Lab/src/main/java$ java edu.hassan.RPC_Server 8080
Server started on port 8080
```

```
drn — hassan@hassan-vm: ~/Desktop/Lab6_Project/RPC_Lab/src/main/java — ssh hassan@172.1...
hassan@hassan-vm:~/Desktop/Lab6_Project/RPC_Lab/src/main/java$ java edu.hassan.RPC_Server 8081
Server started on port 8081
```

$ java edu.hassan.LoadBalancer

```
drn — hassan@hassan-vm: ~/Desktop/FinalLab/RPC_and_WC_Lab/src/main/java — ssh hassan@1...
hassan@hassan-vm:~/Desktop/FinalLab/RPC_and_WC_Lab/src/main/java$ java edu.hassan.LoadBalancer
Load Balancer started on 9090
```

Task 7: Run the RPC Client & Show Results Output

$ java edu.hassan.RPC_Client

```
drn — hassan@hassan-vm: ~/Desktop/FinalLab/RPC_WC_Lab/src/main/java — ssh hassan@172.16.186.132 — 120×30
hassan@hassan-vm:~/Desktop/FinalLab/RPC_WC_Lab/src/main/java$ java edu.hassan.RPC_Client
This the input file before splitting
[algorithm algorithm cloud cloud computing GitHub DevOps GitHub DevOps Cloud]
---------------------------

The first Part
algorithm algorithm cloud cloud computing
---------------------------

The Second Part
GitHub DevOps GitHub DevOps Cloud
---------------------------

Server 1 (Part 1) Results:
cloud: 2
computing: 1
algorithm: 2
---------------------------
Server 2 (Part 2) Results:
cloud: 1
github: 2
devops: 2
---------------------------
All WordCount Results:
cloud: 3
computing: 1
github: 2
devops: 2
algorithm: 2
hassan@hassan-vm:~/Desktop/FinalLab/RPC_WC_Lab/src/main/java$
```

Task 8: Servers Logs

Server 1 (port 8080) logs:

```
drn — hassan@hassan-vm: ~/Desktop/Lab6_Project/RPC_Lab/src/main/java — ssh hassan@172.1...
hassan@hassan-vm:~/Desktop/Lab6_Project/RPC_Lab/src/main/java$ java edu.hassan.RPC_Server 8080
Server started on port 8080
[2025-05-11T08:04:15.279843] A Request received from port: 127.0.0.1:36188
The text request received: algorithm algorithm cloud cloud computing
```

Server 2 (port 8081) logs:

```
drn — hassan@hassan-vm: ~/Desktop/Lab6_Project/RPC_Lab/src/main/java — ssh hassan@172.1...
hassan@hassan-vm:~/Desktop/Lab6_Project/RPC_Lab/src/main/java$ java edu.hassan.RPC_Server 8081
Server started on port 8081
[2025-05-11T08:04:15.341157] A Request received from port: 127.0.0.1:53222
The text request received: GitHub DevOps GitHub DevOps Cloud
```