Dang Quang Tran 1113204
Manh Tan Doan 1113588

MESSPROTOKOL PRAKTIKUM 03

1. Einführung

Im Rahmen dieses Protokolls wird die Simulation, bei einer in Konkurs gehenden Bank um Hilfe zu bitten, beschrieben. Das Ziel besteht darin, wenn die Barreserven einer Bank unter null fallen, sendet sie eine Hilfeanfrage an andere Banken und diese können Geld zur Unterstützung leihen.

In diesem Praktikum haben wir die obige Simulation mithilfe des RPC-Modells, des Apache Thrift-Frameworks und Gradle implementiert. Thrift bietet eine Abstraktionsschicht für die Datenübertragung, Datenserialisierung und Implementierung auf Anwendungsebene. Gradle ist ein Open-Source-Build-Automatisierungstool, das vor allem zum Erstellen, Testen und Bereitstellen von Softwareprojekten verwendet wird.

2. Implementierung des RPC

2.1. Schnittstellenbeschreibung in IDL

Hier definieren wir eigene Datenstruktur, um das LoanRequest zu anzufragen und zu behandeln. Es wird verwendet, um die Kommunikation zwischen Banken einzurichten.

```
namespace java Thrift.src

service BankService {
    void requestLoan(1: LoanRequest request)
    LoanResponse processLoanRequest(1: LoanRequest request)
}
struct LoanRequest {1: double amount}
enum LoanResponse {
    APPROVED,
    REJECTED,
}
```

2.2. Implementierung der Funktion, LoanRequest zu behandeln

Hier ist die Funktion von BankThriftHandler, die dabei hilft, die Reserven der Bank zu prüfen und zu entscheiden, ob sie leihen lassen soll.

```java
    @Override
    public LoanResponse processLoanRequest(LoanRequest request) throws TException {
        if(this.bank.getReserves()> request.getAmount()) {
            System.out.println("Set Portfolio in RPC Money" + request.getAmount());
            this.bank.setReserves(this.bank.getReserves()- request.getAmount());
            bank.setBankValueUpdated(true);
            return LoanResponse.APPROVED;
        }
        return LoanResponse.REJECTED;
    }
```

3. Implementierung des Gradle

In build.gradle wird der dependencies-Block verwendet, um die für Projekt erforderlichen externen Äbhangigkeiten, die die externen Bibliotheken, Frameworks oder Module sind, anzugeben.

```
dependencies {
    implementation 'org.apache.commons:commons-lang3:3.12.0'
    implementation group: 'org.apache.thrift', name: 'libthrift', version: '0.18.1'

    // https://mvnrepository.com/artifact/org.slf4j/slf4j-api
    implementation group: 'org.slf4j', name: 'slf4j-api', version: '1.7.36'

    // https://mvnrepository.com/artifact/org.slf4j/slf4j-log4j12
    testImplementation group: 'org.slf4j', name: 'slf4j-log4j12', version: '1.7.36', ext: 'pom'

    implementation group: 'org.eclipse.paho', name: 'org.eclipse.paho.client.mqttv3', version: '1.2.5'


    implementation group: 'javax.annotation', name: 'javax.annotation-api', version: '1.3.2'
    testImplementation group: 'org.junit.jupiter', name: 'junit-jupiter-api', version: '5.9.3'
}

tasks.register('fatJar', Jar) { Jar it ->
    duplicatesStrategy = DuplicatesStrategy.EXCLUDE
    manifest {
        attributes 'Main-Class': "Bank.src.Bank"
    }
    archiveBaseName = 'Bank'
    from { configurations.compileClasspath.collect { File it ->  it.isDirectory() ? it : zipTree(it) } }{
        exclude "META-INF/*.SF"
        exclude "META-INF/*.DSA"
        exclude "META-INF/*.RSA"
    }
    with jar
}
```

4. Beispiel für eine um Ausleihen Bitte und eine erfolgreiche Rettung

Mit der Web-Schnittstelle simulieren wir das, wenn ein Kunden einen größeren Geldbetrag abhebt, als die Bankreserven hat. Daher wird diese Bank in Konkurs gehen und Hilfe von anderen Banken benötigen.

```
Bank_01          | SENDING HELPING REQUEST
WebClient        | RTT:  2 ms | Throughput:  0.6679999999999999 Mbps
WebClient        | Message from Bank: POST: OK
WebClient        |
Bank_03          | Set Portfolio in RPC Money100000.0
Bank_01          | Response: APPROVED came in 27ms
Bank_01          | 172.20.1.3 success to rescue
```

5. Performanz-Messungen

Zur Bewertung der Leistungsfähigkeit des RPC wurde die Zeitspanne zwischen der Einrichtung eines Hilfegesuchs durch eine Bank und dem Erhalt von Geldern von einer anderen Bank. Das Ergebnis dieser Messungen wird in der Konsole angezeigt.

6. Bonus

Im Rahmen des Bonus kann eine gute, durchdachte RPC-Schnittstelle implementiert werden. Außerdem wurden die Analysen ebenfalls aufgezeichnet.

7. Zusammenfassung

Zusammenfassend beschreibt das Protokoll die Implementierung einer RPC-Schnittstelle für Kommunikation zwischen Banken. Diese wurde von Apache Thrift und Gradle erstellt.

Die Hauptfunktionen sind die zur Ausleihe Anfragen und Anfragebearbeitung.

Die Performanz wird durch Messung von Zeitspanne zwischen der Einrichtung eines Hilfegesuchs durch eine Bank und dem Erhalt von Geldern von einer anderen Bank.