Die folgende Aufgabe behandelt die RPC-artige Kommunikation mittels Java Remote Method Invocation (RMI). Sollten Sie die Aufgabe *RPC mit Protocol Buffers* aus Serie 4 noch nicht abschließend bearbeitet haben, so haben Sie die Möglichkeit auszuwählen[1]:

Entweder

- Sie bearbeiten *RPC mit Protocol Buffers* aus Serie 4 weiter oder aber
- Sie bearbeiten die unten gestellte *Java Remote Method Invocation*-Aufgabe.

Weder Protocol-Buffers noch Java-RMI wird im weiteren Verlauf der Übung vorausgesetzt, so dass Sie frei in Ihrer Entscheidung sind.

# Java Remote Method Invocation (optional)

Nun wollen wir den einfachen Datenbankserver per Remote Method Invocation kommunizieren lassen. Gegeben ist folgende Schnittstelle des einfachen Datenbank–Servers aus der Vorlesung:

```
package de.fhwedel.verteilteSysteme;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface DataBase extends Remote {
        public String getRecord(int index) throws RemoteException;
        public void addRecord(int index, String record) throws RemoteException;
        public int getSize() throws RemoteException;
}
```

Für den Fall, dass bei der Operation `getRecord` der zugehörige Datensatz nicht gefunden werden kann, soll `getRecord` den Wert `null` liefern.

23. **RMI–Server**
    a) Implementieren Sie bitte den in der Vorlesung vorgestellten einfachen Datenbank–Server in der Klasse `DataBaseImpl`.
    b) Instanzieren Sie im Hauptprogramm ein `DataBaseImpl`–Objekt, machen Sie es bitte mit `UnicastRemoteObject.exportObject` remote–fähig und veröffentlichen Sie es bitte, in dem sie das zugehörige Stub–Objekt durch die Methode `writeStubToFile` in ein File schreiben.

```
private static void writeStubToFile(String fileName, Remote stub)
    throws FileNotFoundException, IOException {
        FileOutputStream fos = new FileOutputStream(fileName);
        ObjectOutputStream out = new ObjectOutputStream(fos);
        out.writeObject(stub);
        out.close();
    }
```

24. **RMI–Client**
    a) Implementieren Sie ein zum Datenbank–Server passendes Client–Programm, das die Remote–Referenz (das Stub–Objekt) aus dem in Aufgabe 23b geschriebenen File über die Methode

[1] Sind Sie schon weit vorangekommen und haben Serie 4 ganz beendet, so bearbeiten Sie bitte die Aufgaben dieser Woche.

```
private static DataBase readStubFromFile(String fileName)
    throws FileNotFoundException, IOException, ClassNotFoundException {
        /* Deserialize stub using Java Serialization */
          FileInputStream fis = new FileInputStream(fileName);
          ObjectInputStream in = new ObjectInputStream(fis);
          DataBase remoteObj = (DataBase)in.readObject();
          in.close();
          return remoteObj;
    }
```

liest und wiederum

b) die Daten der Tabelle aus Aufgabe 10 in die Datenbank schreibt und ebenfalls Datensätze für die Indizes 4103 und 4107 liest.

c) Rufen Sie bitte auch wieder die Methode `getSize()` auf, um festzustellen, ob die richtige Anzahl von Datensätzen vorhanden ist.

d) Starten Sie den Datenbank–Server im Eclipse–Debugger und lassen Sie Ihren Client (mehrfach) ablaufen. Beobachten Sie den internen Zustand (die Daten) des Datenbank–Servers.