Verteilte Systeme FS 13
Übung 5

Thomas Baumann

20. Mai 2013

1 Beschreibung

Ich habe zwei neue Interfaces eingeführt, das einte Interface *IRemoteAccount* erweitert das Interface *IAccount* um das Marker Interface *java.rmi.Remote* und dementsprechend für die Bank das Interface *IRemoteBank*. Die lokale Bank habe ich so abgeändert, so dass beiden Klassen die neu erstellten Interfaces implementieren und der Rückgabetyp von *getAccount(String number)* ebenfalls dem neuen Interface entspricht. Als letzte Änderung habe ich Zeile 45 eingefügt, damit alle Anforderungen für RMI abgedeckt sind.

Wie bereits die vorherigen Übungen, ist diese Lösung nicht Thread-Safe.

2 Code

Listing 1: Interface des Accounts für RMI

```
1  package bank;
2
3  import java.rmi.Remote;
4
5  /**
6   * This interface extends the IAccount Interface with the marker interface java.rmi.Remote
7   *
8   * @see IAccount, Remote
9   * @author Thomas Baumann
10  * @version 1.0
11  */
12 public interface IRemoteAccount extends IAccount, Remote {
13
14 }
```

Listing 2: Interface der Bank für RMI

```
1  package bank;
2
3  import java.rmi.Remote;
4
5  /**
6   * This interface extends the IBank Interface with the marker interface java.rmi.Remote
7   *
8   * @see IBank, Remote
9   * @author Thomas Baumann
10  * @version 1.0
11  */
12 public interface IRemoteBank extends IBank, Remote {
13
14 }
```

Listing 3: Lokale Bank

```
1  package bank.local;
2
3  import java.io.IOException;
4  import java.rmi.RemoteException;
5  import java.rmi.server.UnicastRemoteObject;
6  import java.util.HashMap;
7  import java.util.HashSet;
8  import java.util.Map;
9  import java.util.Set;
10
11 import bank.IAccount;
12 import bank.IBank;
13 import bank.IRemoteAccount;
14 import bank.IRemoteBank;
15 import bank.InactiveException;
16 import bank.OverdrawException;
17
18 /**
19  * Implementation of the IBank interface with full functionality and the inner class
20  * Account with the implementation of the IAccount interface.
21  *
22  * @see IBank
23  * @see IAccount
24  * @author Thomas Baumann
25  * @version 1.1
26  */
27 public class Bank implements IRemoteBank {
28
29     private Map<String, Account> accounts = new HashMap<String, Account>();
30
```

```
31      @Override
32      public Set<String> getAccountNumbers() {
33          Set<String> set = new HashSet<>();
34          for (Account a : this.accounts.values()) {
35              if (a.isActive()) {
36                  set.add(a.getNumber());
37              }
38          }
39          return set;
40      }
41
42      @Override
43      public String createAccount(String owner) throws RemoteException {
44          Account ac = new Account(owner);
45          UnicastRemoteObject.exportObject(ac, 0);
46          this.accounts.put(ac.getNumber(), ac);
47          return ac.getNumber();
48      }
49
50      @Override
51      public boolean closeAccount(String number) {
52          Account a = this.accounts.get(number);
53          if (a != null && a.isActive() && a.getBalance() == 0.0) {
54              a.active = false;
55              return true;
56          }
57          return false;
58      }
59
60      @Override
61      public IRemoteAccount getAccount(String number) {
62          return this.accounts.get(number);
63      }
64
65      @Override
66      public void transfer(IAccount from, IAccount to, double amount) throws IOException,
67              InactiveException, OverdrawException {
68          from.withdraw(amount);
69          try {
70              to.deposit(amount);
71          } catch (Exception e) {
72              from.deposit(amount);
73              throw e;
74          }
75      }
76
77      static class Account implements IRemoteAccount {
78          private static int accountNumbers;
79          private String number;
80          private String owner;
81          private double balance;
82          private boolean active = true;
83
84          Account(String owner) {
85              this.owner = owner;
86              this.number = Integer.toString(++Account.accountNumbers);
87
88          }
89
90          @Override
91          public double getBalance() {
92              return this.balance;
93          }
94
95          @Override
96          public String getOwner() {
97              return this.owner;
98          }
99
100         @Override
101         public String getNumber() {
102             return this.number;
```

```
103         }
104
105         @Override
106         public boolean isActive() {
107             return this.active;
108         }
109
110         @Override
111         public void deposit(double amount) throws InactiveException {
112             if (amount < 0) {
113                 throw new IllegalArgumentException("Amount can not be less then 0.");
114             }
115             if (!this.isActive()) {
116                 throw new InactiveException("Account is inactive.");
117             }
118             this.balance += amount;
119         }
120
121         @Override
122         public void withdraw(double amount) throws IllegalArgumentException,
123                 InactiveException, OverdrawException {
124             if (amount < 0) {
125                 throw new IllegalArgumentException("Amount can not be less then 0.");
126             }
127             if (!this.isActive()) {
128                 throw new InactiveException("Account is inactive.");
129             }
130             if (amount > this.getBalance()) {
131                 throw new OverdrawException("The account has to less amount.");
132             }
133             this.balance -= amount;
134         }
135
136     }
137 }
```

Listing 4: Server Driver

```
1  package bank.rmi;
2
3  import java.io.IOException;
4  import java.rmi.Naming;
5  import java.rmi.RemoteException;
6  import java.rmi.registry.LocateRegistry;
7  import java.rmi.server.UnicastRemoteObject;
8
9  import bank.IRemoteBank;
10 import bank.IServerDriver;
11 import bank.StartServer;
12 import bank.local.Bank;
13
14 public class ServerDriver implements IServerDriver {
15
16     private IRemoteBank bank = new Bank();
17
18     @Override
19     public void start(String[] args) throws IOException {
20         if (args.length < 2) {
21             System.out.println("Usage: java " + StartServer.class.getName() + " "
22                     + ServerDriver.class.getName() + " <host> <portnumber>");
23             System.exit(1);
24         }
25         String host = args[0];
26         int port = 0;
27         try {
28             port = Integer.parseInt(args[1]);
29         } catch (NumberFormatException e) {
30             System.out.println("Port must be a number");
31             System.exit(1);
32         }
```

```
33
34          try {
35              LocateRegistry.createRegistry(port);
36          } catch (RemoteException e) {
37              System.out.println("registry could not be exported");
38              System.exit(1);
39          }
40
41          UnicastRemoteObject.exportObject(this.bank, 0);
42
43          Naming.rebind("rmi://" + host + ":" + port + "/Bank", this.bank);
44      }
45
46  }
```

Listing 5: Client Driver

```
1  package bank.rmi;
2
3  import java.io.IOException;
4  import java.rmi.Naming;
5  import java.rmi.NotBoundException;
6
7  import bank.IBank;
8  import bank.IBankDriver;
9  import bank.IRemoteBank;
10 import bank.StartClient;
11
12 public class ClientDriver implements IBankDriver {
13
14     private IRemoteBank bank;
15
16     @Override
17     public void connect(String[] args) throws IOException {
18         if (args.length < 2) {
19             System.out.println("Usage: java " + StartClient.class.getName() + " "
20                     + ClientDriver.class.getName() + " <host> <port>");
21             System.exit(1);
22         }
23         try {
24             bank = (IRemoteBank)Naming.lookup("rmi://"+args[0]+":"+args[1]+"/Bank");
25         } catch (NotBoundException e) {
26             System.out.println("Bank can not be found");
27             System.exit(1);
28         }
29     }
30
31     @Override
32     public void disconnect() throws IOException {
33         bank = null;
34     }
35
36
37     @Override
38     public IBank getBank() {
39         return bank;
40     }
41
42 }
```