Verteilte Systeme FS 13
Übung 2

Thomas Baumann

16. März 2013

Inhaltsverzeichnis

1 Beschreibung

Im Vergleich zur ersten Übung habe ich nur noch ein Antwort Objekt für Rückgabewerte und für die vier Exception jeweils ein Antwort Objekt.

2 Server

Listing 1: Servlet Konfiguration

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                             http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
         version="3.0">

    <servlet>
        <servlet-name>BankServerServlet</servlet-name>
        <servlet-class>bank.servlet.ServerServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet>
        <servlet-name>BankWebsiteServlet</servlet-name>
        <servlet-class>bank.servlet.WebsiteServlet</servlet-class>
        <load-on-startup>2</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>BankServerServlet</servlet-name>
        <url-pattern>/java.html</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>BankWebsiteServlet</servlet-name>
        <url-pattern>/index.html</url-pattern>
    </servlet-mapping>

</web-app>
```

Listing 2: Server Servlet

```
1  package bank.servlet;
2
3  import java.io.IOException;
4  import java.io.ObjectInputStream;
5  import java.io.ObjectOutputStream;
6
7  import javax.servlet.ServletException;
8  import javax.servlet.http.HttpServlet;
9  import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 import bank.IBank;
13 import bank.communication.answer.IAnswer;
14 import bank.communication.request.IRequest;
15 import bank.local.Bank;
16
17 /**
18  * This class provides an implementation of a {@link HttpServlet} to handle post requests
19  * from the bank GUI. The content type of the post request must be set to
20  * "application/x-java-serialized-object" otherwise a 400 error will be send. A
21  *
22  * @see HttpServlet
23  * @author Thomas Baumann
24  * @version 1.0
25  */
26 public class ServerServlet extends HttpServlet {
27     private final String CONTENT_TYPE = "application/x-java-serialized-object";
28     private IBank bank = new Bank();
29
30     @Override
31     protected void doPost(HttpServletRequest request, HttpServletResponse response)
32             throws IOException {
33         if (this.CONTENT_TYPE.equals(request.getContentType())) {
34             try {
35                 ObjectInputStream oin = new ObjectInputStream(request.getInputStream());
```

```
                Object o = oin.readObject();
                IAnswer<?> answer = ((IRequest) o).handleRequest(this.bank);

                response.setContentType("application/x-java-serialized-object");
                ObjectOutputStream oout = new ObjectOutputStream(
                        response.getOutputStream());
                oout.writeObject(answer);
            } catch (Exception e) {
                response.sendError(400, "Bad Request");
            }
        } else {
            response.sendError(400, "Bad Request");
        }
    }

    @Override
    public void init() throws ServletException {
        super.init();
        this.getServletContext().setAttribute("bank", this.bank);
    }
}
```

Listing 3: Server Servlet

```
package bank.servlet;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Set;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import bank.IAccount;
import bank.IBank;
import bank.local.Bank;

/**
 * This class provides an implementation of a {@link HttpServlet} to handle get and post
 * requests which shows all active accounts.
 *
 * @see HttpServlet
 * @author Thomas Baumann
 * @version 1.0
 */
public class WebsiteServlet extends HttpServlet {

    private final String PASSWORD = "bank";
    private IBank bank;

    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println(" <head>");
        out.println("  <title>Bank Server</title>");
        out.println(" </head>");
        out.println(" <body>");
        out.println("  <h1>Bankkonten</h1>");
        String password = request.getParameter("password");
        if (this.PASSWORD.equals(password)) {
            if (this.bank != null) {
                this.showAccounts(out);
            } else {
                out.println("Die Konten k&ouml;nnen nicht abgerufen werden.");
            }
```

```
47          } else {
48              out.println("    <form name=\"form\" action=\"" + request.getRequestURI()
49                      + "\" method=\"post\">");
50              out.println("     <label for=\"password\">Passwort:</label><br>");
51              out.println("     <input type=\"password\" id=\"password\" name=\"password\"><br
                    >");
52              out.println("     <br>");
53              out.println("     <input type=\"submit\" value=\"Anmelden\">");
54              out.println("    </form>");
55          }
56          out.println(" </body>");
57          out.println("</html>");
58      }
59
60      @Override
61      protected void doPost(HttpServletRequest request, HttpServletResponse response)
62              throws IOException {
63          this.doGet(request, response);
64      }
65
66      /**
67       * Prints all accounts to the specified PrintWriter in a table.
68       *
69       * @param out PrintWriter to print accounts
70       */
71      private void showAccounts(PrintWriter out) {
72          try {
73              Set<String> numbers = this.bank.getAccountNumbers();
74              if (numbers.size() > 0) {
75                  out.println("  <table border=\"1px\">");
76                  for (String number : numbers) {
77                      IAccount account = this.bank.getAccount(number);
78                      out.println("    <tr>");
79                      out.println("     <td width=\"50px\">" + account.getNumber() + "</td>");
80                      out.println("     <td width=\"200px\">" + account.getOwner() + "</td>");
81                      out.println("     <td width=\"100px\">" + account.getBalance()
82                              + "</td>");
83                      out.println("    </tr>");
84                  }
85                  out.println("  </table>");
86              } else {
87                  out.println("  <p>Es sind keine Konten vorhanden.</p");
88              }
89          } catch (IOException e) {
90              // ignore, should not happen
91          }
92      }
93
94      @Override
95      public void init() throws ServletException {
96          super.init();
97          this.bank = (Bank) this.getServletContext().getAttribute("bank");
98      }
99  }
```

3 Client

Listing 4: Abstract Client Driver

```
1  package bank.communication;
2
3  import java.io.IOException;
4  import java.util.Set;
5
6  import bank.IAccount;
7  import bank.IBank;
8  import bank.IBankDriver;
9  import bank.InactiveException;
10 import bank.OverdrawException;
11 import bank.communication.request.CloseAccountRequest;
12 import bank.communication.request.CreateAccountRequest;
13 import bank.communication.request.DepositRequest;
14 import bank.communication.request.GetAccountNumbersRequest;
15 import bank.communication.request.GetAccountRequest;
16 import bank.communication.request.GetBalanceRequest;
17 import bank.communication.request.GetOwnerRequest;
18 import bank.communication.request.IRequest;
19 import bank.communication.request.IsActiveRequest;
20 import bank.communication.request.TransferRequest;
21 import bank.communication.request.WithdrawRequest;
22
23 /**
24  * This abstract class provides an implementation of the IBankDriver interface with the
25  * additional method handleMessage(...) to send and receive objects.
26  *
27  * @see IBankDriver
28  * @author Thomas Baumann
29  * @version 1.1
30  */
31 public abstract class AbstractClientDriver implements IBankDriver {
32
33     protected IBank bank;
34
35     @Override
36     public final IBank getBank() {
37         return this.bank;
38     }
39
40     /**
41      * Sends a request object and receives afterwards the answer object and returns the
42      * data from the answer object.
43      *
44      * @param r the request to write
45      * @return answer object
46      * @throws ClassNotFoundException Class of a read object cannot be found
47      * @throws IOException When an IO problem occurs
48      * @throws IllegalArgumentException When answer is an IllegalArgumentException
49      * @throws OverdrawException When answer is an OverdrawException
50      * @throws InactiveException When answer is an InactiveException
51      */
52     protected abstract <T> T handleMessage(IRequest r) throws ClassNotFoundException,
53             IOException, IllegalArgumentException, OverdrawException, InactiveException,
54             ClassCastException;
55
56     protected final class SocketBank implements IBank {
57
58         // public constructor for visibility
59         public SocketBank() {}
60
61         @Override
62         public Set<String> getAccountNumbers() throws IOException {
63             try {
64                 return AbstractClientDriver.this
65                         .handleMessage(new GetAccountNumbersRequest());
66             } catch (ClassNotFoundException | IllegalArgumentException
67                     | OverdrawException | InactiveException | ClassCastException e) {
```

```
68                  throw new IOException(e);
69              }
70          }
71
72          @Override
73          public String createAccount(String owner) throws IOException {
74              try {
75                  return AbstractClientDriver.this.handleMessage(new CreateAccountRequest(
76                          owner));
77              } catch (ClassNotFoundException | IllegalArgumentException
78                      | OverdrawException | InactiveException | ClassCastException e) {
79                  throw new IOException(e);
80              }
81          }
82
83          @Override
84          public boolean closeAccount(String number) throws IOException {
85              try {
86                  return AbstractClientDriver.this.handleMessage(new CloseAccountRequest(
87                          number));
88              } catch (ClassNotFoundException | IllegalArgumentException
89                      | OverdrawException | InactiveException | ClassCastException e) {
90                  throw new IOException(e);
91              }
92          }
93
94          @Override
95          public IAccount getAccount(String number) throws IOException {
96              try {
97                  if (AbstractClientDriver.this
98                          .handleMessage(new GetAccountRequest(number))) {
99                      return new SocketAccount(number);
100                 } else {
101                     return null;
102                 }
103             } catch (ClassNotFoundException | IllegalArgumentException
104                     | OverdrawException | InactiveException | ClassCastException e) {
105                 throw new IOException(e);
106             }
107         }
108
109         @Override
110         public void transfer(IAccount from, IAccount to, double amount)
111                 throws IOException, IllegalArgumentException, OverdrawException,
112                 InactiveException {
113             try {
114                 AbstractClientDriver.this.handleMessage(new TransferRequest(from
115                         .getNumber(), to.getNumber(), amount));
116             } catch (ClassNotFoundException | ClassCastException e) {
117                 throw new IOException(e);
118             }
119         }
120
121     }
122
123     protected final class SocketAccount implements IAccount {
124         private String number;
125
126         public SocketAccount(String number) {
127             this.number = number;
128         }
129
130         @Override
131         public double getBalance() throws IOException {
132             try {
133                 return AbstractClientDriver.this.handleMessage(new GetBalanceRequest(
134                         this.number));
135             } catch (ClassNotFoundException | IllegalArgumentException
136                     | OverdrawException | InactiveException | ClassCastException e) {
137                 throw new IOException(e);
138             }
139         }
```

```
140
141         @Override
142         public String getOwner() throws IOException {
143             try {
144                 return AbstractClientDriver.this.handleMessage(new GetOwnerRequest(
145                         this.number));
146             } catch (ClassNotFoundException | IllegalArgumentException
147                     | OverdrawException | InactiveException | ClassCastException e) {
148                 throw new IOException(e);
149             }
150         }
151
152         @Override
153         public String getNumber() {
154             return this.number;
155         }
156
157         @Override
158         public boolean isActive() throws IOException {
159             try {
160                 return AbstractClientDriver.this.handleMessage(new IsActiveRequest(
161                         this.number));
162             } catch (ClassNotFoundException | IllegalArgumentException
163                     | OverdrawException | InactiveException | ClassCastException e) {
164                 throw new IOException(e);
165             }
166         }
167
168         @Override
169         public void deposit(double amount) throws IllegalArgumentException,
170                 InactiveException, IOException {
171             try {
172                 AbstractClientDriver.this.handleMessage(new DepositRequest(this.number,
173                         amount));
174             } catch (ClassNotFoundException | OverdrawException | ClassCastException e) {
175                 throw new IOException(e);
176             }
177         }
178
179         @Override
180         public void withdraw(double amount) throws IllegalArgumentException,
181                 InactiveException, OverdrawException, IOException {
182             try {
183                 AbstractClientDriver.this.handleMessage(new WithdrawRequest(this.number,
184                         amount));
185             } catch (ClassNotFoundException | ClassCastException e) {
186                 throw new IOException(e);
187             }
188         }
189     }
190
191 }
```

Listing 5: Servlet Client Driver

```
1  package bank.servlet;
2
3  import java.io.IOException;
4  import java.io.ObjectInputStream;
5  import java.io.ObjectOutputStream;
6  import java.net.HttpURLConnection;
7  import java.net.URL;
8
9  import bank.InactiveException;
10 import bank.OverdrawException;
11 import bank.StartClient;
12 import bank.communication.AbstractClientDriver;
13 import bank.communication.answer.IAnswer;
14 import bank.communication.request.IRequest;
15
```

```java
/**
 * This class provides an implementation of the AbstractClientDriver with HTTP.
 *
 * @see AbstractClientDriver
 * @author Thomas Baumann
 * @version 1.0
 */
public final class ClientDriver extends AbstractClientDriver {

    private URL url;

    @Override
    public void connect(String[] args) throws IOException {
        if (args.length < 1) {
            System.out.println("Usage: java " + StartClient.class.getName() + " "
                    + ClientDriver.class.getName() + " <server>");
            System.exit(1);
        }
        this.url = new URL(args[0]);
        this.bank = new SocketBank();
    }

    @Override
    public void disconnect() throws IOException {
        this.bank = null;
    }

    @SuppressWarnings("unchecked")
    @Override
    protected <T> T handleMessage(IRequest request) throws ClassNotFoundException,
            IOException, IllegalArgumentException, OverdrawException, InactiveException,
            ClassCastException {
        HttpURLConnection c = (HttpURLConnection) this.url.openConnection();
        c.setRequestMethod("POST");
        c.setRequestProperty("Content-Type", "application/x-java-serialized-object");
        c.setDoOutput(true);
        c.setDoInput(true);
        c.connect();

        // write object
        ObjectOutputStream oout = new ObjectOutputStream(c.getOutputStream());
        oout.writeObject(request);
        oout.flush();
        oout.close();

        // read object
        ObjectInputStream oin = new ObjectInputStream(c.getInputStream());
        Object o = oin.readObject();
        oin.close();

        c.disconnect();
        return ((IAnswer<T>) o).getData();

    }
}
```

4 Kommunikation Objekte

4.1 Anfragen

Listing 6: Interface für Request Objekte

```java
package bank.communication.request;

import java.io.Serializable;

import bank.IBank;
import bank.communication.answer.IAnswer;

/**
 * This interface must be used as request object for the socket communication from the
 * client to the server.
 *
 * @author Thomas Baumann
 * @version 1.1
 */
public interface IRequest extends Serializable {

    /**
     * Handles the request with the specified bank.
     *
     * @param b Bank to hande the request
     * @return answer object to send back
     */
    public IAnswer<?> handleRequest(IBank b);

}
```

Listing 7: Anfrage Objekt um Konto zu eröffnen

```java
package bank.communication.request;

import java.io.IOException;

import bank.IBank;
import bank.communication.answer.Answer;
import bank.communication.answer.IAnswer;
import bank.communication.answer.IOExceptionAnswer;

/**
 * This class provides a create account request.
 *
 * @see IRequest
 * @author Thomas Baumann
 * @version 1.1
 */
public class CreateAccountRequest implements IRequest {

    private String owner;

    public CreateAccountRequest(String owner) {
        this.owner = owner;
    }

    @Override
    public IAnswer<?> handleRequest(IBank b) {
        try {
            String s = b.createAccount(this.owner);
            return new Answer<String>(s);
        } catch (IOException e) {
            return new IOExceptionAnswer(e);
        }
    }

```

```
35 }
```

Listing 8: Anfrage Objekt um Konto zu schliessen

```
1  package bank.communication.request;
2
3  import java.io.IOException;
4
5  import bank.IBank;
6  import bank.communication.answer.Answer;
7  import bank.communication.answer.IAnswer;
8  import bank.communication.answer.IOExceptionAnswer;
9
10 /**
11  * This class provides a close account request.
12  *
13  * @see IRequest
14  * @author Thomas Baumann
15  * @version 1.1
16  */
17 public class CloseAccountRequest implements IRequest {
18
19     private String number;
20
21     public CloseAccountRequest(String number) {
22         this.number = number;
23     }
24
25     @Override
26     public IAnswer<?> handleRequest(IBank b) {
27         try {
28             boolean ans = b.closeAccount(this.number);
29             return new Answer<Boolean>(ans);
30         } catch (IOException e) {
31             return new IOExceptionAnswer(e);
32         }
33     }
34
35 }
```

Listing 9: Anfrage Objekt um Konto abzufragen

```
1  package bank.communication.request;
2
3  import java.io.IOException;
4
5  import bank.IBank;
6  import bank.communication.answer.Answer;
7  import bank.communication.answer.IAnswer;
8  import bank.communication.answer.IOExceptionAnswer;
9
10 /**
11  * This class provides a get account request.
12  *
13  * @see IRequest
14  * @author Thomas Baumann
15  * @version 1.1
16  */
17 public class GetAccountRequest implements IRequest {
18
19     private String number;
20
21     public GetAccountRequest(String number) {
22         this.number = number;
23     }
24
25     @Override
26     public IAnswer<?> handleRequest(IBank b) {
27         try {
```

```
28              return new Answer<Boolean>(b.getAccount(this.number) != null);
29          } catch (IOException e) {
30              return new IOExceptionAnswer(e);
31          }
32      }
33
34  }
```

Listing 10: Anfrage Objekt um Kontonummer abzufragen

```
1  package bank.communication.request;
2
3  import java.io.IOException;
4  import java.util.Set;
5
6  import bank.IBank;
7  import bank.communication.answer.Answer;
8  import bank.communication.answer.IAnswer;
9  import bank.communication.answer.IOExceptionAnswer;
10
11 /**
12  * This class provides a get account numbers request.
13  *
14  * @see IRequest
15  * @author Thomas Baumann
16  * @version 1.1
17  */
18 public class GetAccountNumbersRequest implements IRequest {
19
20      @Override
21      public IAnswer<?> handleRequest(IBank b) {
22          try {
23              Set<String> s = b.getAccountNumbers();
24              return new Answer<Set<String>>(s);
25          } catch (IOException e) {
26              return new IOExceptionAnswer(e);
27          }
28
29      }
30
31 }
```

Listing 11: Anfrage Objekt um Geld zu transferieren

```
1  package bank.communication.request;
2
3  import java.io.IOException;
4
5  import bank.IAccount;
6  import bank.IBank;
7  import bank.InactiveException;
8  import bank.OverdrawException;
9  import bank.communication.answer.Answer;
10 import bank.communication.answer.IAnswer;
11 import bank.communication.answer.IOExceptionAnswer;
12 import bank.communication.answer.IllegalArgumentExceptionAnswer;
13 import bank.communication.answer.InactiveExceptionAnswer;
14 import bank.communication.answer.OverdrawExceptionAnswer;
15
16 /**
17  * This class provides a transfer request.
18  *
19  * @see IRequest
20  * @author Thomas Baumann
21  * @version 1.1
22  */
23 public class TransferRequest implements IRequest {
24
25      private String numberFrom;
```

```
26      private String numberTo;
27      private Double amount;
28
29      public TransferRequest(String numberFrom, String numberTo, Double amount) {
30          this.numberFrom = numberFrom;
31          this.numberTo = numberTo;
32          this.amount = amount;
33      }
34
35      @Override
36      public IAnswer<?> handleRequest(IBank b) {
37          try {
38              IAccount f = b.getAccount(this.numberFrom);
39              IAccount t = b.getAccount(this.numberTo);
40              b.transfer(f, t, this.amount);
41              return new Answer<Object>(null);
42          } catch (IllegalArgumentException e) {
43              return new IllegalArgumentExceptionAnswer(e);
44          } catch (IOException e) {
45              return new IOExceptionAnswer(e);
46          } catch (OverdrawException e) {
47              return new OverdrawExceptionAnswer(e);
48          } catch (InactiveException e) {
49              return new InactiveExceptionAnswer(e);
50          }
51      }
52
53  }
```

Listing 12: Anfrage Objekt um Kontobesitzer abzufragen

```
1  package bank.communication.request;
2
3  import java.io.IOException;
4
5  import bank.IBank;
6  import bank.communication.answer.Answer;
7  import bank.communication.answer.IAnswer;
8  import bank.communication.answer.IOExceptionAnswer;
9
10 /**
11  * This class provides a get owner request for an account.
12  *
13  * @see IRequest
14  * @author Thomas Baumann
15  * @version 1.1
16  */
17 public class GetOwnerRequest implements IRequest {
18
19      private String number;
20
21      public GetOwnerRequest(String number) {
22          this.number = number;
23      }
24
25      @Override
26      public IAnswer<?> handleRequest(IBank b) {
27          try {
28              String owner = b.getAccount(this.number).getOwner();
29              return new Answer<String>(owner);
30          } catch (IOException e) {
31              return new IOExceptionAnswer(e);
32          }
33      }
34
35  }
```

Listing 13: Anfrage Objekt um Kontostand abzufragen

```java
package bank.communication.request;

import java.io.IOException;

import bank.IBank;
import bank.communication.answer.Answer;
import bank.communication.answer.IAnswer;
import bank.communication.answer.IOExceptionAnswer;

/**
 * This class provides a get balance request for an account.
 *
 * @see IRequest
 * @author Thomas Baumann
 * @version 1.1
 */
public class GetBalanceRequest implements IRequest {

    private String number;

    public GetBalanceRequest(String number) {
        this.number = number;
    }

    @Override
    public IAnswer<?> handleRequest(IBank b) {
        try {
            Double balance = b.getAccount(this.number).getBalance();
            return new Answer<Double>(balance);
        } catch (IOException e) {
            return new IOExceptionAnswer(e);
        }
    }

}
```

Listing 14: Anfrage Objekt für Aktiv/Inaktiv Zustand

```java
package bank.communication.request;

import java.io.IOException;

import bank.IBank;
import bank.communication.answer.Answer;
import bank.communication.answer.IAnswer;
import bank.communication.answer.IOExceptionAnswer;

/**
 * This class provides an is active request for an account.
 *
 * @see IRequest
 * @author Thomas Baumann
 * @version 1.1
 */
public class IsActiveRequest implements IRequest {

    private String number;

    public IsActiveRequest(String number) {
        this.number = number;
    }

    @Override
    public IAnswer<?> handleRequest(IBank b) {
        try {
            boolean ans = b.getAccount(this.number).isActive();
            return new Answer<Boolean>(ans);
        } catch (IOException e) {
            return new IOExceptionAnswer(e);
        }
```

```
33      }
34
35  }
```

Listing 15: Anfrage Objekt um Geld abzuheben

```
1  package bank.communication.request;
2
3  import java.io.IOException;
4
5  import bank.IBank;
6  import bank.InactiveException;
7  import bank.communication.answer.Answer;
8  import bank.communication.answer.IAnswer;
9  import bank.communication.answer.IOExceptionAnswer;
10 import bank.communication.answer.IllegalArgumentExceptionAnswer;
11 import bank.communication.answer.InactiveExceptionAnswer;
12
13 /**
14  * This class provides a deposit request for an account.
15  *
16  * @see IRequest
17  * @author Thomas Baumann
18  * @version 1.1
19  */
20 public class DepositRequest implements IRequest {
21
22     private String number;
23     private Double amount;
24
25     public DepositRequest(String number, Double amount) {
26         this.number = number;
27         this.amount = amount;
28     }
29
30     @Override
31     public IAnswer<?> handleRequest(IBank b) {
32         try {
33             b.getAccount(this.number).deposit(this.amount);
34             return new Answer<Object>(null);
35         } catch (IOException e) {
36             return new IOExceptionAnswer(e);
37         } catch (IllegalArgumentException e) {
38             return new IllegalArgumentExceptionAnswer(e);
39         } catch (InactiveException e) {
40             return new InactiveExceptionAnswer(e);
41         }
42
43     }
44
45 }
```

Listing 16: Anfrage Objekt um Geld einzuzahlen

```
1  package bank.communication.request;
2
3  import java.io.IOException;
4
5  import bank.IBank;
6  import bank.InactiveException;
7  import bank.OverdrawException;
8  import bank.communication.answer.Answer;
9  import bank.communication.answer.IAnswer;
10 import bank.communication.answer.IOExceptionAnswer;
11 import bank.communication.answer.IllegalArgumentExceptionAnswer;
12 import bank.communication.answer.InactiveExceptionAnswer;
13 import bank.communication.answer.OverdrawExceptionAnswer;
14
15 /**
```

```
 * This class provides a withdraw request for an account.
 *
 * @see IRequest
 * @author Thomas Baumann
 * @version 1.1
 */
public class WithdrawRequest implements IRequest {

    private String number;
    private Double amount;

    public WithdrawRequest(String number, Double amount) {
        this.number = number;
        this.amount = amount;
    }

    @Override
    public IAnswer<?> handleRequest(IBank b) {
        try {
            b.getAccount(this.number).withdraw(this.amount);
            return new Answer<Object>(null);
        } catch (IOException e) {
            return new IOExceptionAnswer(e);
        } catch (IllegalArgumentException e) {
            return new IllegalArgumentExceptionAnswer(e);
        } catch (OverdrawException e) {
            return new OverdrawExceptionAnswer(e);
        } catch (InactiveException e) {
            return new InactiveExceptionAnswer(e);
        }
    }

}
```

4.2 Antworten

Listing 17: Interface für Antwort Objekte

```
package bank.communication.answer;

import java.io.IOException;
import java.io.Serializable;

import bank.InactiveException;
import bank.OverdrawException;

/**
 * This interface must be used as answer object for the socket communication from the
 * server to the client.
 *
 * @author Thomas Baumann
 * @version 1.0
 * @param <T>
 */
public interface IAnswer<T> extends Serializable {

    /**
     * Returns an object or throws an exception.
     *
     * @return Returns the answer
     * @throws IllegalArgumentException When answer is an IllegalArgumentException
     * @throws IOException When an IO problem occurs
     * @throws OverdrawException When answer is an OverdrawException
     * @throws InactiveException When answer is an InactiveException
     */
    public T getData() throws IllegalArgumentException, IOException, OverdrawException,
            InactiveException;

}
```

Listing 18: Allgemeines Antwort Ojekt

```java
package bank.communication.answer;

/**
 * This class provides a close account answer. It includes an boolean which specifies, if
 * the close account request was successful or not.
 *
 * @see IAnswer
 * @author Thomas Baumann
 * @version 1.0
 */
public class Answer<T> implements IAnswer<T> {

    private T value;

    public Answer(T value) {
        this.value = value;
    }

    @Override
    public T getData() {
        return this.value;
    }

}
```

Listing 19: Antwort Objekt für IllegalArgumentException

```java
package bank.communication.answer;

/**
 * This class provides a IllegalArgumentException answer. It includes an exception of the
 * type illegal argument. When the getData method will be called the exception will be
 * thrown.
 *
 * @see IAnswer
 * @author Thomas Baumann
 * @version 1.0
 */
public class IllegalArgumentExceptionAnswer implements IAnswer<Object> {

    private IllegalArgumentException e;

    public IllegalArgumentExceptionAnswer(IllegalArgumentException e) {
        this.e = e;
    }

    @Override
    public Object getData() throws IllegalArgumentException {
        throw this.e;
    }

}
```

Listing 20: Antwort Objekt für InactiveException

```java
package bank.communication.answer;

import bank.InactiveException;

/**
 * This class provides a InactiveException answer. It includes an exception of the type
 * inactive. When the getData method will be called the exception will be thrown.
 *
 * @see IAnswer
 * @author Thomas Baumann
 * @version 1.0
 */
```

```
13 public class InactiveExceptionAnswer implements IAnswer<Object> {
14
15     private InactiveException e;
16
17     public InactiveExceptionAnswer(InactiveException e) {
18         this.e = e;
19     }
20
21     @Override
22     public Object getData() throws InactiveException {
23         throw this.e;
24     }
25
26 }
```

Listing 21: Antwort Objekt für IOException

```
1  package bank.communication.answer;
2
3  import java.io.IOException;
4
5  /**
6   * This class provides a IOException answer. It includes an exception of the type IO. When
7   * the getData method will be called the exception will be thrown.
8   *
9   * @see IAnswer
10  * @author Thomas Baumann
11  * @version 1.0
12  */
13 public class IOExceptionAnswer implements IAnswer<Object> {
14
15     private IOException e;
16
17     public IOExceptionAnswer(IOException e) {
18         this.e = e;
19     }
20
21     @Override
22     public Object getData() throws IOException {
23         throw this.e;
24     }
25
26 }
```

Listing 22: Antwort Objekt für OverdrawException

```
1  package bank.communication.answer;
2
3  import bank.OverdrawException;
4
5  /**
6   * This class provides a OverdrawException answer. It includes an exception of the type
7   * Overdraw. When the getData method will be called the exception will be thrown.
8   *
9   * @see IAnswer
10  * @author Thomas Baumann
11  * @version 1.0
12  */
13 public class OverdrawExceptionAnswer implements IAnswer<Object> {
14
15     private OverdrawException e;
16
17     public OverdrawExceptionAnswer(OverdrawException e) {
18         this.e = e;
19     }
20
21     @Override
22     public Object getData() throws OverdrawException {
23         throw this.e;
```

```
24        }
25
26    }
```