Verteilte Systeme FS 13
Übung 2

Thomas Baumann

15. 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

```java
package bank.servlet;

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

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

import bank.IAccount;
import bank.IBank;
import bank.communication.answer.IAnswer;
import bank.communication.request.IRequest;
import bank.local.Bank;

/**
 * This class provides an implementation of a {@link HttpServlet} to handle post requests
 * with a bank and returns all active bank accounts for a get request.
 *
 * @see HttpServlet
 * @author Thomas Baumann
 * @version 1.0
 */
public class ServerServlet extends HttpServlet {

    private IBank bank = new Bank();

    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println(" <body>");
        out.println("  <h1>Bankkonten</h1>");
        Set<String> numbers = this.bank.getAccountNumbers();
        if (numbers.size() > 0) {
            out.println("  <table border=\"1px\">");
            for (String number : numbers) {
                IAccount account = this.bank.getAccount(number);
                out.println("   <tr>");
                out.println("    <td width=\"50px\">" + account.getNumber() + "</td>");
                out.println("    <td width=\"200px\">" + account.getOwner() + "</td>");
                out.println("    <td width=\"50px\">" + account.getBalance() + "</td>");
                out.println("   </tr>");
            }
            out.println("  </table>");
        } else {
            out.println("  <p>Es sind keine Konten vorhanden.</p>");
        }
        out.println(" </body>");
        out.println("</html>");
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws IOException {
        if (request.getContentType().equals("application/x-java-serialized-object")) {
            try {
                ObjectInputStream oin = new ObjectInputStream(request.getInputStream());
                Object o = oin.readObject();
                IAnswer<?> answer = ((IRequest) o).handleRequest(this.bank);

                response.setContentType("application/x-java-serialized-object");
```

```
68              ObjectOutputStream oout = new ObjectOutputStream(
69                      response.getOutputStream());
70              oout.writeObject(answer);
71          } catch (Exception e) {
72              System.err.println("invalid object arrived");
73              e.printStackTrace();
74          }
75      } else {
76          this.doGet(request, response);
77      }
78    }
79
80 }
```

Listing 2: Servlet Konfiguration

```xml
<?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>
    </servlet>

    <servlet-mapping>
        <servlet-name>BankServerServlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

</web-app>
```

3 Client

```java
package bank.communication;

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

import bank.IAccount;
import bank.IBank;
import bank.IBankDriver;
import bank.InactiveException;
import bank.OverdrawException;
import bank.communication.request.CloseAccountRequest;
import bank.communication.request.CreateAccountRequest;
import bank.communication.request.DepositRequest;
import bank.communication.request.GetAccountNumbersRequest;
import bank.communication.request.GetAccountRequest;
import bank.communication.request.GetBalanceRequest;
import bank.communication.request.GetOwnerRequest;
import bank.communication.request.IRequest;
import bank.communication.request.IsActiveRequest;
import bank.communication.request.TransferRequest;
import bank.communication.request.WithdrawRequest;

/**
 * This abstract class provides an implementation of the IBankDriver interface with the
 * additional method handleMessage(...) to send and receive objects.
 *
 * @see IBankDriver
 * @author Thomas Baumann
 * @version 1.1
 */
public abstract class AbstractClientDriver implements IBankDriver {

    protected IBank bank;

    @Override
    public final IBank getBank() {
        return this.bank;
    }

    /**
     * Sends a request object and receives afterwards the answer object and returns the
     * data from the answer object.
     *
     * @param r the request to write
     * @return answer object
     * @throws ClassNotFoundException Class of a read object cannot be found
     * @throws IOException When an IO problem occurs
     * @throws IllegalArgumentException When answer is an IllegalArgumentException
     * @throws OverdrawException When answer is an OverdrawException
     * @throws InactiveException When answer is an InactiveException
     */
    protected abstract <T> T handleMessage(IRequest r) throws ClassNotFoundException,
            IOException, IllegalArgumentException, OverdrawException, InactiveException,
            ClassCastException;

    protected final class SocketBank implements IBank {

        // public constructor for visibility
        public SocketBank() {}

        @Override
        public Set<String> getAccountNumbers() throws IOException {
            try {
                return AbstractClientDriver.this
                        .handleMessage(new GetAccountNumbersRequest());
            } catch (ClassNotFoundException | IllegalArgumentException
                    | OverdrawException | InactiveException | ClassCastException e) {
```

```java
 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 4: 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
16  /**
17   * This class provides an implementation of the AbstractClientDriver with HTTP.
18   *
19   * @see AbstractClientDriver
20   * @author Thomas Baumann
21   * @version 1.0
22   */
23  public final class ClientDriver extends AbstractClientDriver {
24
25      private URL url;
26
27      @Override
28      public void connect(String[] args) throws IOException {
29          if (args.length < 1) {
30              System.out.println("Usage: java " + StartClient.class.getName() + " "
31                      + ClientDriver.class.getName() + " <server>");
32              System.exit(1);
33          }
34          this.url = new URL(args[0]);
35          this.bank = new SocketBank();
36      }
37
38      @Override
39      public void disconnect() throws IOException {
40          this.bank = null;
41      }
42
43      @SuppressWarnings("unchecked")
44      @Override
45      protected <T> T handleMessage(IRequest request) throws ClassNotFoundException,
46              IOException, IllegalArgumentException, OverdrawException, InactiveException,
47              ClassCastException {
48          HttpURLConnection c = (HttpURLConnection) this.url.openConnection();
49          c.setRequestMethod("POST");
50          c.setRequestProperty("Content-Type", "application/x-java-serialized-object");
51          c.setDoOutput(true);
52          c.setDoInput(true);
53          c.connect();
54
55          // write object
56          ObjectOutputStream oout = new ObjectOutputStream(c.getOutputStream());
57          oout.writeObject(request);
58          oout.flush();
59          oout.close();
60
61          // read object
62          ObjectInputStream oin = new ObjectInputStream(c.getInputStream());
63          Object o = oin.readObject();
64          oin.close();
65
66          c.disconnect();
67          return ((IAnswer<T>) o).getData();
68
69      }
70  }
```

4 Kommunikation Objekte

4.1 Anfragen

```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);

}
```

```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 }
```

```java
 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 }
```

```java
 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 9: 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 10: 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 11: 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 12: Anfrage Objekt um Kontostand abzufragen

11

```java
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 balance request for an account.
12  *
13  * @see IRequest
14  * @author Thomas Baumann
15  * @version 1.1
16  */
17 public class GetBalanceRequest implements IRequest {
18
19     private String number;
20
21     public GetBalanceRequest(String number) {
22         this.number = number;
23     }
24
25     @Override
26     public IAnswer<?> handleRequest(IBank b) {
27         try {
28             Double balance = b.getAccount(this.number).getBalance();
29             return new Answer<Double>(balance);
30         } catch (IOException e) {
31             return new IOExceptionAnswer(e);
32         }
33     }
34
35 }
```

Listing 13: Anfrage Objekt für Aktiv/Inaktiv Zustand

```java
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 an is active request for an account.
12  *
13  * @see IRequest
14  * @author Thomas Baumann
15  * @version 1.1
16  */
17 public class IsActiveRequest implements IRequest {
18
19     private String number;
20
21     public IsActiveRequest(String number) {
22         this.number = number;
23     }
24
25     @Override
26     public IAnswer<?> handleRequest(IBank b) {
27         try {
28             boolean ans = b.getAccount(this.number).isActive();
29             return new Answer<Boolean>(ans);
30         } catch (IOException e) {
31             return new IOExceptionAnswer(e);
32         }
```

```
33        }
34
35  }
```

```
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  }
```

```
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  /**
```

```
16   * This class provides a withdraw request for an account.
17   *
18   * @see IRequest
19   * @author Thomas Baumann
20   * @version 1.1
21   */
22  public class WithdrawRequest implements IRequest {
23
24      private String number;
25      private Double amount;
26
27      public WithdrawRequest(String number, Double amount) {
28          this.number = number;
29          this.amount = amount;
30      }
31
32      @Override
33      public IAnswer<?> handleRequest(IBank b) {
34          try {
35              b.getAccount(this.number).withdraw(this.amount);
36              return new Answer<Object>(null);
37          } catch (IOException e) {
38              return new IOExceptionAnswer(e);
39          } catch (IllegalArgumentException e) {
40              return new IllegalArgumentExceptionAnswer(e);
41          } catch (OverdrawException e) {
42              return new OverdrawExceptionAnswer(e);
43          } catch (InactiveException e) {
44              return new InactiveExceptionAnswer(e);
45          }
46      }
47
48  }
```

4.2 Antworten

Listing 16: Interface für Antwort Objekte

```
1  package bank.communication.answer;
2
3  import java.io.IOException;
4  import java.io.Serializable;
5
6  import bank.InactiveException;
7  import bank.OverdrawException;
8
9  /**
10  * This interface must be used as answer object for the socket communication from the
11  * server to the client.
12  *
13  * @author Thomas Baumann
14  * @version 1.0
15  * @param <T>
16  */
17  public interface IAnswer<T> extends Serializable {
18
19      /**
20       * Returns an object or throws an exception.
21       *
22       * @return Returns the answer
23       * @throws IllegalArgumentException When answer is an IllegalArgumentException
24       * @throws IOException When an IO problem occurs
25       * @throws OverdrawException When answer is an OverdrawException
26       * @throws InactiveException When answer is an InactiveException
27       */
28      public T getData() throws IllegalArgumentException, IOException,
29              OverdrawException, InactiveException;
30
31  }
```

```
1  package bank.communication.answer;
2
3  /**
4   * This class provides a close account answer. It includes an boolean which specifies, if
5   * the close account request was successful or not.
6   *
7   * @see IAnswer
8   * @author Thomas Baumann
9   * @version 1.0
10  */
11 public class Answer<T> implements IAnswer<T> {
12
13     private T value;
14
15     public Answer(T value) {
16         this.value = value;
17     }
18
19     @Override
20     public T getData() {
21         return this.value;
22     }
23
24 }
```

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

```
1  package bank.communication.answer;
2
3  import bank.InactiveException;
4
5  /**
6   * This class provides a InactiveException answer. It includes an exception of the type
7   * inactive. 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 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 20: 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 21: 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 }
```