A. Scenario 1

Listing A.1: Cloud.sj

```
1  //$ bin/sjc tests/src/thesis/Cloud.sj -d tests/classes/
2  //$ bin/sj -cp tests/classes/ thesis.V3na 9999
3
4  package thesis;
5
6  import sj.runtime.*;
7  import sj.runtime.net.*;
8
9
10 import java.io.IOException;
11
12 import java.util.*;
13 import java.net.UnknownHostException;
14 import org.json.simple.JSONValue;
15 import org.json.simple.parser.*;
16
17
18
19 public class Cloud {
20   public static final int SUCCESS = 1;
21   public static final int UNSUCCESS = 0;
22
23   public static void main(String []argv) {
24     try {
25       new Cloud(Integer.parseInt(argv[0]), argv[1], Integer.parseInt(argv[2]));
26     } catch(IOException ioe) {
27       ioe.printStackTrace();
28     }
29
30   }
31   private protocol http_req_rep {
32       !<HttpRequestJSONMessage>.?(HttpResponseJSONMessage)
33   }
34
35   private protocol p_vu {
36     begin.?(JSONMessage).!{
37       OK: !<JSONMessage>.!<int>,
38       FAIL:
39     }
40   }
41
42   private void print(String s) {
```

```
43      System.out.println(s);
44    }
45    private boolean verify_msg(JSONMessage info) {
46      return true;
47    }
48    public Cloud(int portN, String saas_hname, int saas_port) throws IOException {
49      SJServerSocket v3naS_vu = SJRServerSocket.create(p_vu, portN);
50      SJSocket user_vu = null;
51
52      protocol p_vs { begin.@http_req_rep }
53
54      try(user_vu) {
55        user_vu = v3naS_vu.accept();
56        JSONMessage request_info = user_vu.receive();
57        print("Request for connection: " + request_info);
58        if(this.verify_msg(request_info)) {
59          user_vu.outbranch(OK) {
60            Map msg = new LinkedHashMap();
61            msg.put("status", "3");
62            msg.put("message", "Wait please.");
63            user_vu.send(JSONMessage.create(JSONValue.toJSONString(msg)));
64
65
66            SJServerAddress addr_vs = SJServerAddress.create(
67                p_vs, saas_hname, saas_port);
68            SJSocket s_vs = SJRSocket.create(addr_vs);
69            Map http_resp = null;
70            try(s_vs) {
71                s_vs.request();
72                s_vs.send(new HttpRequestJSONMessage(request_info.toString()));
73                http_resp = ((HttpResponseJSONMessage) s_vs.receive()).parse();
74            } catch(UnknownHostException uhe) {
75                uhe.printStackTrace();
76            }
77            user_vu.send(Integer.parseInt((String) http_resp.get("status")));
78          }
79        } else {
80          user_vu.outbranch(FAIL) {
81            System.out.println("CONNECTION FAILED");
82          }
83        }
84      } catch (SJIOException ioe) {
85          ioe.printStackTrace();
86      } catch(SJIncompatibleSessionException stise) {
87          stise.printStackTrace();
88      } catch(ClassNotFoundException cnfe) {
89          cnfe.printStackTrace();
90      }
```

```
91    }
92 }
```

Listing A.2: Saas.sj

```
1  //$ bin/sjc tests/src/thesis/Cloud.sj -d tests/classes/
2  //$ bin/sj -cp tests/classes/ thesis.V3na 9999
3
4  package thesis;
5  import sj.runtime.*;
6  import sj.runtime.net.*;
7
8  import java.io.IOException;
9  import java.util.*;
10 import org.json.simple.JSONValue;
11 import org.json.simple.parser.*;
12
13 class Saas {
14     private static final int OK = 1;
15     public static void main(String []argv) {
16         try {
17             new Saas(Integer.parseInt(argv[0]));
18         } catch(IOException ioe) {
19             ioe.printStackTrace();
20         }
21
22     }
23     private protocol p_sv {
24         begin.?(HttpRequestJSONMessage).!<HttpResponseJSONMessage>
25     }
26
27     public Saas(int portNumber) throws IOException{
28         SJServerSocket server_sv = SJRServerSocket.create(p_sv, portNumber);
29         SJSocket v3na_sv = null;
30         try(v3na_sv) {
31             v3na_sv = server_sv.accept();
32             HttpRequestJSONMessage msg = v3na_sv.receive();
33             Map response = new LinkedHashMap();
34             response.put("message", "Created.");
35             response.put("status", "1");
36             v3na_sv.send(new HttpResponseJSONMessage(JSONValue.toJSONString(
                   response)));
37         } catch(Exception e) {
38             e.printStackTrace();
39         }
40     }
41 }
```

Listing A.3: User.sj

```
1  //$ bin/sjc tests/src/thesis/User.sj -d tests/classes/
2  //$ bin/sj -cp tests/classes/ thesis.User localhost 9999
3
4
5  package thesis;
6
7  import sj.runtime.*;
8  import sj.runtime.net.*;
9
10 import java.util.*;
11
12 import org.json.simple.JSONValue;
13 import org.json.simple.parser.*;
14 import java.util.Date;
15 import java.sql.Timestamp;
16 // import thesis.utils.JSON;
17 public class User
18 {
19   public static void main(String []argv) {
20     try {
21
22       new User(argv[0], Integer.parseInt(argv[1]));
23
24     } catch(SJIOException sjioe) {
25
26       sjioe.printStackTrace();
27
28     }
29
30   }
31   private static protocol p_uv {
32       begin.
33       !<JSONMessage>.
34       ?{
35         OK: ?(JSONMessage).?(int),
36         FAIL:
37       }
38   }
39
40   private Map buildConnectionRequest() {
41       long curTime = new Timestamp((new Date()).getTime()).getTime();
42       Map msg = new LinkedHashMap();
43       msg.put("action", "CONNECTION");
44       msg.put("client_email", "r.kamun@gmail.com");
45       msg.put("ts", curTime + "");
46       return msg;
47   }
```

```
48    private void print(String s) {
49        System.out.println(s);
50    }
51
52    public User(String hname, int port) throws SJIOException {
53        SJServerAddress v3na_addr = SJServerAddress.create(p_uv, hname, port);
54        SJSocket s_uv = SJRSocket.create(v3na_addr);
55
56        try(s_uv) {
57          s_uv.request();
58
59          Map msg = this.buildConnectionRequest();
60          s_uv.send(JSONMessage.create(JSONValue.toJSONString(msg)));
61          msg = null;
62
63          s_uv.inbranch() {
64            case OK: {
65              msg = ((JSONMessage) s_uv.receive()).parse();
66              print("Status: " + msg.get("message"));
67              int status = s_uv.receiveInt();
68              print("Status: " + status);
69            }
70            case FAIL: {
71              print("FAILED");
72            }
73          }
74        } catch(SJIOException sjioe) {
75            sjioe.printStackTrace();
76        } catch(Exception e) {
77            e.printStackTrace();
78        }
79    }
80 }
```

B. Scenario 2

Listing B.1: Cloud.sj

```
1  //$ bin/sjc tests/src/thesis/Cloud.sj -d tests/classes/
2  //$ bin/sj -cp tests/classes/ thesis.V3na 9999
3
4  package thesis.scenario2;
5
6  import sj.runtime.*;
7  import sj.runtime.net.*;
8
9
10 import java.io.IOException;
11
12 import java.util.*;
13 import java.net.UnknownHostException;
14 import org.json.simple.JSONValue;
15 import org.json.simple.parser.*;
16
17
18
19 public class Cloud {
20   public static final int SUCCESS = 1;
21   public static final int UNSUCCESS = 0;
22   private Map database = new LinkedHashMap();
23   public static void main(String []argv) {
24     try {
25       new Cloud(Integer.parseInt(argv[0]), argv[1], Integer.parseInt(argv[2]));
26     } catch(IOException ioe) {
27       ioe.printStackTrace();
28     }
29
30   }
31   private protocol p_vs {
32       begin.
33       !< !{
34             OK: !<JSONMessage>,
35             FAIL: !<JSONMessage>
36       } >.!<JSONMessage>
37   }
38
39   private protocol p_vu {
40     begin.
41     ![
42       ?(String).?(String)      // login password
```

```
43      ]*.
44        !{
45          ACCESS: ?(JSONMessage).
46            !{
47              OK: !<JSONMessage>,
48              FAIL: !<JSONMessage>
49            },
50          DENY: !<String>
51        }
52    }
53
54    private void print(String s) {
55       System.out.println(s);
56    }
57    private boolean verify_msg(JSONMessage info) {
58      return true;
59    }
60    private boolean is_authenticated(String login, String password) {
61        try {
62            return this.database.get(login).equals(password);
63        } catch(java.lang.NullPointerException exc) {
64            return false;
65        }
66    }
67    private void connectToDB() {
68        this.database.put("r.kamun@gmail.com", "00112358");
69    }
70    public Cloud(int portN, String saas_hname, int saas_port) throws IOException {
71      this.connectToDB();
72
73      SJServerSocket v3naS_vu = SJRServerSocket.create(p_vu, portN);
74      SJSocket user_vu = null;
75
76      try(user_vu) {
77        user_vu = v3naS_vu.accept();
78        boolean exit = false;
79        int counter = 0, max_atempts = 5;
80        user_vu.outwhile(!exit) {
81            String login = user_vu.receive();
82            String password = user_vu.receive();
83            counter ++;
84            if(this.is_authenticated(login, password) || (counter >= max_atempts))
                  {
85                exit = true;
86            }
87        }
88        if(counter < max_atempts) {
89            user_vu.outbranch(ACCESS) {
```

```
90                JSONMessage req_info = user_vu.receive();
91                SJServerAddress addr_vs = SJServerAddress.create(
92                    p_vs, saas_hname, saas_port);
93                SJSocket s_vs = SJRSocket.create(addr_vs);
94                try(s_vs) {
95                    s_vs.request();
96                    s_vs.send(user_vu);    // pass the remaining protocol
97                    s_vs.send(req_info);
98                } catch(UnknownHostException uhe) {
99                    uhe.printStackTrace();
100               }
101           }
102       } else {
103           user_vu.outbranch(DENY) {
104               user_vu.send("You have no permissions. BYE!");
105           }
106       }
107
108   } catch (SJIOException ioe) {
109       ioe.printStackTrace();
110   } catch(SJIncompatibleSessionException stise) {
111       stise.printStackTrace();
112   } catch(ClassNotFoundException cnfe) {
113       cnfe.printStackTrace();
114   }
115  }
116 }
```

Listing B.2: Saas.sj

```
1  //$ bin/sjc tests/src/thesis/Cloud.sj -d tests/classes/
2  //$ bin/sj -cp tests/classes/ thesis.V3na 9999
3
4  package thesis.scenario2;
5  import sj.runtime.*;
6  import sj.runtime.net.*;
7
8  import java.io.IOException;
9  import java.util.*;
10 import org.json.simple.JSONValue;
11 import org.json.simple.parser.*;
12
13 class Saas {
14     private static final int OK = 1;
15     public static void main(String []argv) {
16         try {
17             new Saas(Integer.parseInt(argv[0]));
18         } catch(IOException ioe) {
19             ioe.printStackTrace();
```

```
20          }
21
22      }
23      private protocol p_msg {
24          !{
25            OK: !<JSONMessage>,
26            FAIL: !<JSONMessage>
27          }
28      }
29
30      private protocol p_sv {
31          begin.?(@p_msg).?(JSONMessage)
32      }
33      private boolean verify_msg(JSONMessage params) {
34          return 1 == 1;
35      }
36      private boolean validate_msg(JSONMessage params) {
37          return true;
38      }
39      private JSONMessage genResponse(String status, String message) {
40          Map m = new LinkedHashMap();
41          m.put("status", status);
42          m.put("message", message);
43          return JSONMessage.create(JSONValue.toJSONString(m));
44      }
45      public Saas(int portNumber) throws IOException{
46          SJServerSocket server_sv = SJRServerSocket.create(p_sv, portNumber);
47          SJSocket v3na_sv = null;
48          SJSocket v3na_user_socket = null;
49          try(v3na_sv, v3na_user_socket) {
50              v3na_sv = server_sv.accept();
51              v3na_user_socket = (@p_msg) v3na_sv.receive();
52              JSONMessage req_params = (JSONMessage) v3na_sv.receive();
53              boolean allowed = this.verify_msg(req_params) && this.validate_msg(
                    req_params);
54              if(allowed) {
55                  v3na_user_socket.outbranch(OK) {
56                      v3na_user_socket.send(
57                          this.genResponse("1", "USER CREATED. Email has been sent
                                .")
58                      );
59                  }
60              } else {
61                  v3na_user_socket.outbranch(FAIL) {
62                        v3na_user_socket.send(this.genResponse("0", "FAILED."));
63                  }
64              }
65          } catch(Exception e) {
```

```
66              e.printStackTrace();
67          }
68      }
69  }
```

Listing B.3: User.sj

```
1  package thesis.scenario2;
2
3  import sj.runtime.*;
4  import sj.runtime.net.*;
5
6  import java.util.*;
7
8  import org.json.simple.JSONValue;
9  import org.json.simple.parser.*;
10 import java.util.Date;
11 import java.sql.Timestamp;
12 import java.io.*;
13 public class User
14 {
15   public static void main(String []argv) {
16     try {
17
18       new User(argv[0], Integer.parseInt(argv[1]));
19
20     } catch(SJIOException sjioe) {
21
22       sjioe.printStackTrace();
23
24     }
25
26   }
27   private static protocol p_uv {
28       begin.?[
29         !<String>.!<String>
30       ]*.
31         ?{
32           ACCESS: !<JSONMessage>.
33             ?{
34                 OK: ?(JSONMessage),
35                 FAIL: ?(JSONMessage)
36             },
37           DENY: ?(String)
38       }
39   }
40
41   private Map buildConnectionRequest() {
42       long curTime = new Timestamp((new Date()).getTime()).getTime();
```

```
43          Map msg = new LinkedHashMap();
44          msg.put("action", "CONNECTION");
45          msg.put("client_email", "r.kamun@gmail.com");
46          msg.put("ts", curTime + "");
47          return msg;
48      }
49      private void print(String s) {
50          System.out.println(s);
51      }
52
53      private String login;
54      private String password;
55      Scanner sc = new Scanner(new InputStreamReader(System.in));
56      public User(String hname, int port) throws SJIOException {
57          SJServerAddress v3na_addr = SJServerAddress.create(p_uv, hname, port);
58          SJSocket s_uv = SJRSocket.create(v3na_addr);
59
60          try(s_uv) {
61            s_uv.request();
62            // login password
63            s_uv.inwhile() {
64              System.out.println("Enter your login:");
65              this.login = sc.nextLine();
66              System.out.println("Enter your password:");
67              this.password = sc.nextLine();
68              s_uv.send(this.login); s_uv.send(this.password);
69            }
70            s_uv.inbranch() {
71                case ACCESS: {
72                    s_uv.send(
73                        JSONMessage.create(
74                            JSONValue.toJSONString(this.buildConnectionRequest())
75                        )
76                    );
77                    s_uv.inbranch() {
78                      case OK: {
79                        Map msg = ((JSONMessage) s_uv.receive()).parse();
80                        print("Result: " + msg.get("message"));
81                      }
82                      case FAIL: {
83                        Map msg = ((JSONMessage) s_uv.receive()).parse();
84                        print("Reason: " + msg.get("message"));
85                      }
86                    }
87                }
88                case DENY: {
89                    print((String)s_uv.receive());
90                }
```

```
91            }
92
93        } catch(SJIOException sjioe) {
94            sjioe.printStackTrace();
95        } catch(Exception e) {
96            e.printStackTrace();
97        }
98    }
99 }
```