Distributed Computing - Assignment 1

Jaeseok Huh, 2015005241, Department of Computer Science and Engineering
jaeseok@hanyang.ac.kr

Environment

Ubuntu 18.04 LTS 64bit
JDK-11, JRE-11

How to compile

```
$ ./build.sh
```

How to run

```
$ ./run_rmiregistry.sh
```

Wait for a few seconds. Then, start new session.

```
$ ./run_server.sh
```

In another session,

```
$ ./run_client.sh
===========================
1. Add Schedule
2. Delete Schedule
3. Retrieve Schedule
4. Exit
===========================
1
Enter start time (year month day hour minute) :
2018 10 01 13 00
Enter end time (year month day hour minute) :
2018 10 01 14 00
Title: Algorithm
Sucessfully added. ID: 0
===========================
1. Add Schedule
2. Delete Schedule
3. Retrieve Schedule
4. Exit
===========================
1
Enter start time (year month day hour minute) :
2018 10 01 11 30
Enter end time (year month day hour minute) :
2018 10 01 12 30
```

```
Title: Lunch
Sucessfully added. ID: 1
============================
1. Add Schedule
2. Delete Schedule
3. Retrieve Schedule
4. Exit
============================
1
Enter start time (year month day hour minute) :
2018 10 01 10 00
Enter end time (year month day hour minute) :
2018 10 01 14 00
Title: Date
Overlapping event(s) exists.
============================
1. Add Schedule
2. Delete Schedule
3. Retrieve Schedule
4. Exit
============================
3
Enter start time (year month day hour minute) :
1 1 1 1 1
Enter end time (year month day hour minute) :
2018 10 01 12 00
0 event(s) found.
============================
1. Add Schedule
2. Delete Schedule
3. Retrieve Schedule
4. Exit
============================
3
Enter start time (year month day hour minute) :
2018 10 01 10 00
Enter end time (year month day hour minute) :
2018 10 01 13 00
1 event(s) found.
Event Lunch from 2018/10/1 11:30 to 2018/10/1 12:30
============================
1. Add Schedule
2. Delete Schedule
3. Retrieve Schedule
4. Exit
============================
2
ID: 1
============================
1. Add Schedule
2. Delete Schedule
3. Retrieve Schedule
4. Exit
============================
3
Enter start time (year month day hour minute) :
2018 10 01 00 00
Enter end time (year month day hour minute) :
```

```
2018 10 01 15 00
1 event(s) found.
Event Algorithm from 2018/10/1 13:00 to 2018/10/1 14:00
===========================
1. Add Schedule
2. Delete Schedule
3. Retrieve Schedule
4. Exit
===========================
4
```

Design Structure

```
/
    doc/
        README.md  # report
    run_client.sh
    run_rmiregistry.sh
    run-server.sh
    src/
        client/
            Client.java
        schedule/
            CalendarEvent.java   # Class for storing event
            CalenaarService.java # Interface for both client and server
        server/
            Server.java
```

Client serves as an user interface for accessing data in *Server*.
Since the range of *GregorianCalendar* is 0-11, *Client* makes the month of input to fit in by reducing it by 1. *Client* tries to locate a registry and look up for the name of pre-determined interface, "CalendarService".

schedule/CalendarEvent provides a class, which serves as an instance for events. It contains the description(*desc*) and *id* (auto-increment). Also, a function to check whether its event overlaps with other event or not is implemented.

schedule/CalendarSerivce serves as an interface, which is implemented by *server*. *Client* looks up for it through rmiregistry and *Server* binds to it. Since relations among them are susceptible to exceptions (due to network instability, etc), in those cases, *RemoteException* should be thrown.

server/Server initializes variables for storing data, and exports and bind the implementation of the common interface shared with client. Please note that the result of *retrieveSchedule* consist of *CalendarEvent*.