Zebo Xiong – A04907051 (HW3)

1. (60 pts) Modify/enhance the Java RMI program WeatherService discussed in class as follows:

   a. WeatherServiceServer will periodically update its weather information, say every one hour.

1) Run server:

```
dior@MacBook-Pro WeatherServiceCode % ls
WeatherBean.class               WeatherService.class            imagenames.properties.bak
WeatherBean.java                WeatherService.java             images
WeatherCellRenderer.class       WeatherServiceClient.class      policy-all.txt
WeatherCellRenderer.java        WeatherServiceClient.java       policy1-file.txt
WeatherItem.class               WeatherServiceCode.iml          rnusers.c
WeatherItem.java                WeatherServiceServer.class      test.html
WeatherListModel.class          WeatherServiceServer.java
WeatherListModel.java           imagenames.properties
dior@MacBook-Pro WeatherServiceCode % !javac
javac *.java
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
dior@MacBook-Pro WeatherServiceCode % java -Djava.security.policy=policy1-file.txt WeatherServic
eServer
```

2) The server will update weather information every one hour

```
YAKIMA            --UNNY --47 -- 53/83  -- SUNNY--  51/86--
condition --> SUNNY<--------------------------------------------------
City:YAKIMA, condition:SUNNY,temp:47  81,nextDayT: 53/83   ,nextDayC: SUNNY ,next
Update weather bean: City=YAKIMAtemp=47  81condition=SUNNY
url_1 ---->images/sunny.jpg
url_2 ---->images/sunny.jpg
YOUNGSTOWN        --UNNY --58 -- 59/82  -- SUNNY--  60/86--
condition --> SUNNY<--------------------------------------------------
City:YOUNGSTOWN, condition:SUNNY,temp:58  82,nextDayT: 59/82   ,nextDayC: SUNNY ,
0/86
Update weather bean: City=YOUNGSTOWNtemp=58  82condition=SUNNY
url_1 ---->images/sunny.jpg
url_2 ---->images/sunny.jpg
YUMA              --RYHOT--76 -- 74/110 -- VRYHO--  75/11--
condition --> VRYHOT<--------------------------------------------------
City:YUMA, condition:VRYHOT,temp:76 108,nextDayT: 74/110  ,nextDayC: VRYHOT,nextN
Update weather bean: City=YUMAtemp=76 108condition=VRYHOT
url_1 ---->images/vryhot.jpg
url_2 ---->images/vryhot.jpg
Initializing WeatherService...5
Weather information updated.
WeatherService running....  <----- It is the 4(st/rd/th) time update!!
```

3) The code in server side:

```
// launch WeatherService remote object
public static void main( String args[] )
{

    System.setProperty("java.rmi.server.hostname", "localhost");

    try {

        int count = 1;

        while(true){

            int page = 0;

            System.setSecurityManager (new SecurityManager());

            System.err.println( "Initializing WeatherService..." );

            Registry registry = LocateRegistry.getRegistry( host: "localhost",  port: 9999);

            for(int i = 1; i <=4; i++){

                page = i;

                WeatherServiceServer obj = new WeatherServiceServer(page);
                WeatherService stub = (WeatherService) UnicastRemoteObject.exportObject(obj,  port: 0);

                registry.rebind( name: "WeatherService" + page, stub);

                System.err.println( "WeatherService running....  <----- It is the " + count + "(st/rd/th) time update!!" );

            }

            count++;
            Thread.sleep( millis: 3600000);
        }

    } catch (Exception e) {
        System.err.println("Server exception: " + e.toString());
        e.printStackTrace();
    }
```

The while loop was used and it will pull the data from HTTP server every 3600 seconds.

4) Also, in the client side,

```
dior@MacBook-Pro Desktop % cd WeatherServiceCode
dior@MacBook-Pro WeatherServiceCode % ls
WeatherBean.class               WeatherServiceClient.java
WeatherBean.java                WeatherServiceCode.iml
WeatherCellRenderer.class       WeatherServiceServer.class
WeatherCellRenderer.java        WeatherServiceServer.java
WeatherItem.class               imagenames.properties
WeatherItem.java                imagenames.properties.bak
WeatherListModel.class          images
WeatherListModel.java           policy-all.txt
WeatherService.class            policy1-file.txt
WeatherService.java             rnusers.c
WeatherServiceClient.class      test.html
dior@MacBook-Pro WeatherServiceCode % java -Djava.security.policy=policy1-file.txt Wea
therServiceClient
weatherBean.getCityName().toString():  = ABILENE TX
weatherBean.getCityName().toString():  = AKRON CANTON
weatherBean.getCityName().toString():  = ALBANY NY
```

5) This is the code from client: (call data from server every one hour)

```
58  public static void main( String args[] ) throws InterruptedException {
59
60      while(true){
61
62          WeatherServiceClient[] clients = new WeatherServiceClient[5];
63
64          System.setSecurityManager (new SecurityManager());
65
66          int page;
67
68          for(int i = 1; i <=4; i++){
69
70              page = i;
71
72              if ( args.length == 0 )
73                  clients[i] = new WeatherServiceClient( server: "localhost", page);
74              else
75                  clients[i] = new WeatherServiceClient( server: "localhost", page);
76
77
78              // configure and display application window
79              clients[i].setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
80              clients[i].pack();
81              clients[i].setResizable( true );
82              clients[i].setVisible( true );
83
84          }
85
86          Thread.sleep( millis: 3600000);
87
88          for(WeatherServiceClient client:clients){
89              client.setVisible(false); //you can't see me!
90              client.dispose(); //Destroy the JFrame object
91          }
92      }
93
94  }
```

b. There are four sections of the city weather forecase information on each page. Modify the program so that the server receives and saves all four section data. And the clients will pop four windows, one for each of the four section. in addition to city name and weather condition, your server now will record information of all columns as shown above and send them back to a client.

1) I pull the data from four different sections:

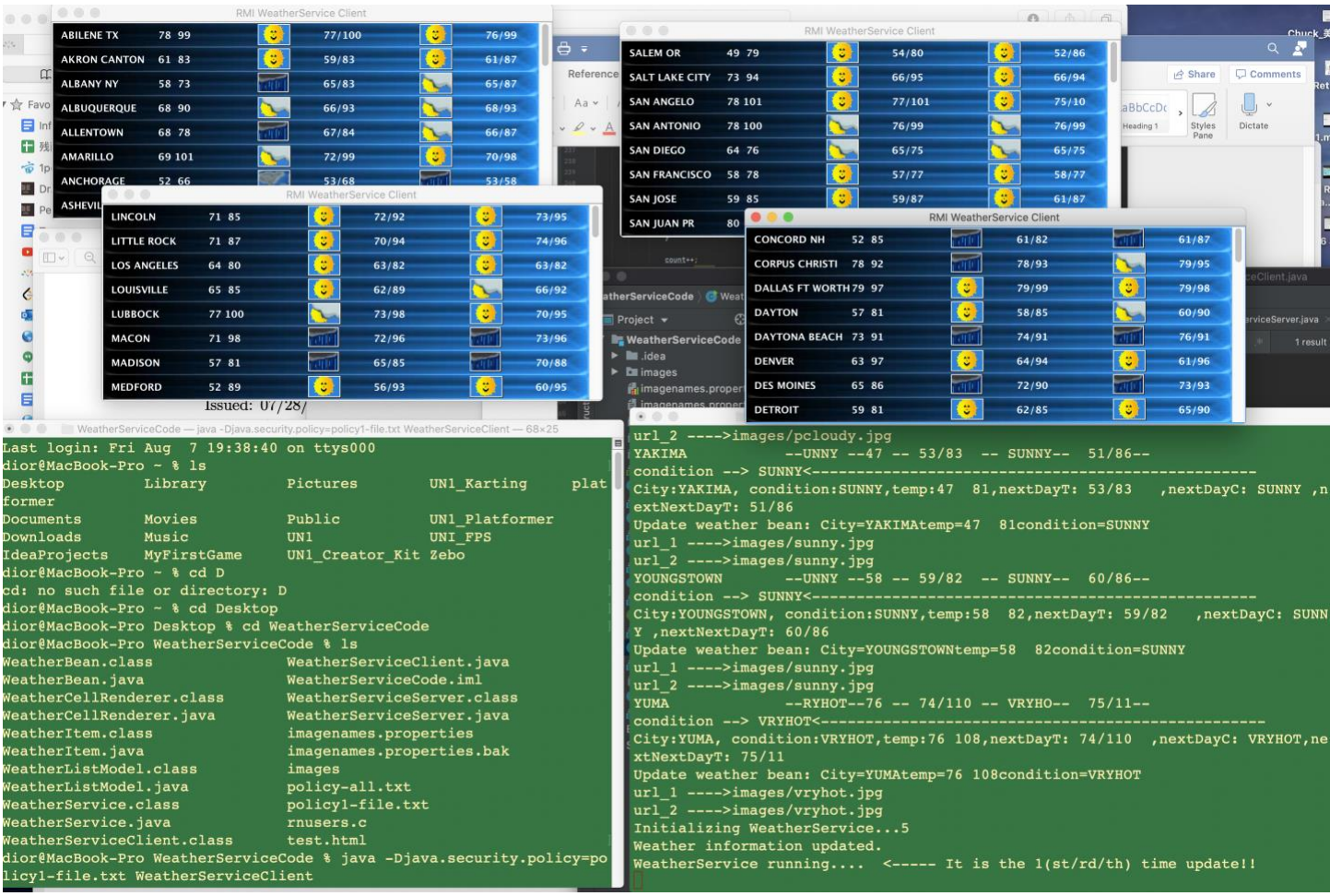
```
232                 Registry registry = LocateRegistry.getRegistry( host: "localhost",  port: 9999);
233
234                 for(int i = 1; i <=4; i++){
235
236                     page = i;
237
238                     WeatherServiceServer obj = new WeatherServiceServer(page);
239                     WeatherService stub = (WeatherService) UnicastRemoteObject.exportObject(obj,  port: 0);
240
241                     registry.rebind( name: "WeatherService" + page, stub);
242
243                     System.err.println( "WeatherService running....  <----- It is the " + count + "(st/rd/th) time update!!" );
244
245                 }
246
247                 count++;
248                 Thread.sleep( millis: 3600000);
```

2) I used variable page to indicate the different sections

3) Also in client side, I will display the four sections in different windows:

```
60      while(true){
61
62          WeatherServiceClient[] clients = new WeatherServiceClient[5];
63
64          System.setSecurityManager (new SecurityManager());
65
66          int page;
67
68          for(int i = 1; i <=4; i++){
69
70              page = i;
71
72              if ( args.length == 0 )
73                  clients[i] = new WeatherServiceClient(  server: "localhost", page);
74              else
75                  clients[i] = new WeatherServiceClient(  server: "localhost", page);
76
77
78              // configure and display application window
79              clients[i].setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
80              clients[i].pack();
81              clients[i].setResizable( true );
82              clients[i].setVisible( true );
83
84          }
```

4) Four windows has shown:



5) Besides, I mapped “Mostly Cloudy” icon:

6) The code to add extra column in WeatherServiceServer:

```
138
139                 nextDayT = inputLine.substring( 39, 48 );
140                 nextDayC = inputLine.substring( 47, 54 );
141                 nextNextDayT = inputLine.substring( 55, 61 );
142
143                 cityName = cityName.trim();
144                 temperatures = temperatures.trim();
145                 condition = condition.trim();
146                 nextDayT.trim();
147                 nextDayC.trim();
148                 nextNextDayT.trim();
149
150                 WeatherBean weather = new WeatherBean(
151                                                       cityName,
152                                                       condition,
153                                                       temperatures,
154                                                       nextDayT,
155                                                       nextDayC,
156                                                       nextNextDayT
157
158                                                       );
159
160                 // add WeatherBean to List
161                 weatherInformation.add( weather );
```

7) The code to be added in WeatherBean:

```
57 @    public WeatherBean(
58                         String city,
59                         String weatherDescription,
60                         String cityTemperature,
61                         String nextDayT,
62                         String nextDayC,
63                         String nextNextDayT)
64     {
65         this.cityName = city;
66         this.temperature = cityTemperature;
67         this.description = weatherDescription.trim();
68
69         this.nextDayT = nextDayT;
70         this.nextDayC = nextDayC.trim();
71         this.nextNextDayT = nextNextDayT;
72
73
74         String aa = "images/" +  imageNames.getProperty( description,  defaultValue: "noinfo.jpg" );
75         String bb = "images/" +  imageNames.getProperty( this.nextDayC,  defaultValue: "noinfo.jpg" );
76
77         URL url_1 = WeatherBean.class.getResource( aa );
78         URL url_2 = WeatherBean.class.getResource( bb );
79
80         image_1 = new ImageIcon( url_1 );
81
82         image_2 = new ImageIcon( url_2 );
83     }
84
```

```
108
109     public String getTemperature_2()
110     {
111     return this.nextDayT ;
112     }
113     public String getTemperature_3()
114     {
115     return this.nextNextDayT;
116     }
117
118     // get weather image
119     public ImageIcon getImage_2()
120     {
121         return image_2;
122     }
123 }
```

8) The code added to WeatherItem:

```
54      if(weatherBean.getDescription().trim() != "City Name"){
55
56          // draw background
57          backgroundImage.paintIcon( c: this, g, x: 0, y: 0 );
58
59          // set font and drawing color,
60          // then display city name and temperature
61          Font font = new Font( name: "SansSerif", Font.BOLD, size: 12 );
62          g.setFont( font );
63          g.setColor( Color.white );
64          g.drawString( weatherBean.getCityName(), x: 10, y: 19 );
65          g.drawString( weatherBean.getTemperature(), x: 130, y: 19 );
66          weatherBean.getImage().paintIcon( c: this, g, x: 253, y: 1 );
67
68          g.drawString( weatherBean.getTemperature_2(), x: 330, y: 19 );
69          weatherBean.getImage_2().paintIcon( c: this, g, x: 453, y: 1 );
70          g.drawString( weatherBean.getTemperature_3(), x: 530, y: 19 );
71
72      } else {
73          headerImage.paintIcon( c: this, g, x: 0, y: 0 );
74      }
```

Also, I tried to compile on server zeus, but not successful. I login server and copied code to zeus:

1. ssh z_x3@zeus.cs.txstate.edu
2. scp WeatherServiceCode.zip z_x3@zeus.cs.txstate.edu:/home/Students/z_x3/5352_summer
3. unzip WeatherServiceCode.zip
4. I updated the policy file and even test with absolute path. However, “WeatherBean.class.getResource()” keep return null. It seems that it could not find the correct image path.

2. $(20 \times 2 = 40$ pts) Based on your understanding, in a solution to the Byzantine general problem:

(1) Briefly explain in your own words why two good generals cannot overcome one bad general to reach an agreement among them?

As we discussed in class, if there are three generals, and one of them is bad, it is no way to align the action. Let us say they are general A, B, C.

Case one: General A is bad. He give true message to B, and false message to C. Even though B and C exchange their received message from A, both B and C have 50% true message and 50% false message. B and C cannot tell which one is correct

Case two: General A is good. He will both true message to B and C. However, B is bad. B exchange a bad message to C, and then C have 50% true message (from A) and 50% false message (from B). C cannot make decision.

Case three: same as case two if A is good while C is bad. (A will still give true message to both B and C)

They cannot reach an agreement – the "majority" function is not working in this 50% true 50% false situation.

(2) Why multiple stages of communications are necessary?

1) There exists bad general (faulty process) who may modify the contents of an incoming message, and forward wrong message deliberately. It can be accident or being hacked.

2) The name (id) of the message of a message is unknown to the receiver;

3) The bad general may choose not to forward any message after receiving;

4) It means any messenger can be attacked or lost or stopped during the communication.

5) In this case, we need multiple stage of communication to overcome the faulty processes (bad general). By multiple stage of communication, the good general can finally pass the correct message to the receiver.

6) After multiple stages, all the good general (except the traitor) can use "majority rule" to make the decision.

7) If the maximum number of faulty processes is T at beginning, then any solution to the problem will need at least T + 1 stages of message exchanges to arrive at a consensus.

8) To sum up, we need the multiple stage to overcome the inconsistent information and failure situation, and finally reach a correct decision with all generals. (The so-called "Practical Byzantine Fault Tolerance" derives from this)