SOFT8023 – Distributed Systems Programming

Assignment 2 Part 2

Worth: 10% **Due:** End of Week 12

Overview

Write an RMI-based application with 1 client and 2 server objects.

The client is an alert system situated in a national forest. When humidity is low, the temperature is high and there is a very low level of cloud cover, the risk of a forest fire increases.

The 2 server objects are:

- A sensor that takes 1.5 seconds to respond with a reading for temperature and humidity
- A "cloud cover" remote service that looks up the meteorological service to get a percentage cloud cover for the current hour; it takes 3 seconds to respond with the percentage

The risk is calculated as temperature (Celcius) x (100 - Humidity %) x (100 - Cloud Cover %). For example, if the sensor returned temp of 28C and humidity of 15%, and the met service returned cloud cover of 18%, then this would be 28 x (1 – 0.15) x (1 – 0.18) = 19.516. If the risk score is greater than 20, you must raise an alarm (a message to the console is enough).

You should check the risk once every 10 seconds. Based on the response times of the 2 remote objects, it should take about 3 seconds to calculate the response, which is well within the 10 seconds per risk assessment.

For example code, see the modified Fruit **Polling** example attached to the assignment on Blackboard. This is similar to the previous Java Swing version, but removes Swing and simply checks for the next fruit 3 times with various gaps in between, printing the fruit returned to the console. You can modify this code to add a second service. In the client, simply check the status of both remote services (doneYet returns true for both). In the service classes, simulate readings using random numbers – just restrict the random number ranges so that an alert will usually be raised within a minute.

Sample output (with 10 seconds between lines, number of lines will vary):

```
Checking… no risk
Checking… no risk
Checking… no risk
Checking… risk detected (21.312)!!!
```

Marks

The above code is worth 8 marks of the 10: 5 marks for getting RMI and polling working, 1 mark for the threading and 2 marks for correct functionality.

For the final 2 marks, answer the following question:

There is arguably a slightly better way of implementing this type of program instead of using polling. What is it and briefly explain, in about 2 sentences, what the approach entails and why it might be an improvement?

Insert your answer into the text box when submitting your assignment in Blackboard.