JAV745 Fall 2019: Project 2 - (12.5%)

Dr. Eden Burton
School of ICT, Seneca College of Applied Arts and Technology
Fall 2019

Due Saturday December 7, 2019 - 11:30 pm

Instructions

Please read the instructions carefully and follow the naming conventions specified for each question. Solutions must be submitted in the Blackboard Dropbox created for Project 2.

The deliverable instructions must be strictly followed. Your solution should be well documented using the JavaDoc utility to describe both your interface and your solution design.

Note that the deadline is strictly enforced. The system tracks the exact time that submissions are uploaded. **There is a 10% per day penalty for late submissions.**

Additional Notes

- You may use any IDE for development but note that demonstrations and professor testing will be done exclusively on the command line.

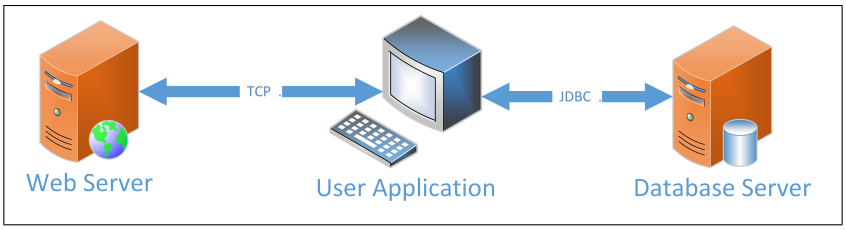

Figure 1: Weather Reading Application Architecture

Case Description

Your task is to build a system which stores and edits a list of weather readings (as in Lab 5). It shall also persist this information in a database. The solution will involve three independent processes running at the same time. They communicate with each other as shown in the Figure 1.

A description of how the various components should be built is provided below.

User Application

This component is where a user will manage the reading objects stored in the system. It will be stored in an *observable* data structure. The information stored here is the *model* of the whole application.

The given user interface will provide a *view* of model. It gives the user the ability to visualize (see) the information being stored. In this particular application, the user can also modify the current view using one of the following methods.

1. via user input, a user can manually input a new weather reading. Its contents must be validated before it is inserted into the model. A user must be able to edit or delete readings as well. Edits must validated before being modified in the model. Delete requests must be confirmed before being executed.
2. via external source, the user can send a TCP request to retrieve readings from a web server. A successfully serviced request will spawn a response which contains reading to be inserted into the model.
3. via a data store, upon application startup the model should be populated with information persisted before the last shutdown.

Any changes made to the model should be persisted to a database server when the user requests it. A JDBC database connection will be used to "flush" any modifications into a database. Your model should track changes to the model, so only necessary SQL calls are made to the database server. **Credentials required to connect to the database server must be passed into program via the command line.**

Use the supplied FXML file as a starting point for your design. Note that the ability to trigger a request to the web server is not in this file. You must add this capability.

Web Server

This will be a separate application / process from the User Application. Since the two processes do not share memory, they exchange data using sockets. Here, **the server will wait for requests on a configurable port supplied via the command line.** Upon receiving a request, the server will send a response containing a number of weather readings. The first line of the response will include the number of readings being sent. The subsequent lines contain data included for each reading. Each line represents the data for a single reading. The structure of the data sent is your design decision. The UI Application sending the request must interpret the response and add it to the model.

Use the `Reading` class from lab 5 to generate objects to be transmitted within a response. The number of readings sent in a response should be randomly determined. It shall send a minimum of 1 reading and a maximum of 5. Use the `Random` class in the `java.util` library to help generate random numbers.

Database Server

The model should be persisted into an database. Design a table so that reading objects can be persisted. Changes to the model should be sent to the database to ensure that a copy is stored to persistent storage. Upon the User Application restart, it should be able to reconstruct the model using the persisted data.

A script to create the table with instructions on how to add the database credentials should be submitted.

Deliverables

- an executable jar for the web server. The command `java -jar JAV745P2Server.jar` <port number> should start the application.
- an executable jar for the user application (this should include files necessary to run your application. In particular your fxml file and JDBC driver should be included). The command `java -jar JAV745P2UI.jar` <JDBC connection information> should start the application. You should specify the format of the connection string
- a script to create relevant tables
- a document which explains your system design and assumptions/design decisions

Note that the jar files submitted must and contain both source (*.java) and bytecode (*class) files.

Marking Criteria

- functionality ...Does the system actually work? Does it satisfy the requirements? Can it run using submitted information?
    1. *web server*. Does it successfully listen on a configurable port when the command specified above runs? Can it accept requests? Does it generate well formed responses?
    2. *user interface*. Does it allow the user to add, modify, delete readings? Can the model be retrieved/persisted in the database once the correct credentials are supplied? Is it possible to input invalid data? Are suitable error messages displayed to the user when problems occur?
    3. *database server*. Does the script create table(s) required for the application without manual intervention?
- does your documentation provide enough detail so that someone can run/test your solution without assistance. Do the internal comments describe the system in enough detail that another programmer can understand it? Does running the JavaDocs utility provide meanful documentation for developers?