David Reidenbaugh (dmr117)

Eric Copeland (ejc76)

**Team04 Report for CS1555 HW5**

User Manual for Forest Registry Interface

**Step 1:** Extract all files from team04.zip. Keep the sql file and all java files in the same folder.

Note: The file “postgresql-42.2.18” should be included, but if it is not download it at the following link:

https://jdbc.postgresql.org/download/postgresql-42.2.18.jar

**Step 2:** Run the SQL file team04.sql in order to create a new schema

**Step 3:** Compile all java files

Note: This can be accomplished by opening command prompt (or another terminal) and navigating to the directory where these files are stored. All java files will compile after you compile the file “ForestRegistryInterface.java” using the following command:

javac -cp "postgresql-42.2.18.jar;." ForestRegistryInterface.java

**Step 4:** While in the directory, run the forest registry user interface by typing the following command in your terminal:

java -cp "postgresql-42.2.18.jar;." ForestRegistryInterface

Step5: You will be presented with a list of potential actions to take. Typing the following numbers will allow you to perform the corresponding actions:

1: Add Forest

2: Add Worker

3: Add Sensor

4: Switch Workers' Duties

5: Update Sensor Status

6: Update Forest Covered Area

7: Find Top-K Busy Workers

8: Display Sensors Ranking

9: Show Options Again

0: Quit

System Limitations (Missing or Non-Implemented Functionality/Assumptions)

We believe that our submission meets all functionality requirements for the tasks. Some notable limitations and behaviors of the system based on our interpretation of the instructions include:

* When adding a forest, a forest will only be added to the FOREST table if it can also be added to the COVERAGE table.
* When doing any update to the area attribute of COVERAGE, the sum of percentage values for a particular forest is not allowed to go above 1 (100%).
* All forests, worker names, and state abbreviations are converted to strings with all capital letters upon insertion.
* forest\_no and sensor\_id are incremented using triggers (rather than the serial data type).
* When adding sensor, the user must specify ‘null’ for the maintainer’s ID if there is not supposed to be a maintainer (rather than the absence of a maintainer being caught by a trigger).
* If two workers who have no sensors get their duties swapped, no special message is displayed (it is assumed that it is fine to swap nothing with nothing).
* Updating a sensor (reading its status) does not involve moving the sensor to a new location. It is assumed that the user only wants to read its status.
* For top-k workers, a max of ‘k’ workers will be displayed (if there are more than ‘k’ top workers, they are not all displayed ). In the result of a tie workers with lower alphabetical names will be given preference.

Possibilities for Improvement

Some notable possibilities for improvement include:

* Task 5 (update sensor status), could be modified so that it first asks the user whether they want to read, move, or remove the sensor before taking the appropriate action.
  + This way sensors could be easily deleted or relocated.
* For task 7 (top-k workers), post-action messages could be displayed for the following cases.
  + If there are more than k workers tied, a message could print to the user telling them how many additional workers weren’t displayed but are part of that tie.
  + Rather than telling users upfront that there may be less than ‘k’ workers displayed, this could be handled in a post-action error message.
* In addition to task 7 (top-k workers), it could be convenient to also have functionality to retrieve the least busy workers. This way some duties could be swapped between the busiest and least busy to spread out the workload.
* For task 8 (display sensor rankings), we could modify this so that it displays the busiest sensors in a specified time period.
  + This would be helpful since a group of sensors might have been extremely busy in the past, but relatively inactive more recently.