Learning Team: Programming Improvement Week 5

Jason Garrick, Jonathon McNeil, Reuben Ellis

PRG/421

August 14, 2016

Edward Spear

Animal Database Interaction Modification

We decided to use Reuben's Animal Database Interaction project to optimize the program and display the use of threads within the program. One of the main changes made to the application is the addition of the animalDirectoryViewing class. The class implements Callable<String> so animalDirectoryViewing is usable as a thread and contains thread functionality. Any class which implements Callable must Override the call method and the method must match the type of the Callable which in this case is the String type. The call method assigns the current thread number to a variable and contains an if-else-if statement which checks the number of a thread and assigns an animal name to a variable respectively. The variable containing the animal name is a parameter for the method call viewAnimalFromDatabase. The viewAnimalFromDatabase method replaces the same process which was previously within the AnimalDatabaseInteraction class. The method contains the keyword synchronized, which only allows one thread to access the database and print the results rather than all four printing results at the same time. We chose to implement the synchronized results because while testing we noticed some of the animal information was printing with another animal information block and the data was not displaying in the console together. The method initializes a variable for the database connection, the PreparedStatement and a query which gets the values from the animal table based on the animal name. Within a try-catch-finally block, the Connection variable calls the mySQLDatabaseConnection method which checks to see if the drivers exist for the MySQL connection and if they do, then the variable receives the database connection details and user credentials and establishes a connection. The PreparedStatement executes the query for the animal table data and then assigns a value to the where clause where the value is the animal name passed from the call method. A ResultSet receives the values from the database, and if a record exists, the values print to the console for the user to view. Once the function is complete, the method closes the PreparedStatement and Connection before returning to the call process and completing the class call and returning to the main method.

The changes within the Main method are in how the user can view animal information from the database. The checks are in place for the application to get input from the user and whether the user enters view or add, but now if the user enters 'view' the data populates differently. An ExecutorService variable initializes, and the method newFixedThreadPool(4) creates four threads for use in retrieving data from the database. The program initializes four instances of the animalDirectoryViewing class one instance for each of the four threads. The next step is for the submit method of the ExecutorService to place a call to the class using each of the four instances. The call goes through the class functionality which receives a full description above. Once the threads are complete, a try-catch-finally block waits two seconds for the thread while the thread completes the current work and then if the thread flag is not indicative of a terminated status, the ExecutableService method shutdownNow ends the thread. This step ensures the application can close properly once the user is done and attempts to enter quit because a program will not close while threads are performing work or if threads are still alive. A Thread.sleep(6000) is necessary before the message asking the user to enter quit to end the program or anything else to continue because this message may appear before the animal data if some threads take longer than others. The Thread.sleep places the main thread on hold for 6 seconds while the other threads complete their work.

As a last change, we added a data export file of the MySQL schema animalDirectory and the animal table, so users can import the schema, table, and data within the table from any location and on any machine which contains the MySQL Workbench. The user only has to open MySQL Workbench and then connect to a server. Then the user can click the Tools tab and click data import and browse for the data file containing the schema. With these simple steps, the user will have the database in a usable format and the application will function correctly, so the user does not have to create a schema and table with categories to try and use the application. The project did not originally contain the file, and we felt it was necessary for ease of use and to provide a finished solution to the user.

The multi-threading within the project is only one of many ways the application can use threads to perform actions and automate functions. Another alternative could have been to continue to accept user input and have a thread pass the animal name from the user, but only after the user finishes entering in animal names for viewing. We chose to use the threads in their current functionality because it was a definitive way to display the use of multiple threads within an application.

References

Laird, C. (2013). Modern threading for not-quite-beginners | Java World. Retrieved from http://www.javaworld.com/article/2078679/java-concurrency/java-concurrency-modern threading-for-not-quite-beginners.html

Sierra, K. & Bates, B. (2015). OCA/OCP Java SE 7 Programmer I & II Study Guide (Exams 1Z0 803 & 1Z0-804. New York: McGraw-Hill Education.