Project: MP1
Course: ITM-411
Author: Brian T. Bailey

Project Description:

The object of this project was to create an object model and do some minor data analysis on a set of records of Intraplate Earthquakes provided by the USGS at their website listed below. The data included events from the year 495 to 2003.

For this project we needed to produce an object model with an abstract superclass called EarthquakeRecord and subclasses that extend that superclass for each continent called ContinentEarthquakeRecord. Also using an interface we needed to implement 4 methods in the superclass for finding the oldest, newest, minimum and maximum earthquakes per continent.

The output for the application needs to display a few pieces of data to both the screen and a data file. The data that needs to be output is all the earthquake records in string form, the oldest and newest earthquake per continent, the minimum and maximum earthquakes per continent and a table of tectonic-association by continent.

http://earthquake.usgs.gov/research/data/scr_catalog.php

Installation, Compile and Run-Time Requirements:

This project was written in Java using JDK 1.6. The IDE used to write and compile the application was NetBeans 7.1 on the Macintosh platform. The computer used was an Intel core i7 MacBook Pro running OS X Lion.

The application folder can be copied to a folder on the system and opened in NetBeans. The project can then be build and run using the play button in netbeans. The application can also be run from the standard terminal. To do this the data directory needs to be copied to the dist directory. The application jar file can then be run with the command “java -jar mp1.jar” from within the dist directory.

Insights and Expected Results:

Overall I found this project to be fairly straight forward. The biggest problem was the data file itself. When first looking at the data file I noticed some records were split between two lines making it difficult to just read in one line per record. This problem with the data file was the only thing that really held me up for a while.

First I tried to find a way to read the data by just changing the delimiter character used. That was causing an issue because there were fields with spaces that also had newline characters erroneously in them. Another problem in the data file was some records did not have a field 35 or 36.

My next thought was to read until I saw the date column because it was the only column with 7 or 8 digits in it. I setup a regular expression, "[0-9]{7,8}", to search for that pattern. I planned on starting a

new record when I found that pattern. If I found that pattern before I filled all 36 fields I would fill the others with a default value. When I tested this I found the next issue. There is a date in another field, number 33. I had to make sure I wasn't starting records on that date.

That brought me on to my next idea. I would test for a date and if I found one I would test again and see if the next field had the characters hist or instr in them with a regular expression, "(?:.*hist.*)|(?:.*instr.*)". This appeared to be working at first. I was still having problems with the erroneous newline characters in with the fields with spaces. I tried not splitting on spaces or newlines. That caused a problem at the end of the line. I needed only the newlines in fields with spaces to go away. I decided to look at the data more closely in my text editor with invisible characters on.

What I found in the data was interesting. The only places there were the erroneous newline characters in the data they were immediately preceded by a space character. Also when looking at the newline characters at the end of the lines they were never preceded by a space. I did a quick find and replace in my text editor to remove the space newlines with just spaces and found that it fixed the data file and all the fields lined up correctly. I didn't want to edit the data file outside the application but now I knew what the problem was.

Since I would have to loop through and remove the newlines with my previous approach anyway, came up with the algorithm to fix the data file before I start parsing it. The algorithm entailed reading the file one line at a time and appending it to a StringBuilder object. I would also have to append a newline to the StringBuilder object since the act of reading the line removed it. After I had the entire file in the StringBuilder exactly like it was I converted it to a String. I then used the String.replaceAll() method and replaced all the space newlines with spaces and put the resulting String in a new variable. Now I could read one line at a time and get only one record. I read those lines into an ArrayList as a list of Strings. That ArrayList would be used for further parsing and processing.

Other than the data there was really only one thing I needed to figure out. That was how I could sort the data arrays. I found calling the Arrays.sort() method would sort based on the natural order of the objects. This was defined by the compareTo() method. I found I needed to implement the Comparable interface in my EarthquakeRecord object and override a compareTo() method. I implemented this method to compare the date field and return the proper value for the method. Now I could sort my arrays.

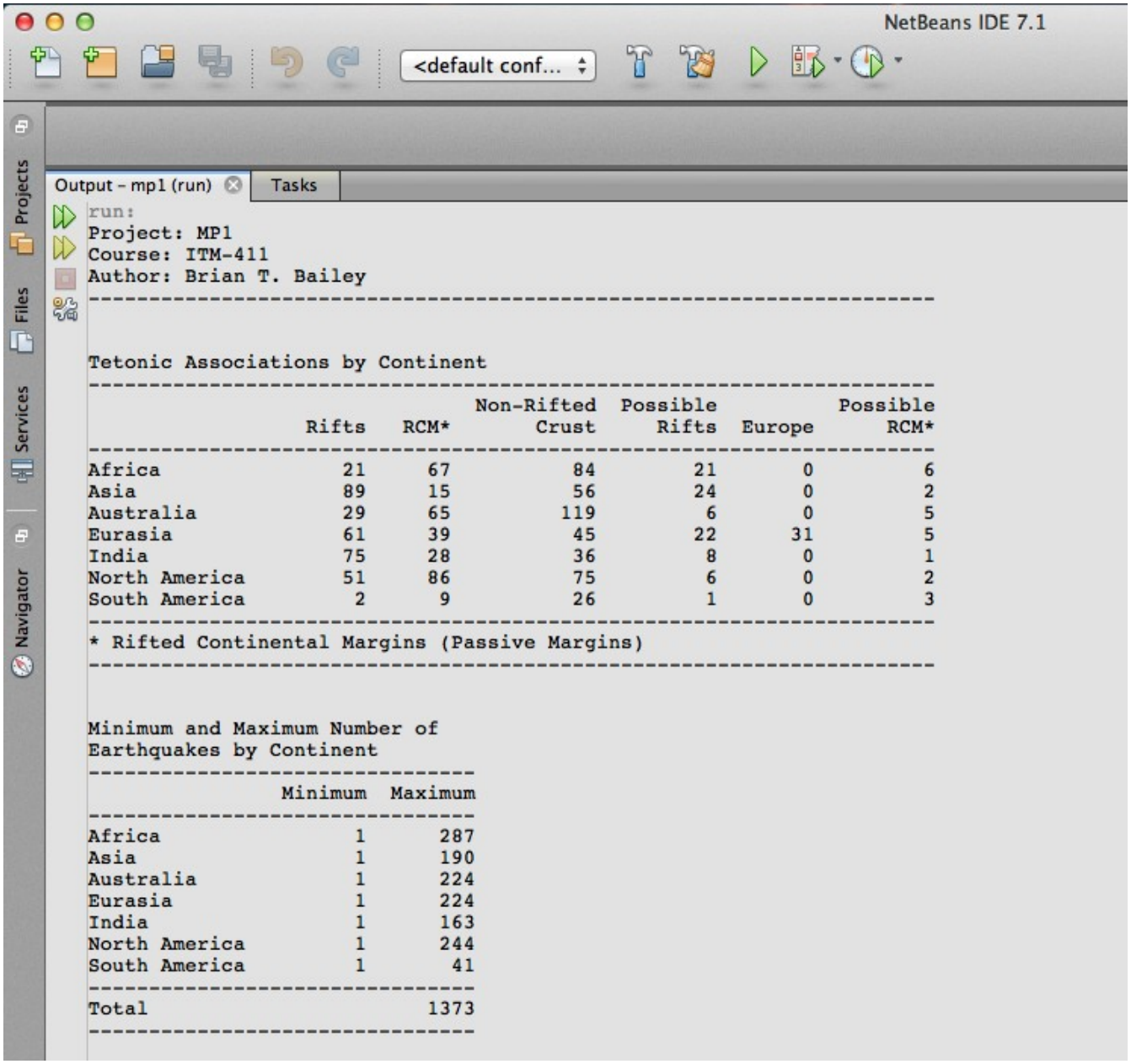
The final step was to implement all the methods to display the proper data to the screen and file. I went through and built methods to do all the output for me and called them from within the main method. I did this to make the code more readable and minimize the size of the main method.

Overall I found the project to be rather easy once I got past the data issues. It probably took me a little longer to get all the formatting of the output right because I was being rather picky on how it was looking. But all in all I think it turned out good.

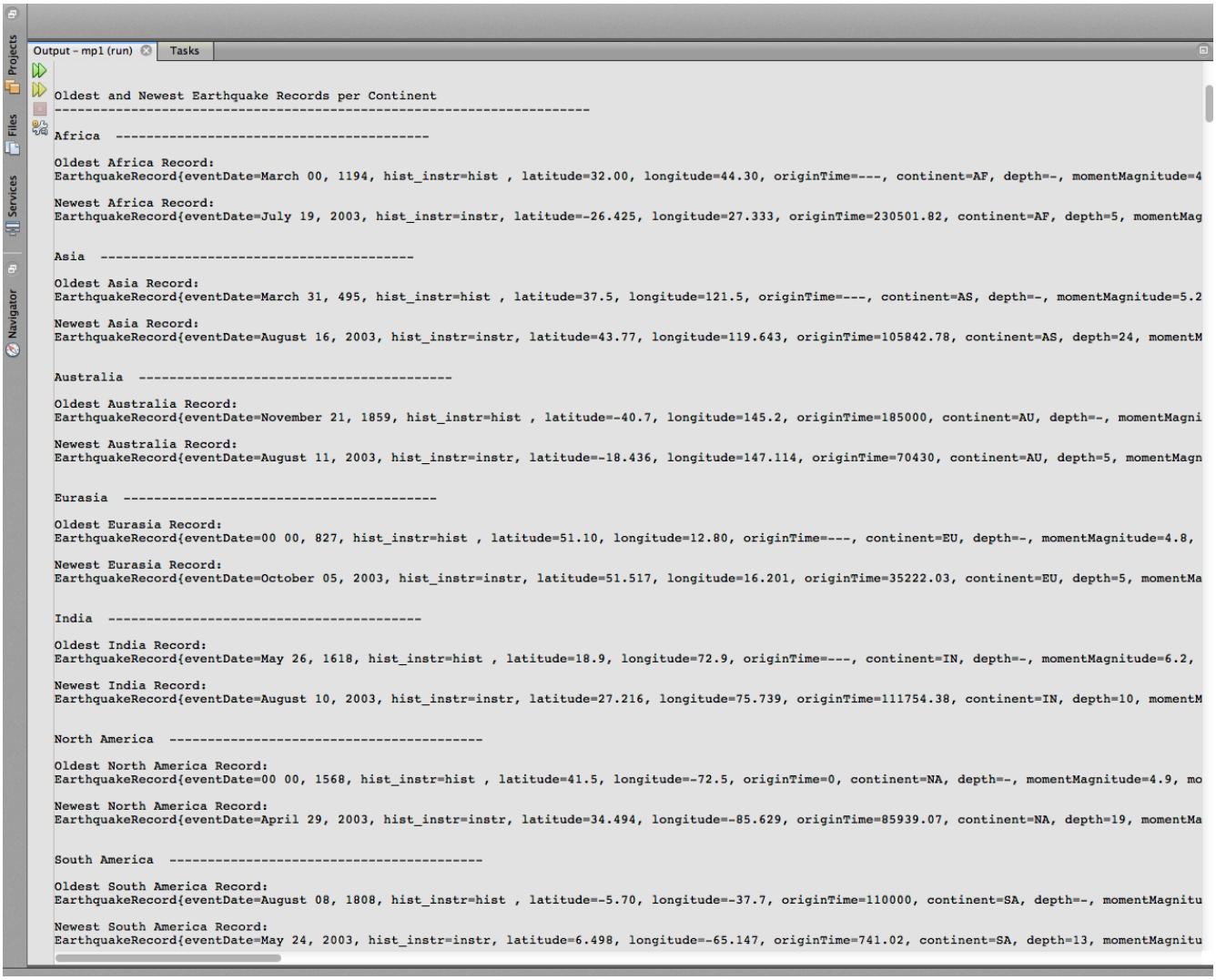
Screenshots Demonstrating Application:

Screenshots of the application running appear on the following pages.

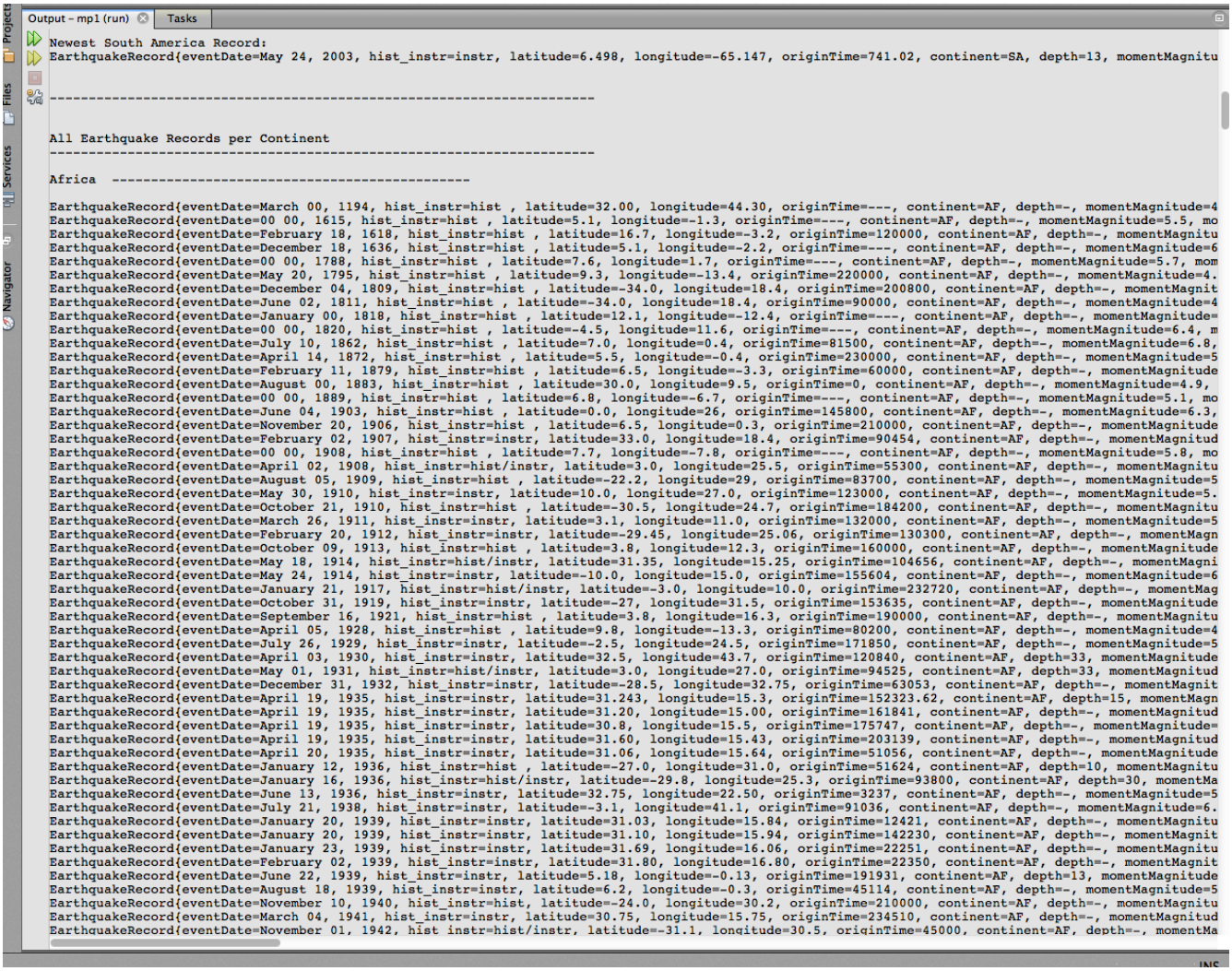
Screenshot of the application starting and displaying the table of tetonic associations by continent and a table of the minimum and maximum earthquakes per continent.



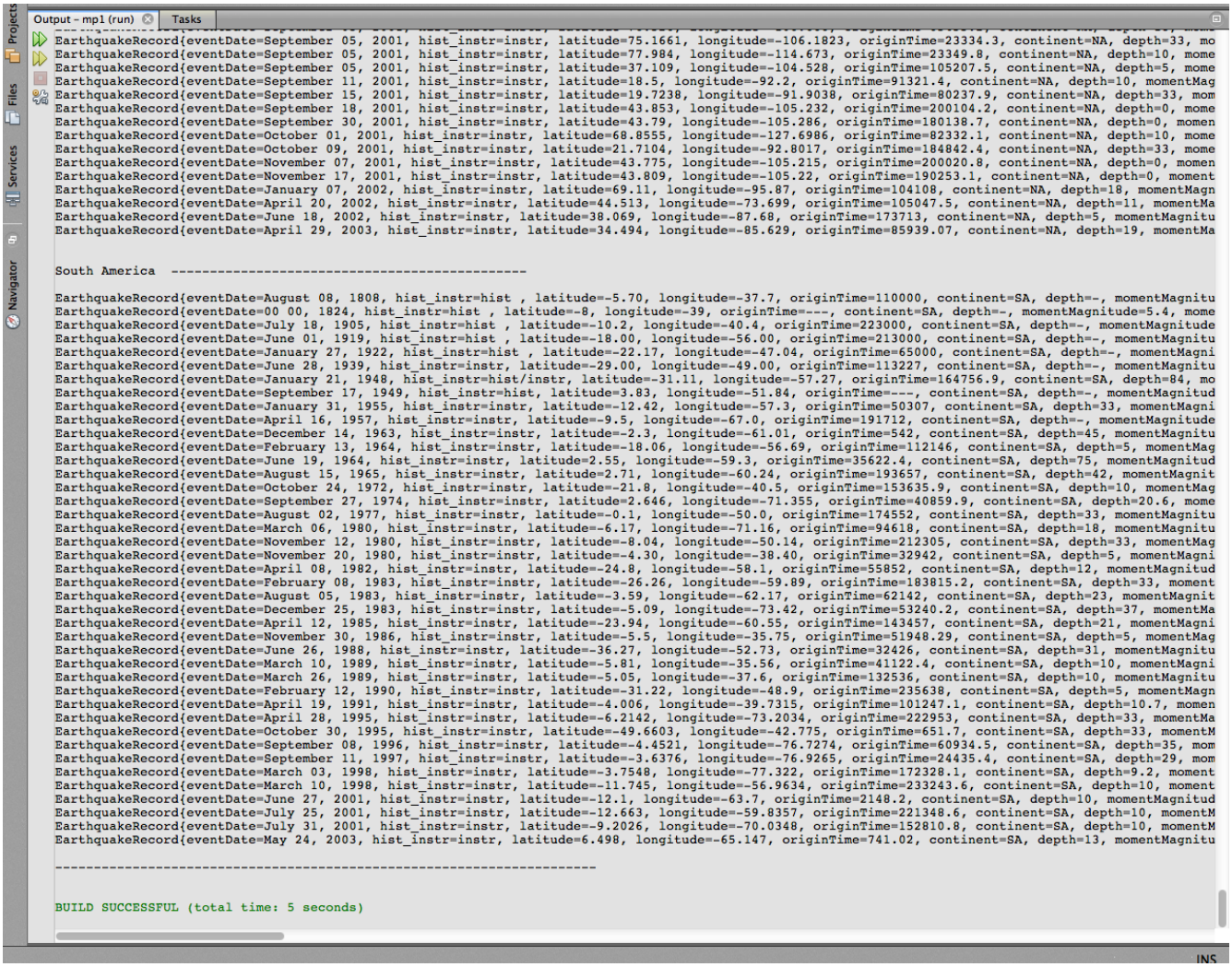
Screenshot of the application displaying the oldest and newest earthquake records grouped by the continent where they happened.



This screenshot shows the application starting to display all the earthquake records. They are output grouped by continent.



This screenshot shows the end of the list of all records and the indication that the build was successful and the application finished running.



This final screenshot shows the mp1_out.txt file produced by this application opened in a text editor.