Programming Exercise: Filtering Data

For the following assignments, you will start with the files provided, using most of the classes, and modifying only a few of them, and create new classes. First there are several classes provided:

- The class **Location**, from the Android platform and revised for this course, a data class representing a geographic location. One of the constructors has parameters **latitude** and **longitude**, and one of the public methods is **distanceTo**.
- The class **QuakeEntry**, which has a constructor that requires latitude, longitude, magnitude, title, and depth. It has several get methods and a **toString** method.
- The class **EarthQuakeParser**, which has a read method with one String parameter that represents an XML earthquake data file and returns an ArrayList of QuakeEntry objects.

There are several new classes and an interface that are provided for you.

- The interface **Filter**, which has one method signature named **satisfies**. Note that **satisfies** is not defined. Any class you write that implements Filter must provide the satisfies method.
- The class **MinMagFilter** that implements Filter. This class has a private double variable named **magMin**, representing the minimum magnitude of earthquakes to consider for filtering. This class has a constructor with one parameter named **min** to initialize **magMin**, and a **satisfies** method that returns true if its QuakeEntry parameter **qe** has a magnitude greater than or equal to **magMin**.
- The class **EarthQuakeClient2**, which has been started for you. It contains several methods including the following.
  - The method **filter** that has two parameters, an ArrayList of type QuakeEntry named **quakeData**, and a Filter named **f**. This method returns an ArrayList of QuakeEntry's from **quakeData** that meet the requirements of Filter **f**'s **satisfies** method.
  - The void method **quakesWithFilter** that has no parameters. This method creates an EarthQuakeParser to read in an earthquake data file, creating an

ArrayList of QuakeEntrys. It then creates a MinMagFilter with minimum magnitude 4.0, and then calls the **filter** method with the MinMagFilter to create an ArrayList of QuakeEntry's that have magnitude 4.0 or greater.

Assignment 1: Implementing Filters

In this assignment you will create several new filters and add code to test them.

Specifically, for this assignment, you will:

- Write the class **MagnitudeFilter** that implements Filter. This class should include private instance variables for a minimum and maximum magnitude, a constructor to initialize those variables, and a **satisfies** method that returns true if a QuakeEntry's magnitude is between the minimum and maximum magnitudes, or equal to one of them. Otherwise it should return false.

- Write the class **DepthFilter** that implements Filter. This class should include private instance variables for a minimum and maximum depth, a constructor to initialize those variables, and a **satisfies** method that returns true if a QuakeEntry's depth is between the minimum and maximum depths or equal to one of them. Otherwise it should return false.

- Write the class **DistanceFilter** that implements Filter. This class should include private instance variables for a location and a maximum distance, a constructor to initialize those variables, and a **satisfies** method that returns true if a QuakeEntry's distance from the given location is less than the specified maximum distance. Otherwise it should return false.

- Write the class **PhraseFilter** that implements Filter. This class should include two private instance variables for 1) a String representing the type of request that indicates where to search in the title and has one of three values: ("start", "end", or "any"), and 2) a String indicating the phrase to search for in the title of the earthquake (Note the title of the earthquake can be obtained through the **getInfo** method). This class also has a constructor to initialize those variables, and a **satisfies** method that returns true if the phrase is found at the requested location in the title. If the phrase is not found, this method should return false.

- Modify the code in the **quakesWithFilter** method in the EarthQuakeClient2 class to filter earthquakes using two criteria, those with magnitude between 4.0 and 5.0 and depth between -35000.0 and -12000.0. You'll need to use both the MagnitudeFilter and the DepthFilter. Use one and then use the other on the result from the first. After writing this

method, when you run your program on the file **nov20quakedatasmall.atom**, you will

see the following two earthquakes meet that criteria:

```
(8.53, -71.34), mag = 5.00, depth = -25160.00, title = 5km ENE of Lagunillas,
Venezuela
(38.27, 142.53), mag = 4.60, depth = -30500.00, title = 109km E of Ishinomaki,
Japan
```

- Comment out the previous code in **quakesWithFilter** in the EarthQuakeClient2 class
  and add additional code to the **quakesWithFilter** method in the EarthQuakeClient2
  class to filter earthquakes using two criteria, those that are 10,000,000 meters (10,000
  km) from Tokyo, Japan whose location is (35.42, 139.43), and that are in Japan (this
  means "Japan" should be the last word in the title). After writing this method, when you
  run your program on the file **nov20quakedatasmall.atom**, you will see the following two
  earthquakes meet that criteria:

```
(26.38, 142.71), mag = 5.50, depth = -12890.00, title = 91km SSE of
Chichi-shima, Japan
(38.27, 142.53), mag = 4.60, depth = -30500.00, title = 109km E of Ishinomaki,
Japan
```

Assignment 2: MatchAllFilter and Modification to the Interface

In this assignment you will write a class named **MatchAllFilter** that can store and apply many filters, and you will also modify the **Filter** interface to store the name of the filter.

- Write the class **MatchAllFilter** that implements Filter. This class has a private ArrayList of Filters that is created in the constructor that has no parameters. This class has two methods, 1) a void method named **addFilter** with one Filter parameter that adds the Filter to its private ArrayList, and 2) a method named **satisfies** that has one QuakeEntry parameter and returns true if the QuakeEntry satisfies all the filter conditions, otherwise it returns false.

- Write a new void method named **testMatchAllFilter** in the EarthQuakeClient2 class. This method reads in earthquake data from a source and stores them into an ArrayList of type QuakeEntry. Then it prints all the earthquakes and how many earthquakes that were from the source. You should read in earthquakes from the small file **nov20quakedatasmall.atom**, print all the earthquakes and also print how many there are. After this works you should comment out the printing of all the earthquakes, but continue to print out the total number of earthquakes read in. Then create a MatchAllFilter named **maf** and use the **addFilter** method to add three Filters to test the magnitude between 0.0 and 2.0, to test the depth between -100000.0 and -10000.0, and if the letter "a" is in the title. Then use **filter(list, maf)** to use all three filters and print out the resulting list of earthquakes. You will see the following two earthquakes meet that criteria:

```
(33.54, -116.66), mag = 0.30, depth = -10410.00, title = 2km SE of Anza,
California
(63.25, -150.43), mag = 1.70, depth = -99900.00, title = 75km WSW of Cantwell,
Alaska
```

- Write a new void method named **testMatchAllFilter2** in the EarthQuakeClient2 class. This method should be identical to the method **testMatchAllFilter**, but will create different Filters. You should read in earthquakes from the small file **nov20quakedatasmall.atom**.Then create a MatchAllFilter named **maf**, and use the **addFilter** method to add three Filters to test the magnitude between 0.0 and 3.0, to test for the distance from Tulsa, Oklahoma at location (36.1314, -95.9372) is less than 10000000 meters (10000 km), and if the substring "Ca" is in the title. Then use **filter(list,**

maf) to use all three filters and print out the resulting list of earthquakes. You will see the following seven earthquakes meet that criteria:

```
(33.54, -116.66), mag = 0.30, depth = -10410.00, title = 2km SE of Anza,
California
(63.44, -147.62), mag = 1.60, depth = -7400.00, title = 66km E of Cantwell,
Alaska
(36.27, -121.66), mag = 2.00, depth = -7630.00, title = 28km SSE of Carmel
Valley Village, California
(63.25, -150.43), mag = 1.70, depth = -99900.00, title = 75km WSW of Cantwell,
Alaska
(35.00, -118.21), mag = 1.30, depth = 1010.00, title = Quarry Blast - 7km SSW
of Mojave, California
(49.39, -120.44), mag = 2.40, depth = -20.00, title = Explosion - 8km SSE of
Princeton, Canada
(34.05, -117.36), mag = 1.20, depth = 1040.00, title = Quarry Blast - 4km WNW
of Grand Terrace, California
```

- Modify the **Filter** interface to include a new method named **getName** that returns the name of the filter. The line added to the Filter interface should be:

  ```
  public String getName();
  ```

  What changes need to be made to all the Filter classes? The user should be able to specify what they want the name of the filter to be when they create a specific filter. For the **MatchAllFilter** class, a **getName** method should return a String of all the Filter names in its ArrayList.

- Add to the end of the method **testMatchAllFilter** a call to the MatchAllFilter **getName** method to print out all the Filter names used. For the example above, printing "Filters used are: " followed by the call to **getName** could result in the output:

  ```
  Filters used are: Magnitude Depth Phrase
  ```