Classes, Inheritance, Abstract Classes, and Polymorphism

Assignment 8: Due date/time is Thursday, April 14th, at 11:59 PM.

Part 1: ***(15 Points)***

Create a java project that has the following units:

- A class named **Document** that contains a member variable of type **String** named **text** that stores any textual content for the document. The class has the proper constructors, getters, and setters, and has the toString, equals method overridden.
- Define a class for **Email** that is derived from **Document**, and that includes the following member variables: **sender, recipient**, and **title** (email subject) of an e-mail message. Implement appropriate getters/accessors. The body of the e-mail message should be stored in the inherited variable **text**.
- Similarly, define a class for**File** that is derived from **Document**, and that includes a member variable for the pathname (or only file name). Implement appropriate getters/accessors methods for the pathname member variable. Make sure that you override the **toString** and **equals** methods for derived classes as well.
- Finally, create a static method **ContainsKeyword** (in a driver class **DocumentApp**) that would use the inheritance-based Polymorphism to receive an object of a class (Email, File, or Documents) and returns **true** if the object contains a specified keyword (string) sent as a parameter to the static method as well.

For example, the following method call (from the main method in class DocumentApp) would return **true** if the text of the ***emailObj*** contains the word (substring) **money**:

ContainsKeyword(*emailObj*, "money");

- To test your program, make sure that the data is read from test files.

That is, you need to make up some files that include emails or files' information. The program prompts the user to enter the file name and then the file is read, and data is stored in a proper object/s.

Part 2: ***(15 Points)***

Modify class **Document** to be an Abstract Class then use it to derive the Email and File classes. The Abstract class has an extra method signature **fileLength**() (abstract method) that returns the length of the text (how many characters) saved in objects of the derivate classes. That is, derived classes need to provide concrete implementations for the **fileLength**() method.