CS 1400-03 Introduction to Programming and Problem Solving
Coding Practice #11
(Due: 11:59 PM, Friday, 4/30/2021)

Except Coding Practice #1, I will not grade your coding practice submissions. Instead, they will be treated as participation points. On blackboard, you will receive full points as long as you work on the exercises, which don't necessary mean they are all correct. Please check your own programs carefully and make sure they do generate the desired output.

Objectives:

- Be able to use inheritance, polymorphic reference, the `super` keyword
- Be able to use overloading and overriding
- Be able to test and debug a program

Change your working directory to `cs1400/codingPractice` for this assignment.

Task #1 Document, Email, File

Define a class named `Document` that contains an instance variable of type `String` named `text` that stores any textual content for the document. Define a one-argument constructor to set this value and a no-argument constructor to initialize it to empty string. Create a method named `toString` that returns the text field.

Next, define a class for `Email` that is derived from `Document` and includes instance variables for the `sender`, `recipient`, and `title` of an email message. Define a 4-argument constructor to set these values and the body of the email message should be stored in the inherited variable `text`. Define a no-argument constructor to initialize everything to empty string. Implement appropriate accessor and mutator methods. Redefine the `toString` method to concatenate all text fields.

Similarly, define a class for `File` that is derived from `Document` and includes an instance variable for the `pathname`. Define a 2-argument constructor to set these values and the textual contents of the file should be stored in the inherited variable `text`. Define a no-argument constructor to initialize everything to empty string. Implement appropriate accessor and mutator methods. Redefine the `toString` method to concatenate all text fields.

The following driver program creates several sample objects of type `Email` and `File`. The method `containsKeyword` in the driver program will return true if the parameter object contains the specified keyword in the text.

```
public class DocumentTest
{
   public static void main(String[] args)
   {
      Email email1= new Email("Body about programming in Java",
                              "Larry", "David", "Programming");
      Email email2 = new Email("Body about running marathons",
                               "Mary", "Emily", "races");
```

```
      File file1 = new File("Contents about some Java file",
                            "/user/file.txt");
      File file2 = new File("Contents about marathon races",
                            "run.txt");

      System.out.println("Email1 --\n" + email1);
      System.out.println("Email2 --\n" + email2);
      System.out.println("File1 --\n" + file1);
      System.out.println("File2 --\n" + file2);

      System.out.println("Which contains Java?");
      if (ContainsKeyword(email1,"Java"))
         System.out.println("  Email1");
      if (ContainsKeyword(email2,"Java"))
         System.out.println("  Email2");
      if (ContainsKeyword(file1,"Java"))
         System.out.println("  File1");
      if (ContainsKeyword(file2,"Java"))
         System.out.println("  File2");
   }
   public static boolean ContainsKeyword(Document docObject,
                                         String keyword)
   {
      if (docObject.toString().indexOf(keyword) >= 0)
         return true;
      return false;
   }
}
```

The following is **required** output when running the program:

```
fcsang@fluffy ~/cs1400/codingPractice $ java DocumentTest
Email1 --
Sender: Larry
Recipient: David
Title: Programming
Content: Body about programming in Java
Email2 --
Sender: Mary
Recipient: Emily
Title: races
Content: Body about running marathons
File1 --
Pathname: /user/file.txt
Body: Contents about some Java file
File2 --
Pathname: run.txt
Body: Contents about marathon races
Which contains Java?
  Email1
  File1
```

Task #2 Polymorphism, Overloading and Overriding

Given the following code for the base class:

```
import java.io.*;
public class FileIO
{
   // This method opens file s,
   // reads and outputs to screen its contents
   void fileRead(String s) throws IOException
   {
      System.out.println("fileRead in the base class reached");
   }

   // This method opens file s
   // and writes array a to it
   void fileWrite(String s, String[] a) throws IOException
   {
      System.out.println("fileWrite in the base class reached");
   }
}
```

Write a class named `FileIOSubClass` that inherits from `FileIO` and override each method so they actually work. If input file does not exist, give an appropriate error message and terminate the program. In addition, you should overload `fileRead` so that it can take a `File` object as an argument instead of a string.

Write a test driver program `FileIOTest.java` that creates a `FileIO` reference variable pointing to a `FileIOSubClass` object, and demonstrates all of the capabilities and features of the `FileIOSubClass`.

The following are sample executions:

```
fcsang@fluffy ~/cs1400/codingPractice $ java FileIOTest
Please enter an input file name: grade.txt
Test fileRead(String) --
Error: input file grade.txt does not exist

fcsang@fluffy ~/cs1400/codingPractice $ java FileIOTest
Please enter an input file name: grade1.txt
Test fileRead(String) --
10
30
50
70
90
100
80
60
40
20
```

```
Please enter an output file name: out.out
enter how many lines you have to print to the output file: 3
enter line 1: CS1400 Coding Practice #11
enter line 2: Task #2
enter line 3: overloading vs. overriding demo

Test fileWrite --
check output file: out.out

Please enter another input filename: grade3.txt
Test fileRead(File) --
100
100
100

fcsang@fluffy ~/cs1400/codingPractice $ cat out.out
CS1400 Coding Practice #11
Task #2
overloading vs overriding demo
```

Submission:
Generate a `script` file `practice11.txt` with appropriate time stamps and the following steps visible:

1) a `pwd` to show the current working directory
2) a `ls -l` to show in long format the files in your `cs1400/codingPractice` directory
3) display `Document.java, Email.java, File.java, DocumentTest.java`
4) compile `DocumentTest.java`
5) run `DocumentTest`
6) display `FileIO.java, FileIOSubClass.java, FileIOTest.java`
7) compile `FileIOTest.java`
8) run `FileIOTest`

Submit the script file `practice11.txt` on Bb, under the Coding Practice Folder, Practice #11 link.