LAB: EXPERIMENTING WITH ACCESSIBILITY/VISIBILITY

Getting Ready: Before going any further you should:

1. Make a directory on your `N:` drive for this lab.
2. Setup your development environment.
3. Create a file named `Document.java` that contains the following:

Document.java

```
import java.util.*;

/**
 * A very simple Document class that can be used to explore
 * issues related to accessibility/visibility
 *
 * @author  Prof. David Bernstein, James Madison University
 * @version 1.0
 */
public class Document
{
    // Note that the attributes are private
    private String       delimiters, text;


    /**
     * Explicit Value Constuctor
     *
     * @param text   The text of the document
     */
    public Document(String text)
    {
         this.text = text;
         delimiters = " ,.;:!?\t\n\r";
    }


    /**
     * Append additional text to the end of this Document
     *
     * @param addition   The text to append
     */
    public void append(String addition)
    {
         text = text + addition;
    }


    /**
     * Get the characters used to delimit words
     *
     * Note: This method is public but there is no reason
     *       for non-child classes to have access
     *
     * @return  A String containing the delimiters
     */
    public String getDelimiters()
    {
         return delimiters;
    }


    /**
     * Get a description of this Document that
     * includes a statistical summary
     *
     * @return  The description
     */
    public String getDescription()
    {
         String       result;

         result = "Contains " + getWordCount() + " word(s).";

         return result;
    }


    /**
     * Get the text of this Document
     *
     * @return  The text
     */
    public String getText()
    {
         return text;
    }



    /**
     * Get the number of words in this Document
     *
     * @return  The number of words
     */
    public int getWordCount()
    {
         int                count;
         StringTokenizer    tokenizer;

         tokenizer = new StringTokenizer(text, delimiters);

         count = tokenizer.countTokens();

         return count;
    }

}
```

4. Create a file named `FormattedDocument.java` that contains the following:

FormattedDocument.java

```
import java.util.*;

/**
 * A very simple FormattedDocument class that can be used to explore
 * issues related to accessibility/visibility
 *
 * Compared to its parent, this class modifies:
 *
 *     1. The getText() method (provides line-wrap at word boundaries)
 *     2. The getDescription() method (provides additional detail)
 *
 * Compared to its parent, this class adds:
 *
 *     1. A maxWidth attribute (used for line-wrap)
 *     1. A setWidth() method
 *
 * @author  Prof. David Bernstein, James Madison University
 * @version 1.0
 */
public class FormattedDocument extends Document
{
    private int          maxWidth;


    /**
     * Explicit Value Constuctor
     *
     * @param text   The text of the document
     * @param width  The maximum width of a line
     */
    public FormattedDocument(String text, int width)
    {
         super(text);

         maxWidth = width;
    }


    /**
     * Get a description of this Document that
     * includes a statistical summary
     *
     * @return  The description
     */
    public String getDescription()
    {
         int          count;
         String       result, temp;

         temp   = super.getText();
         count  = getWordCount();

         result = "This document has " + count;
         if (count == 1) result += " word ";
         else            result += " words ";

         result += "and at least " + temp.length()/maxWidth +
                   " lines.";

         return result;
    }



    /**
     * Get the text of this Document
     *
     * @return  The text
     */
    public String getText()
    {
         int                 currentWidth, wordWidth;
         String              delim, result, temp, word;
         StringTokenizer     tokenizer;
         // Construct the tokenizer
         temp  = super.getText();
         delim = super.getDelimiters();
         tokenizer = new StringTokenizer(temp, delim);

         // Initialization
         currentWidth =  0;
         result       = "";

         // Loop through the words in the text
         while (tokenizer.hasMoreTokens())
         {
             word = tokenizer.nextToken();
             wordWidth = word.length();


             if ((currentWidth + wordWidth + 1) > maxWidth)
             {
                  // Time for a new line
                  result += "\n" + word;
                  currentWidth = wordWidth;

             } else {

                  // Put this word on the current line
                  if (currentWidth == 0)
                {
                      // First word on the line
                      result += word;
                      currentWidth = currentWidth + wordWidth;

                  } else {

                      // Not the first word on the line
                      result += " " + word;
                      currentWidth = currentWidth + wordWidth + 1;
                  }
             }
         }
         result += "\n";

         return result;
    }




    /**
     * Set the maximum width (in characters)
     * of a line
     *
     * @param width  The maximum line width
     */
    public void setWidth(int width)
    {
         maxWidth = width;
    }
}
```

5. Create a file named `Driver.java` that contains the following:

Driver.java

```
/**
 * A driver for testing the Document and FormattedDocument
 * classes
 */
public class Driver
{
    /**
     * The entry point of the application
     *
     * @param args     The command line arguments
     */
    public static void main(String[] args)
    {
         Document                 doc;
         String                   text;


         text = "George is a little monkey, "+
               "and all monkeys are curious. "+
               "But no monkey is as curious "+
               "as George.";


         doc = new FormattedDocument(text, 20);

         System.out.println();
         System.out.println(doc.getDescription());
         System.out.println();
         System.out.println(doc.getText());
    }
}
```

6. Make sure you understand the classes you just created.

Part I: This part of the lab is a review of material from earlier in the semester.

1. Compile all of the classes and execute the driver. Did you get the output you expected?

2. Don't change the declaration of the variable named `doc` but change the line containing `doc = new Document(text);` to `doc = new String(text);`. Re-compile the driver. What error was generated? Why?

3. Don't change the declaration of the variable named `doc` but change the line that now contains `doc = new String(text);` to `doc = new FormattedDocument(text, 20);`.

4. Re-compile the driver. Why did it compile even though there appear to be incompatible types?

5. Re-execute the driver. Did it output what you expected? If so, why? If not, why not?

6. The `getText()` method in the `FormattedDocument` class contains the line `temp = super.getText();`. Explain this line of code.

7. Replace the line `temp = super.getText();` with the line `temp = getText()`. Re-compile this class and re-execute the driver. What happened and why?

Part II: In this part of the lab you will experiment with changing the accessibility/visibility of attributes and methods.

1. Make the accessibility of the `getDelimiters()` method in the `Document` class `private`. Re-compile only the `Document` class. What happens and why?

2. Re-compile only the `FormattedDocument` class. What happens and why?

3. Now make the `getDelimiters()` method `protected` and re-compile all of the classes. What happens and why?

4. What's the difference between the `public` version and the `protected` version? Which is better? Why?

5. Perform the same experiments with the `getWordCount()` method. Which version is better? Why?

6. Change the accessibility of the `delimiters` and `text` attributes in the `Document` class to protected. What changes can you now make to the `FormattedDocument` class? (Hint: Think about how the `FormattedDocument` class accesses these attributes.)

7. Do you like these changes? Why or why not?

8. Now that the `delimiters` attribute is `protected`, do you still need the `getDelimiters()` method?

9. Should either the `getDescription()` or `getText()` mehods in the `Document` class be protected? Why or why not?