CS 1301 - Homework 14

Due before Thursday, December 4, 1:50 *pm* on Vista (WebCT). You can resubmit your work up until the deadline. This assignment will count twice since it has extra problems. The assignment is now complete.

1. Write a program, **StringTest1.java** to test the static methods below.

   String allCombinations(String s) – This method returns *null* if the length of *s* is not 3. Otherwise, it returns a string with all possible pairs composed from characters in the *s*. For instance, if "ABC" is the input, then "ABACBC" is the output.

   String radialSubstring(String s, int center, int radius) – This method returns a substring of *s* composed of the characters between (inclusive) *center-radius* and *center+radius*. For instance, if the input is: ("Waffle House",5,3) then the output will be: "ffle ho". If the input requests characters before the beginning or after the end, then the method should return an empty string. The *center* and *radius* must be 0 or greater, otherwise, return an empty string.

   String getWordInString(String s, String[] words) – Return the first *word* in *words* that is found in *s*. Otherwise, return null. For instance, if the input is: "My horse is fast", { "car", "horse", "bike", "fast" } then the output is "horse".

2. Write a program, **PasswordChecker.java** to test the static methods below. Consider these password conditions:

   at least one lower case letter
   at least one upper case letter
   as least one digit
   at least one other character

   A *Level 1* password must meet at least 2 of the conditions while a *Level 2* password must meet at least 3 of the conditions.

   boolean isLevel1 (String password) – Length must be at least 6 and meet *Level 1* criteria.

   boolean isLevel2 (String password) – Length must be at least 6 and meet *Level 2* criteria.

   int passwordLevel (String password) – Returns 1 if *password* is a *Level 1* password, 2 if it is a *Level 2* password, and 0 otherwise.

3. Write a program, **CircleBuilder.java** to test the static method below.

   Circle[] makeCircles (String input) – Input can specify: *quantity* circle *radius.* For instance: "4 circle 2.5" means that this method should return an array of 4 circles, each with radius 2.5. However, the *quantity* and *radius* are both optional. If *quantity* is not present, then it is by default, 1 and the same for the radius. For example it the input is: "circle" then an array with 1 Circle of radius 1 is returned. Similarly, "circle 4.25", then an array with 1 Circle of radius 4.25 is returned. The input can be separated by spaces, comma, colon, semicolon, combinations of these with spaces. The input can also contain leading and ending spaces.

   Hints: How many different forms can the input come in? Write them down. How will your code tell the difference?

   There will also be a main that tests this method.

   The Circle class should also be written. It has a no-arg constructor and one that takes a radius. It has getArea, and toString methods only.

4. Write a program, **WordCounter.java** to test the static method below. Warning, this problem will require some thought! Hint: how many arrays will you need? (Answer 2, one for the "uniqueWords" and one for the "counts".

   void wordCount (String input) – Displays a list of the words in *input*, in order of occurrence, and their frequency. For simplicity, assume that there are no words with an apostrophe and ignore case. For instance, "The red king has the ring. Red is the color of the ring, the king says." would produce:

   ```
   the     5
   red     2
   king    2
   has     1
   ring    2
   is      1
   color   1
   of      1
   says    1
   ```

5. Write a program, **WordReverse.java** to test the static method below.

   String wordReverse (String input) – Use the StringBuilder class to build a string of the words in input, reversed. For simplicity, assume that words are simply separated by spaces. For instance, "The red king has the ring" would produce: "ring the has king red The".

6. Write a class, **Employee.java** that has read-only properties for name, hours worked, and pay rate. These values will be supplied via a constructor. The class also has a method to get the pay (hours worked * pay rate).

   Write a program **EmployeeTester.java** which will read names, hours worked, and pay rate from a text file (**worked.txt**, you must create this yourself) which will look like this:

   ```
   Dave 40 50.00
   Sarah 20 20.00
   Sue 10 15.00
   ...
   ```

   Your program must create Employee objects from this data and then write out a payroll sheet to a file (**payroll.txt**) which will look like this:

   ```
   Dave $2000.00
   Sarah $400.00
   Sue $150.00
   ...
   ```

   Yes, you don't "need" an Employee class for this simple task, but better to get the practice using objects.