600.107 Introductory Programming in Java, Summer 2015
Homework 6
Due: Friday, July 17th at 11:59pm

Please review the **Homework Guidelines** handout for this course before you begin.

Restriction: For all tasks in this assignment, you are NOT permitted to use arrays or ArrayLists.

Repetition Statements: "while", "for", or "do-while"- type statements may NOT be used in this assignment.

Task 1. As you likely recall, the letters in the English language which are vowels are `a`, `e`, `i`, `o`, and `u`. (Well, and sometimes `y`, but not for the purposes of this problem.) Write a Java class called `Vowels` which contains the following Javadoc comment and the **recursive** method it describes.

```
/**
 * Recursively determine the number of vowel characters present in a String.
 * This method counts both upper-case and lower-case vowels in the total,
 * and assumes that 'y' and 'Y' are consonants (and so does not count them
 * as vowels).
 *
 * @param    str   the String in which to count
 * @return         the number of question marks appearing in the argument
 */
public static int countVowels(String s)
```

This method should collect no input and should produce no output. The method must be written *recursively. Iterative methods which solve this problem will receive no points.* In addition, write a `main` method which collects as a `String` a line of input from the user, calls the `countVowels` method, and outputs the value that the method returns. Below is a single sample execution:

```
Please enter a phrase:  This IS rEAlly the best sentence evEr.
The number of vowels in that sentence is 11.
```

Task 2. Suppose we have a satellite in orbit. To communicate to the satellite, we can send messages composed of two signals, dot ($\cdot$) and dash ($-$). Dot takes 2 microseconds to send, and dash takes 3 microseconds to send. Imagine that we want to know the number of different messages $M(k)$ that can be sent in exactly k microseconds, without any "dead air".

For example, when k equals 6, the only messages that take exactly k microseconds to send are $\cdot\cdot\cdot$ or $--$. (Convince yourself that there aren't any more.) Therefore, the value of $M(6)$ is 2. Furthermore, the only messages that take exactly k equals 7 microseconds are: $\cdot\cdot-$ and $\cdot-\cdot$, and $-\cdot\cdot$. That is, $M(7)$ equals 3. In addition, $M(1)$ equals 0, and we say that $M(0)$ equals 1 (the empty message).

We observe the following about calculating $M(k)$:

- If k is larger than 3, we know that the message may start with a dot or a dash. If the message starts with dot, the number of possible length-k messages that can result is $M(k-2)$. If the message starts with a dot, the number of possible length-k messages that can result is $M(k-3)$. Therefore, the number of messages that can be sent in k microseconds in this case is $M(k-2)+M(k-3)$.
- If k is smaller than 4, the the value of $M(k)$ depends on the exact value of k: sometimes only a single message is possible, and sometimes zero messages are possible. (Remember, dead air is not allowed!)

Write a Java class called `Satellite` which contains the following Javadoc comment and the **recursive** method it describes.

```
/**
 * Recursively determine the number of different messages composed
 * of only dots and dashes that can be sent in exactly a specified time.
 * The calculation assumes that a dot takes 2 microseconds to send, and
 * that a dash takes 3 microseconds to send.
 *
 * @param    k   the length of time, in microseconds, that we have to fill
 * @return       the number of different messages which can be sent
 *               in exactly the length of time specified by the argument
 */
public static int countMessages(int k)
```

This method should collect no input and should produce no output. The method must be written *recursively*. *Iterative methods which solve this problem will receive no points.* In addition, write a `main` method which collects as an `int` a line of input from the user, calls the `countMessages` method, and outputs the value that the method returns. Below is a single sample execution:

```
Please enter an integer number of microseconds:  10
The number of different 10-microsecond-long messages is 7.
```

Task 3. Write a Java class called `CollapseAdjacentRepeats` which contains the following Javadoc comment and the **recursive** method it describes.

```
/**
 *  Recursively remove the adjacent duplicate characters in a String.
 *
 *  @param   original    the String from which duplicates will be removed
 *  @return                    the original but with adjacent duplicates removed
 */
public static String collapse(String original)
```

That is, the method returns a copy of its String argument with all adjacent duplicate characters removed. Note: the case of letters is important for this task: `A` and `a` are not considered duplicate characters, for example. This method should collect no input and should produce no output. The method must be written *recursively*. *Iterative methods which solve this problem will receive no points.* In addition, write a `main` method which collects as a `String` a line of input from the user, calls the `collapse` method, and outputs the value that the method returns. Below are two separate sample executions:

```
Enter a line, and I'll collapse the adjacent repeated characters in it:
JeSse the Hawaiian bookkeeper bought a balloon.
Without repeats:
JeSse the Hawaian bokeper bought a balon.
```

```
Enter a line, and I'll collapse the adjacent repeated characters in it:
   J     H        U!!!!!
Without repeats:
 J H U!
```

Turn-in. Before the due date listed at the top of this file, please submit via the course Blackboard site one complete `.zip` file named `HW6-jhed.zip` (where you replace 'jhed' with your own personal JHED) containing all `.java` files required for the above tasks. Be sure that all submitted code compiles! If you were unable to complete a task, please include as part of your `.zip` submission a text file named `README` explaining anything you'd like the graders to know. Don't forget to submit early and often!