Overview:

Listing 2.12 in the textbook, *Java: An Introduction to Problem Solving and Programming* by Walter Savitch, calculates and displays the number of pennies, nickels, dimes and quarters equal to a given whole number of cents from 1 to 99.

Note: I could post listing 2.12 as a .java file on the website, but part of the learning experience, at least at the beginning of learning Java, is to type it yourself. While you may already be skilled in typing English, typing Java is a bit different - skipping a semicolon or using a parenthesis in place of a curly bracket will create a Java program that does not run. Part of learning Java (not the most fun part, but an important part) is learning to see these tiny differences.

You must make a few modifications to Listing 2.12:

1) Your program must display the input dialog shown above (see section 2.5: JOptionPane in the textbook).

2) You must check that the data entered by the user follows the required syntax. If not, your program must display an error dialog and exit. The input is valid only if all of the following hold:
   - The input has the form pay:cost where pay and cost are both integers.
   - pay is a whole number of dollars (100, 200, 300, ...).
   - pay is less than or equal to 900 cents.
   - pay is greater than or equal to cost.
   - cost is greater than or equal to 5 and can be expressed using only quarters, dimes and nickels (i.e. 5, 10, 15, ... 95, 100, 105, ... 890, 895, or 900).

3) If the input is valid, then your program must display a message dialog similar to the one shown in listing 2.12. Your message dialog must show the
minimum number of nickels, dimes and quarters needed to provide the correct amount of change, where \( \text{change} = \text{pay} - \text{cost} \).

4) Your program must exit when the user closes the output message dialog.

**Import Packages:**

Your program must import `javax.swing.JOptionPane`. **No other packages may be imported.** Your program may use any methods available in packages automatically imported as part of the Java language and any methods in `javax.swing.JOptionPane`.

**Spoilers:**

We have not yet covered loops and there is no need to use a loop in this program. It is possible to use loops to solve this problem, but my solution does not use loops or anything else not yet covered in the assigned reading.

To check the input for errors, you may find the following methods useful:

From `java.lang.String`:
- `char charAt(int index)`

From `java.lang.Character`:
- `static boolean isDigit(char c)`

To avoid a "java.lang.StringIndexOutOfBoundsException" when calling `charAt`, “protect” the call to `charAt` by placing it within an if statement to first verify that the character at the position you want to check exists. Do this with the instance method of the `java.lang.String` class:
- `int length()`

To separate the two integers (pay and cost) from the input string, use:

From `java.lang.String`:
- `String substring(int beginIndex, int endIndex)`

After the input integers have been separated, use:

From `java.lang.Integer`:
- `static int parseInt(String s)`

You will need to use if-else statements form section 3.1.
Grading Rubric [20 points total]:

[1 point]: Attached one file in Blackboard Learn with the file name: `ChangeMaker_yourName.java`, where `yourName` is whatever sequence of characters you want to be known by in this course.

[14 points]: When the user enters an illegal command, an error dialog is displayed with the error icon and an appropriate error message. Then, the program exits without crashing. Test cases will include (2 points per case):

- clicking "cancel" in the input dialog.
- clicking "ok" with no data entered.
- missing ':'.
- pay that is too large or not correct increment.
- cost that is too large or not correct increment.
- cost that is greater than pay.
- invalid characters.

[5 points]: When input is valid, the specified output dialog is displayed in the specified format. Finally, when the change dialog is closed, the program must exit.

Penalties:

[-5 points]: Code does not adhere to those parts of the hallowed CS-259 coding standard thus far covered:

1) Correct indenting (no tabs and two spaces per block level).
2) Correct placement of brackets.
3) A comment at the top of the class giving your full name and the date.
4) In-line comments as needed. "As needed" is, in practice, quite subjective. In early labs, I will point out missing, excessive or otherwise poor comments without taking off points until you have had time to develop good judgment.
5) Must compile without warnings when using IntelliJ’s default warning settings.

Note: all 5 points are lost if any one of the standards is severely broken.

Note: No more than -5 even will be assigned for this section even of the code is a total mess and breaks all our coding standards.

[-5 points]: Code includes a package other than `javax.swing.JOptionPane`. 

-3 of 3-