Homework #3
Make Change (Better version)
ISDS 372, Prof. Robin Burke
Assigned: 9/13/01
Due: 9/25/01

Objective
Rewrite the change making application so that it has better output, error trapping, takes multiple inputs, and can handle either currency.

Activity
• Create a JBuilder project and create a class called BetterChangeApplication.
• Copy the code below (found on the course website or lab folders) into your application.
• Read the program and make sure you understand what it is doing.
• Missing portions of the program are labeled with “// --> Your code here <--” labels and described in program description language. Insert your code in these places.
• The output of your program should be more readable than in homeworks 1 and 2. Denominations with a zero quantity should not be output. Denomination descriptions should be grammatically correct: “1 quarter” not “1 quarters.”

Submission
Students should

Turn in at the start of class a hardcopy of the code of your two class files with a cover page clearly indicating the number and name of the assignment and the student’s name and ID #.

Before class time, submit a folder containing the complete JBuilder project for the Java classes to the on-line course Drop Box for homework #3. This folder is accessible as a shared volume on lab and classroom machines at \Doctor\Assignments\Burke\ISDS 372\HW3\. You must copy the entire folder at once: files and folders placed on the server cannot be modified. Your folder should be named with your last name, the last four digits of your student it and the assignment number. For example: Burke1234_HW3. If you make a mistake and have to submit the folder again, add a letter to the end. The system will not permit you to overwrite your first submission. I will grade the most recent folder (submitted up to classtime).

Assessment
This assignment will be assessed on the completeness of the solution to the problem. Partial solutions will be given partial credit but only for those features of the application that operate. No credit will be given for non-functional code.

Hints and Notes
1. Use the existing functions getValue and getValue. They perform error checking of the input. Read them to see how they work. (You might want to skim 14.1-14.8 in your book for an overview of the try/catch system in Java.)

2. You will need to define two methods that will compute the change in each currency. They should have signatures as follows:

private static void makeChangeUS ( )
private static void makeChangeUK ( )
3. You will need if statements to clean up the output.

**Extra credit**

A better way to clean up the output would be a method that you can call to format each quantity. The method would have the following signature:

```java
private static String formatQuantity (int quantity, String strSingular, String strPlural)
```

Here’s how it should work:

- **method call:** `formatQuantity (1, "quarter", "quarters")`
  - **result:** "1 quarter"

- **method call:** `formatQuantity(2, "penny", "pennies")`
  - **result:** "2 pennies"

- **method call:** `formatQuantity(0, "shilling", "shillings")`
  - **result:** ""

**Java code**

```java
import javax.swing.JOptionPane;

public class BetterChangeApplication {

  // Constants for different numeric conversions
  public static final int INT = 0;
  public static final int DOUBLE = 1;

  public static void main(String[] args) {
    // --> Your code here <--
    // Loop until sentinel value input
    // Get currency type from user or sentinel value
    // Call appropriate change making method
    // --> Your code here <--
    System.exit(0);
  }

  // --> Your code here <--
  // makeChangeUS method
  // Get floating point value from user
  // Convert to pennies
  // Compute change for each denomination
  // Show answer to user
  // --> Your code here <--

  // --> Your code here <--
  // makeChangeUK method
  // Get integer value from user
  // Compute change for each denomination
  // Show answer to user
  // --> Your code here <--

```
private static int getIntValue(String strPrompt) {
    return (int) getValue(strPrompt, INT);
}

private static double getDoubleValue(String strPrompt) {
    return getValue(strPrompt, DOUBLE);
}

private static double getValue(String strPrompt, int type) {
    double answer = 0.0;
    boolean bNoValidInput = true;

    // Loop until a valid input is received
    while (bNoValidInput) {
        try {
            // Prompt user and get input
            String strAnswer = JOptionPane.showInputDialog(strPrompt);

            // Attempt appropriate numeric conversion
            if (type == DOUBLE) {
                answer = Double.parseDouble(strAnswer);
            } else {
                answer = Integer.parseInt(strAnswer);
            }
            bNoValidInput = false;
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(null,
                "Invalid number format", "Error",
                JOptionPane.ERROR_MESSAGE);
        }
    }
    return answer;
}