

CSE 174 – Program 5 – 29 Points – Math and getting user input – Due Sunday, October 6, by 11:59pm

Part 1: Write a Java program that gets user input (3 or more doubles and/or ints) and computes something *interesting*. You will be graded in part on the interestingness of your choice.

- Give your class an appropriate name, beginning with an uppercase letter. The filename must match the name of your class, and needs to have a .java extension. For example, if you name your class DistanceCalculator, then you must name your file DistanceCalculator.java
- Use a series of System.out.print() or .println() statements at the beginning of your program so that the user of your program knows what the program is about, and knows what to enter, and knows how to enter it. Don't assume any familiarity with what you are computing. You may need to help the user (me) to really "get" what your program is about.
- Output your results in sentence form, and include the original user's input as part of that sentence. Make your output look pretty. Watch spacing, punctuation, and so on.
- Include the typical comments in your source code, as well as the typical formatting. Dr. Java has a shortcut for fixing your indentation. Use it (look in the Edit menu of Dr. Java).
- Upload the source code (.java) file to the course website at <http://my.csi.muohio.edu>

Part 2: Enter the program below into a new file in Dr. Java, and fill in the blank using basic mathematical operators to compute the correct number of boxes needed to ship some teddy bears. Boxes hold up to 30 bears. So, for example, 173 bears would require 6 boxes.

- The source code file has to be named BoxCalculator.java
- Don't write any additional lines of code. Your solution should simply fill in the blank below.
- As much as possible, try to use the simplest arithmetic operators. There is beauty in simplicity. Don't use a bulldozer if tweezers will do the job. A correct solution is better than an incorrect solution. A simple correct solution is better than a more complicated correct solution.
- Test thoroughly! I may test your program with several whole numbers in the range 1 through 10000. Your program should work correctly for all of them. Be warned: some incorrect solutions may work well for a lot of the cases, but may fail in some of the other cases.
- Upload your source code file to the course website at <http://my.csi.muohio.edu>

```
/**
 * A program that computes the number of boxes needed to ship some teddy bears.
 * Fill in the blank with one line of code to get the correct answer.
 * Boxes hold 30 teddy bears.
 */
import java.util.Scanner;

public class BoxCalculator {
    public static void main(String[] args) {

        Scanner keyboard = new Scanner(System.in);
        int bears;
        int boxes;

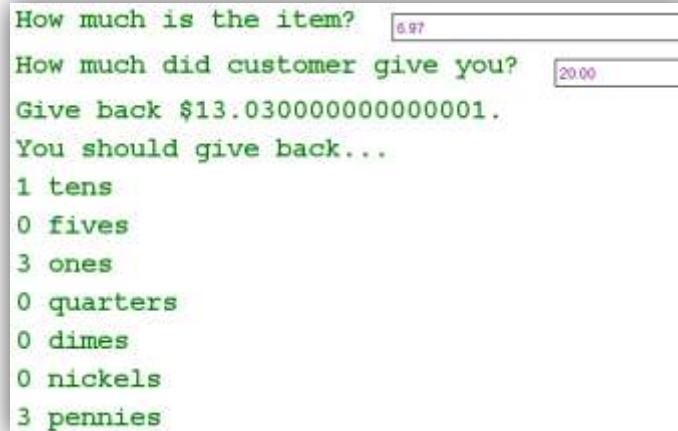
        System.out.print("How many teddy bears do you need to ship? ");
        bears = keyboard.nextInt();

        boxes = _____;

        System.out.println(bears + " bears need " + boxes + " boxes.");
    }
}
```

Part 3: Write a Java program that figures out how to give back change to a customer.

- Give your class an appropriate name, beginning with an uppercase letter. The filename must match the name of your class, and needs to have a .java extension.
- The program should run as shown. Match the wording and the formatting and spacing as closely as possible. Note that the numbers are entered in dollars. That is, if something costs 27 cents, assume that the user will type 0.27.
- The program should compute the number of tens, fives, ones, quarters, nickels, dimes, and pennies to give back.
- Here is a sample run:



```
How much is the item? 6.97
How much did customer give you? 20.00
Give back $13.0300000000000001.
You should give back...
1 tens
0 fives
3 ones
0 quarters
0 dimes
0 nickels
3 pennies
```

- **Advice:**
 - Because of how Java works with doubles, you will sometimes run into rounding issues. Be careful about these issues, *but don't make them your primary focus*. It's OK if the dollar amount is shown with rounding errors (like what is shown above), *but* try to get the change to come out correctly.
 - I personally *recommend* that you convert dollars into cents before trying to do all your calculations. That is, it would probably be helpful to convert the 13.0300000000000001 into 1303 cents, and then start figuring out how to give back change.
 - Absolutely no "if" statements should be used in these calculations. They are unnecessary, and they complicate the problem.
- **A chance to earn one or two extra points:** (this is not a requirement) notice above that there are a lot of lines that begin with zero. It would be nice to only print the lines that have positive numbers. Here, you might want to use "if" statements. That's OK. Don't use "if" statements for your calculations, but you could use them for your printing. You can earn one extra point if you accomplish this. Or, you can earn two extra points if you accomplish it in what I consider to be a "clean" straightforward approach. I won't help you with this. But you can help each other.
- Upload the source code (.java) file to the course website at <http://my.csi.muohio.edu>

Note that you are submitting THREE separate files to the course website. Here's how: Browse and upload, browse and upload, browse and upload. DO NOT ARCHIVE BETWEEN FILES. You should only archive if you wish to RESUBMIT a file. If you do archive a turn-in set, then you will need to REUPLOAD ALL THREE FILES. Any questions? Please ask.

Scoring: The assignment is worth 29 points. The maximum possible score is 31 points.

	Full credit	No credit or Partial credit
Write a program of your own that gets user input (8 points)	You wrote a program that obtains at least 3 numeric values from the user, computes a result, and displays the result.	Your program roughly works, but does not correctly meet one or more of the specifications (getting input, computing results, displaying results)
Program output for your own custom program looks good (4 points)	Your own program adequately communicates with the user (explains background information about the formula, lets the user know what to enter, displays results in a meaningful sentence).	There are parts of your running program that are unclear (background, user prompts, final results), but for the most part, your program is understandable.
Solve the BoxCalculator problem. (5 points)	You found a one-line solution using only arithmetic (no if statements, no ternary operators, no Math methods) that works in all cases.	You found a solution that does not work in some cases, or that ignores specifications.
Solve the ChangeCalculator problem. (5 points)	Your program correctly computes the right change for all test cases.	You found a solution that does not work in some cases, or that ignores specifications.
Format your output in the ChangeCalculator problem as specified. (3 points)	Your output matches the formatting (wording, punctuation, spacing) as given in the example.	Your does no match the expected formatting.
Write source code that is readable and understandable. (4 points)	You wrote code that emphasized readability, including code that follows style guidelines and is consistent with advice from "The Art of Readable Code".	You did not follow all of the guidelines we have been emphasizing in class for code readability.
EXTRA CREDIT: Display only the non-zero change that should be given. (2 points)	You solved this challenge with what your instructor considered a particularly "clean" solution.	You did not solve this challenge, or you solved it using logic that your instructor thinks should be "cleaned up" a bit.