Homework 02 - Tip Calculator & Hawaiian Name

Hello, and welcome! This assignment has two parts, so be sure to complete both.

Part 1: Tip Calculator

You'll be creating a tip calculator to help with all the tricky math one has to do at restaurants. You'll make use of the Scanner class, loops, and if statements.

Name your program `TipCalculator.java`. It should work as follows:

1. Print out instructions for the user.
2. Prompt the user for the prices of the items, one by one. User input will not include the dollar sign (so it will look like `1.99`, not `$1.99`). You do not have to worry about validating user input — you can assume they enter correct dollar / cent amounts.
3. When the user enters -1 for a price, that means they are done entering prices.
4. Prompt the user for the percentage they want to tip. The tip will be entered as a decimal.
5. Print the subtotal (sum of all the item prices), the tip (subtotal times tip percentage), and the total (subtotal + tip). Round to the nearest cent. We recommend checking out this.

Example Output. User input is bolded.

```
Welcome to Tip Calculator!

Enter the prices of your items. Enter -1 to enter tip percentage.
Item 1: $1.99
Item 2: $1.99
Item 3: $2.99
Item 4: $10.99
Item 5: $8.99
Item 6: $5.39
Item 7: $-1

Enter the tip percentage:0.18

Subtotal: $32.34
Tip:      $5.82
Total:    $38.16
```

Part 2: We're Going to Hawaii!

So now that the snow is gone, everyone's probably ready for some warmer weather, and what place has better weather than Hawaii? When going to Hawaii, it's always nice to come up with your own Hawaiian name, so for this homework you will be creating a program that will test out different names to see if they are Hawaiian. This program should be called `HawaiianName.java`.

Start by asking the user to input their possible Hawaiian name. Users will only input a one word first name. You should then test that name to see if it contains only the letters in the Hawaiian alphabet: a, e, i, o, u, k, l, h, m, n. (Don't forget the user may type in capital letters). If the name only contains those letters, print out `Aloha, <insertNameHere>`. For example, if the user were to enter Lilo, your program would print out `Aloha, Lilo`.

Of course, if their name contains any letters not in the Hawaiian alphabet, we can't let them come with us to Hawaii. But that's no fun, now is it? So instead, you'll go through that name character by character and remove each letter that's not Hawaiian! After that, you can print out a welcome like above.

For example, if the user were to enter John, here is what it would print out:

```
Sorry John, you aren't Hawaiian enough to come!
Let's make your name... ohn!
Aloha, ohn
```

After printing out a welcome, your program should ask the user if they would like to try another name (i.e. print out `Would you like to try again? (y/n)`) and if the user enters `y`, your program should run again from the beginning. If they enter `n`, the program should end. The user will always enter a lowercase `y` or `n`.

A complete run of the program could look as follows. User input is bold.

```
Welcome to Hawaii! Let's check if your name is Hawaiian enough.

Enter your name: Lilo
Aloha, Lilo
Would you to try again? (y/n): y

Enter your name: John
Sorry John, you aren't Hawaiian enough to come!
Let's make your name... ohn!
Aloha, ohn
Would you like to try again? (y/n): n
```

Note on capitalization: When a name is Hawaiian and you are removing letters, we don't care if the resulting name is not properly capitalized. Both 'Ohn' and 'ohn' are acceptable! Proper capitalization here would be a little tricky. Feel free to try though!

Rubric

- [50] `TipCalculator.java`
  - [10] Prompts user for new items until the program receives input of -1
  - [10] Prompts for tip percentage
  - [30] Prints subtotal, tip, total correctly
- [50] `HawaiianName.java`
  - [10] Handles Hawaiian name correctly
    - [10] Greets user with Hawaiian name
  - [20] Handles non-Hawaiian name correctly
    - [10] Informs user their name is not Hawaiian
    - [10] Greets user with corrected name (all non-Hawaiian letters dropped)
  - [10] Prompts user for multiple names
  - [10] Stops when user enters `n`, signalling they no longer wish to continue

---

Allowed Imports

To prevent trivialization of the assignment, you may only import `java.util.Scanner`.

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach or break our auto grader. For that reason, do not use any of the following in your final submission:

- var (the reserved keyword)
- System.exit

Collaboration

Collaboration Statement

To ensure that you acknowledge collaboration and give credit where credit is due, **we require that you place a collaboration statement as a comment at the top of at least one java file that you submit**. That collaboration statement should say either:

I worked on the homework assignment alone, using only course materials.

or

In order to help learn course concepts, I worked on the homework with [give the names of the people you worked with], discussed homework topics and issues with [provide names of people], and/or consulted related material that can be found at [cite any other materials not provided as course materials for CS 1331 that assisted your learning].

Recall that comments are special lines in Java that begin with `//`.

Turn-In Procedure

Submission

To submit, upload the files listed below to the corresponding assignment on Gradescope:

- `TipCalculator.java`
- `HawaiianName.java`

Make sure you see the message stating "HW02 submitted successfully". From this point, Gradescope will run a basic autograder on your submission as discussed in the next section.

You can submit as many times as you want before the deadline, so feel free to resubmit as you make substantial progress on the homework. We will only grade your last submission: be sure to **submit every file each time you resubmit**.

Gradescope Autograder

For each submission, you will be able to see the results of a few basic test cases on your code. Each test typically corresponds to a rubric item, and the score returned represents the performance of your code on those rubric items only. If you fail a test, you can look at the output to determine what went wrong and resubmit once you have fixed the issue.

The Gradescope tests serve two main purposes:

1. Prevent upload mistakes (e.g. forgetting checkstyle, non-compiling code)
2. Provide basic formatting and usage validation

In other words, the test cases on Gradescope are by no means comprehensive. Be sure to thoroughly test your code by considering edge cases and writing your own test files. You also should avoid using Gradescope to compile, run, or checkstyle your code; you can do that locally on your machine.

Other portions of your assignment can also be graded by a TA once the submission deadline has passed, so the output on Gradescope may not necessarily reflect your grade for the assignment.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Do not submit `.class` files.
- Test your code in addition to the basic checks on Gradescope
- Submit every file each time you resubmit
- Read the "Allowed Imports" and "Restricted Features" to avoid losing points
- Check on Piazza for all official clarifications