CIS 255 • Java Programming
Laboratory Assignment 4: Loops

In this laboratory assignment you will work with a Java loop structure to allow the user to determine how much a family will have to spend on vacation after saving money for twelve months.

Background

A family is planning a vacation a year ahead of time and wants to have some spending money for the trip. The purpose of your program is to assist the family in determining how much vacation spending money it will have based on setting aside some money every month. The family's savings amount will vary from month to month, so the program must prompt the user for each month's amount.

Step 1: Download the Starter File and Launch TextPad

On Blackboard, click the link for the starter file to download it. Make sure you insert your first initial and last name and the laboratory assignment number where indicated (**FLast_Lab04.java**) and that you specify the file location. Once you have downloaded the file, open it by launching TextPad and using the Open command or by right-clicking the file in Windows and selecting Open With > TextPad.

Step 2: Customize Your Code and Add Import Statements

Edit the comments at the top of the code to include your name, the course number, the laboratory assignment number, and a brief description of the program's contents. You may also wish to include the date (perhaps after the laboratory assignment number). Also, modify the class name to match your file name. As you continue your code, you may go ahead and add comments to the statements or groups of statements that you believe deserve additional explanation, or you may wait until after you have written your entire program to add comments.

You may use the console (`Scanner`) or dialogs (`JOptionPane`) for interaction with the user. Below the comments at the top, add the `import` statement specific to the type of interaction you plan on using. You'll also be formatting numeric output; if using the `printf` method, no additional `import` statement is required, but the `DecimalFormat` class does require another `import` statement.

Step 3: Declare Primitive Variables

Declare the following variables at the beginning of the `main` method:

Description	Data Form	Initial Value
Month counter	Whole number	1
Monthly savings amount	Real number	From input
Total saved for vacation	Real number	0

You do not have to initialize these variables in the declaration statement; you may assign values to them separately. For example, you may initialize the counter variable as part of the loop structure.

Step 4: Declare Objects for Input and Output

If you plan to read input from the console, declare a `Scanner` object variable; if reading input from dialogs, declare a `String` variable to temporarily store the input from each dialog. To format output in dialogs, or to format output in the console without using `printf`, declare a `DecimalFormat` object variable that specifies that the ones digits and two digits after the decimal are required and that commas must be shown in numbers large enough to utilize them.

Step 5: Use a Loop to Obtain Input and Calculate the Total

The program should prompt the user for the amount saved each month with a label (assume the highlighted value comes from a counter storing the current month):

```
Enter the amount you are able to save during month #1:  $
```

Use a **loop** to display this prompt for each of the twelve months, including the month number in each one. After the user enters a value for the current month, add it to the total. This is an example of accumulation, or calculating a running total (section 4.6).

Don't write twelve separate sets of output, input, and assignment statements. Placing a single output, input, and assignment statement in a loop is sufficient to handle all twelve months.

If using dialogs, don't forget to store the value from the input dialog in a temporary `String` variable. Then, use the parse method that corresponds to the desired data type. See Table 2-18.

Step 6: Display the Result

Use a console output statement or a dialog to display the total with an appropriate label. An example follows (assume the highlighted value comes from the variable storing the total):

```
Based on these monthly savings amounts, you will have $592.87 to spend
on your vacation next year.
```

Make sure the program displays the sum with proper formatting. If you declared a `DecimalFormat` object, use the method `format`. However, if you are performing console output, you may use the `printf` method instead (section 3.11).

Step 7: Use a Decision Structure to Display a Message

After the program displays the total amount saved, if the amount is at least $1,000, the program should display a message similar to the following:

```
That's a nice chunk of change!
```

If the amount is less than $100, the program should display a message similar to the following:

```
Maybe you should keep saving another year.
```

If the amount does not meet either of the above criteria, the program should display a message similar to the following:

```
I hope you enjoy your vacation!
```

Step 8: Compile Your Code and Execute the Byte Code

To determine if there are any syntax errors in your source code, and to generate the byte code once your source code is free of syntax errors, select the item **Compile Java** from the **External Tools** submenu of the **Tools** menu. The results of the build process will appear in the **Tool Output** area below your code. If the Tool Output area indicates that there are syntax errors in your code, you may double-click the line that describes each syntax error to see the line that contains the error (or, in some cases, the line after the error).

Once you receive a message in the Tool Output area that the tool completed successfully, you're ready to test your program. In the **External Tools** submenu of the **Tools** menu, select **Run Java Application**. Enter some test values to verify that the results match what you believe they should be.

To test multiple sets of input, you'll need to run the program multiple times. You could nest all of this logic inside another loop to allow multiple customers to use the program, but leave the program as-is for simplicity's sake.

Step 9: Add Inline Comments and Submit Your Code

Make sure that your code has **three or more inline comments** near the statements that you believe warrant the additional description (specifically those that deal with loops). These can be simple line comments at the end of a line of code that needs explanation, or block comments where each precedes a series of statements described in the comment.

You are not required to attach your compiled byte code (the file that ends in `.class`) for any assignment. Only your source code file (the file that ends in `.java`) is required.

Follow the instructions on the course website to attach and submit your source code.

Make sure you complete the submission process! If you attach your file without submitting the assignment, your file will not be received!