CIS 255 • Java Programming
Laboratory Assignment 3: Decisions and Dialogs

In this laboratory assignment you will work with Java decision structures to determine the new balance in a savings account after one month based on the amount of principal and the age of the account holder. The user will enter the account holder's values and see the results via dialogs.

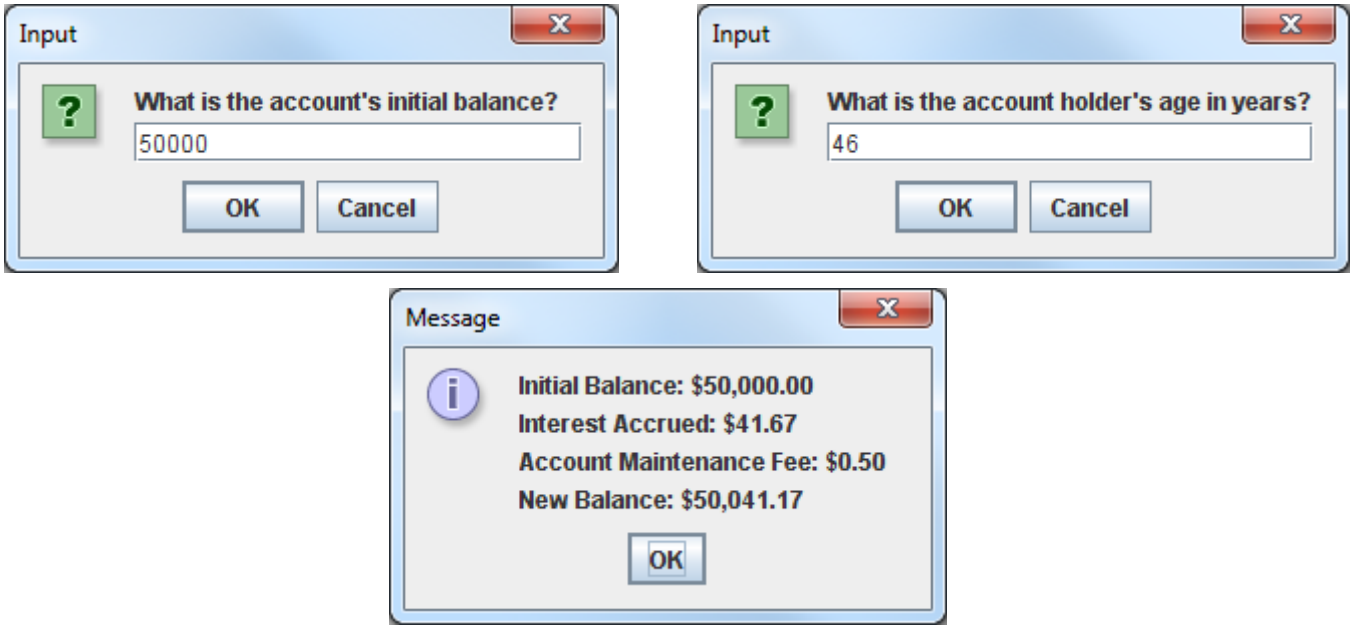
Background

A credit union offers savings accounts with interest rates that vary based on the amount in the account:

Balance	Interest Rate
Below $100.00	No interest
$100.00 to $2,499.99	0.75%
$2,500.00 to $24,999.99	0.85%
$25,000.00 to $99,999.99	1.00%
$100,000 and above	1.25%

Additionally, every account is subject to a $0.50 per month account maintenance fee unless the account holder is under the age of 18 or at least 65 years old (that is, every account holder between the ages of 18 and 64, inclusive, will have this monthly fee deducted from the balance after the interest is applied).

Example Run:



Step 1: Configure Your Computer

If you want to be able to compile and run your Java programs on your own computer, you'll need to install the Java Development Kit (JDK). Additionally, you'll probably want to use a text editor that highlights Java syntax; TextPad is one such program. Refer to the configuration instructions for the full details of what needs to be done.

Step 2: Download the Starter File and Launch TextPad

On the assignment page on Blackboard, click the link for the starter file to download it. Make sure you insert your first initial and last name where indicated (**FLast_Lab03.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.

Pay attention to the comments included in the starter file, as they will show you where to add specific elements to your code. You may remove these comments as you complete each step.

Step 3: Customize Your Code

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. You may also wait until after you have written your entire program to add comments.

Step 4: Add `import` Statements

Since you used the console (`Scanner`) in Laboratory Assignment 2, you will use dialogs (`JOptionPane`) for interaction with the user in this assignment. Below the comments at the top, add the `import` statement for `JOptionPane`. You will also need an `import` statement for `DecimalFormat` to use that class for formatting the output.

For details on the `JOptionPane` class, see section 2.14. For `DecimalFormat`, see section 3.10.

Step 5: Declare Variables

Declare a `DecimalFormat` object. The formatting `String` literal in the parentheses should indicate that the ones digit and two digits after the decimal are required, and that there should be commas if the number is large enough to require them.

Because the input from a dialog box is always returned as a `String`, declare a `String` variable to store this input. Use a generic name; the input from this `String` will be converted to numeric form for each value.

Additionally, declare variables for the following quantities:

Description	Data Form	Initial Value
Initial balance	Real number	From input
Monthly interest accumulated	Real number	Calculated
New balance	Real number	Calculated
Age	Whole number	From input

You can declare all the variables of the same type in a single statement, or you can use one statement for each variable declaration. Remember: variable names cannot contain spaces.

You may also declare named constants for the four interest rates and the monthly account maintenance fee (mentioned in the **Background** section on page 1); each of these values is a real number.

Step 6: Obtain Input in the Proper Form

The first value entered by the user is the account's initial balance. Write a call to the `JOptionPane` method `showInputDialog` with a prompt to the user for the balance. Assign the result of this call to your `String` variable, since the value returned by `showInputDialog` is always a `String`.

The calls to the two dialog box methods must include the class name `JOptionPane`.

The value entered by the user should actually be stored in the numeric variable for initial balance, but it must be converted from the `String` form first. Call the parse method (table 2-18) that will convert the `String` variable's value to the proper type, assigning the result to the initial balance variable.

Be careful of spelling when calling the different parse methods!

Repeat this process for the second input statement to obtain the account holder's age. Display an input dialog with a prompt, and store the user's response in the `String` variable. Then, use the appropriate parse method to convert the `String` variable to the proper type, and assign the result to the variable for the age.

Step 7: Perform the Calculations with Decisions

The program uses the following formulas to calculate the interest accumulated for one month and the new balance:

monthly interest accumulated = initial balance * interest rate / 12
new balance = initial balance + interest

However, the interest rate used depends on the initial balance. Use a decision structures to determine which rate to use; one of the better ways to do this would be to use the `if-else-if` statement (section 3.4), with an `if` or `else if` statement for each of the four balance ranges mentioned in the **Background** section on the first page. Place the interest calculation assignment statement below the decision structure that corresponds to the required balance level. Make sure your variable names match the declarations at the top of your program.

When applying percentages in programming, you use the decimal form of the value. Make sure to use constants in decimal form (e.g., 0.0085 for 0.85%) or divide by 100 in the use of the interest rate.

Add the interest to the initial balance to calculate the new balance. The program then needs to determine whether the user is subject to a fifty cent account maintenance fee. Add another decision structure containing an assignment statement that deducts the fee from the new balance if the account holder's age is

within the range between childhood and senior citizenship as defined in the **Background** section. For details on how to check to see if a value is within a range, see section 3.5 in your textbook (the subsection labeled "Checking Numeric Ranges with Logical Operators").

Don't confuse > and < with >= and <= (the former two exclude the endpoints of the range, whereas the latter two include the endpoints).

Step 8: Display the Results

Use one or more calls to the method `showMessageDialog` to display the results with labels (values in *italics* are variables):

`Initial Balance: $`*initial balance*
`Interest Accrued: $`*interest*
`Account Maintenance Fee: $0.50`
`New Balance: $`*new balance*

The program should only display the account maintenance fee if it applies; otherwise, just display the other three values (use another decision structure to determine whether to include or exclude the fee):

`Initial Balance: $`*initial balance*
`Interest Accrued: $`*interest*
`New Balance: $`*new balance*

Don't forget the argument `null` when calling the method to display a message dialog!

A single dialog can contain multiple lines of text. Use the escape sequence `\n` to indicate where to proceed to the next line (make sure the escape sequence is within quotation marks).

When the program displays the value of each variable, format it properly by placing the variable in the parentheses of a call to the method `format` using your `DecimalFormat` object.

Step 9: Compile Your 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).

Don't try to correct all of your syntax errors at once. Correcting one error listed in the Tool Output area and compiling your code again may resolve or reveal additional errors.

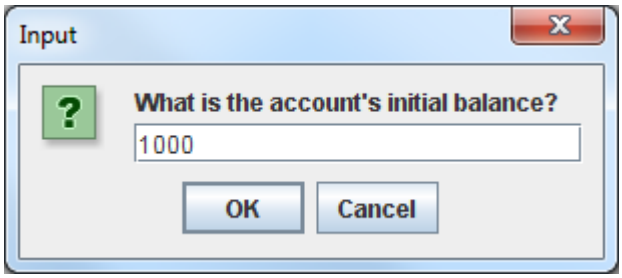
Step 10: Execute the Byte Code

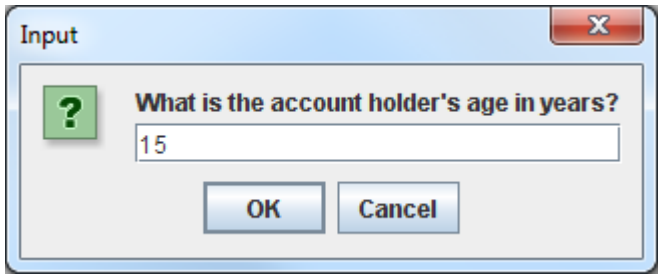
Once you receive a message in the Tool Output area that the tool completed successfully, you're ready to test your program. In the **External Tools** submenu of the **Tools** menu, select **Run Java Application**. Enter some test values to verify that the results match what you believe they should be.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

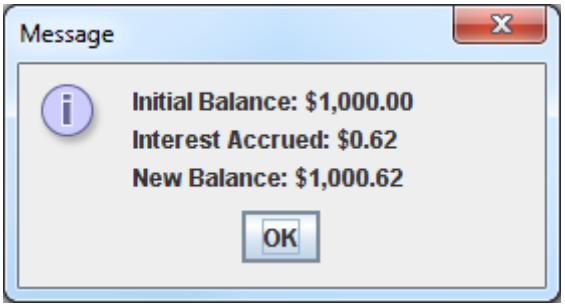
To test multiple sets of input, you'll need to run the program multiple times. In chapter 4, you'll learn how to repeat statements within a single run of a program.

Choose sample input values that will be easy to calculate in your head. If you want to use more complex values, you may need a calculator to verify the math.

Another Example Run:







Step 11: 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 decision structures and dialog boxes). 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!