Lab 4 Mortgage Looping:

You will write a test plan and the Java code for a program that uses a method to calculate the monthly payment for a loan. The formula for calculating a payment is shown further down. This application will produce a list of possible loan payments based on a series of annual interest rates starting with a user entered starting annual interest rate which is incremented in a loop until it reaches a user entered ending annual interest rate. The rate will be incremented by 0.25% (which is the decimal value 0.0025) from the starting annual interest rate to the ending interest rate. Increment the number of years by which the starting term will increment as it progresses towards the ending term by 5.

In summary, the user will need to enter:

- A loan amount such as 100,000 dollars
- A starting annual interest rate such as 4.000%, which is the decimal value 0.0400
- An ending annual interest rate such as 6.000%, which is the decimal value 0.0600
- The first number of years over which the loan will be repaid such as 15
- The last number of years over which the loan will be repaid such as 30

The main method will:

- Call functions to acquire input from the user (while validating the input)
- Input validation:
    - Loan amount: 1000 to 9999999
    - Interest rates: 1.00 to 8.00
    - Number of years: 1 to 45
        - When the number of years is more than 30 years, change the ending year to the starting year plus 30
- Invoke the payment calculation method within a loop.

Hint: this program requires nested loops – an outer loop controlled by controlled by the interest rate as it progresses from the stating rate to the ending and an inner loop rate the term as it progresses from the stating term to the ending term. It will calculate the payment and display them from inside the inner loop. And all this will be inside a do-you-want-to-start-over loop.

Since the user enters annual rates and the number of years for the loan, the program will need to convert the annual values to monthly values:

- Calculate the monthly interest rate, which we'll abbreviate as mir (monthly interest rate), as the annual rate divided by 12.
- Calculate the numbers of months over which the loan will be repaid, which we'll abbreviate as mtp (months to pay), as the number of years times* 12.

Here's how to calculate a loan payment:

Annuity Factor =(mir*(1+mir))^mtp))/(((1+mir)^mtp)-1)

Assuming a 30 year loan for $180,000 and an annual rate of 0.0375 (3.75%) the results of substituting the values in the formula are:

Annuity Factor = (0.003125*(1+0.003125)^360)/(((1+0.003125)^360)-1)

Annuity Factor = 0.004631

Payment = Amount Loaned * Annuity Factor

Payment = 180000 * 0.004631

Payment = 833.61

I used ^ to indicate raising a number to a power because Excel uses the ^ for this purpose. Java, like C, C++ and C#, has no such operator. You must use the pow method in the Math class. This makes writing the annuity factor formula more awkward. You may find it easier to break this calculation into some extra steps, rather than tripping over nested parentheses:

1. Numerator = (mir*(1+mir)^mtp) = (0.003125*(1+0.003125)^360)
2. Denominator =(((1+mir)^mtp)-1)
3. Annuity Factor = Numerator / Denominator = 0.0096088 / 2.0748184

Of course, you will use variables and constants, not hardcoded values like 0.003125 and 360, in your Java program.

Validate the input data for all variables. You can use IR4.java to get user input.

Use the data given in the example below to validate that methods correctly calculate loan payments. You can google "mortgage loan calculator" and use the calculator shown to verify your calculations.

The valid values for the input values are:

Loan amount: 1,000 to 99,999,999

Starting and Ending interest rates: .250 to 7.000

Round down to the closest .25 value

First and Last term in years: 1 to 30

Add 5 years for the report. Only show max of 30 years (6 columns of years)

Each value must make sense, for example the ending interest rate and ending year must be => the starting values.

The input questions must be EXACTLY as shown below and in the same order.
If not, -50% off your grade.

Program Output:

Enter the loan amount : **100000**

Enter the starting annual interest rate as a percent (n.nnn) : **4.000**

Enter the ending annual interest rate as a percent (n.nnn) : **6.000**

Enter the first term in years for calculating payments : **15**

Enter the last term in years for calculating payments **: 30**

Interest Rate	15 Years	20 Years	25 Years	30 Years
4.000	739.69	605.98	527.84	477.42
4.250	752.28	619.23	541.74	491.94
4.500	764.99	632.65	555.83	506.69
4.750	777.83	646.22	570.12	521.65
5.000	790.79	659.96	584.59	536.82
5.250	803.88	673.84	599.25	552.20
5.500	817.08	687.89	614.09	567.79
5.750	830.41	702.08	629.11	583.57
6.000	843.86	716.43	644.30	599.55

Do you want to start over? **No**

------ Program Complete -------

Use **printf** to align the numbers in even columns as shown above. **This may take some tinkering and is a large part of this lab**. Assume this alone will take you an hour or more.

The column headings “15 Years 20 Years 25 Years 30 Years” are somewhat challenging to do. They require having a separate loop before the nested loops. If you can’t manage to display the headings, your project will be reduced 10%, assuming everything else is correct. Restrict the input to prevent the output lines from wrapping around the console.