JSP Assignment

Loan Amortization Calculator with JSP as View/Controller

Objective:

- Build a working application with JSP and Java Components

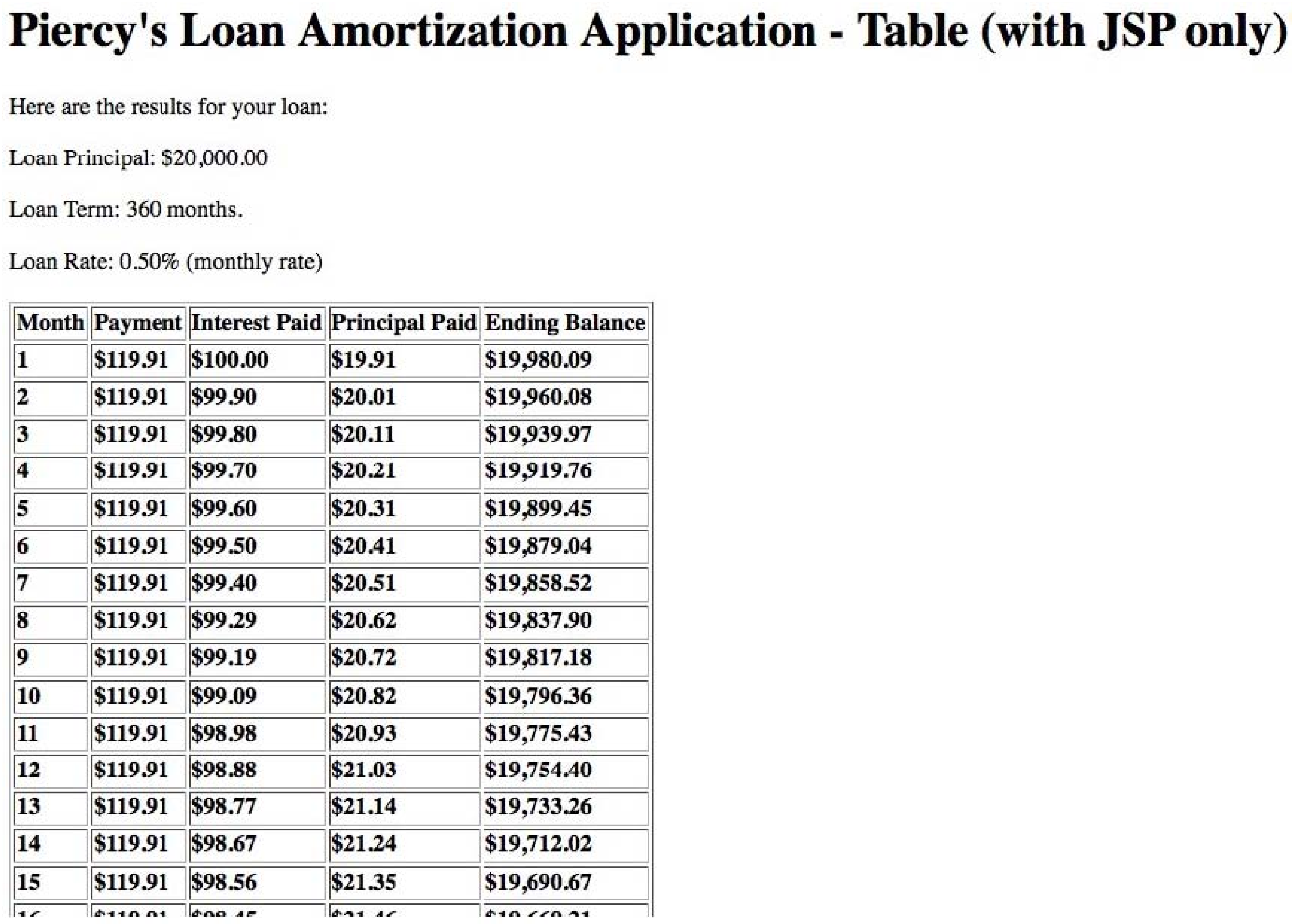
Create a folder to store all of your files generated with this assignment. Name this folder based on your last name concatenated with the letters JSP. For example, if your name is Severian, you'd call your folder SeverianJSP.zip this folder and post it to ELC by the due date.

What to do:

Write a Java Web application using a JSP and Java classes that will calculate an amortization table for a loan.

An amortization table shows the monthly payment, the amount paid to interest, the amount paid down on the balance, and the monthly final balance over the life of a loan. When completed your application should produce output with numbers like that shown in Figures 1 and 2. As you can see in Figure 1, these values are for a 30 year loan with a principle of $20,000 at 6% interest. Your application should work correctly for any reasonable values of loan properties.

Your version of this output should be sent to the browser as the response from a jsp file. In addition, your program should work for all possible input values.



Piercy's Loan Amortization Application - Table (with JSP only)

Here are the results for your loan:

Loan Principal: $20,000.00

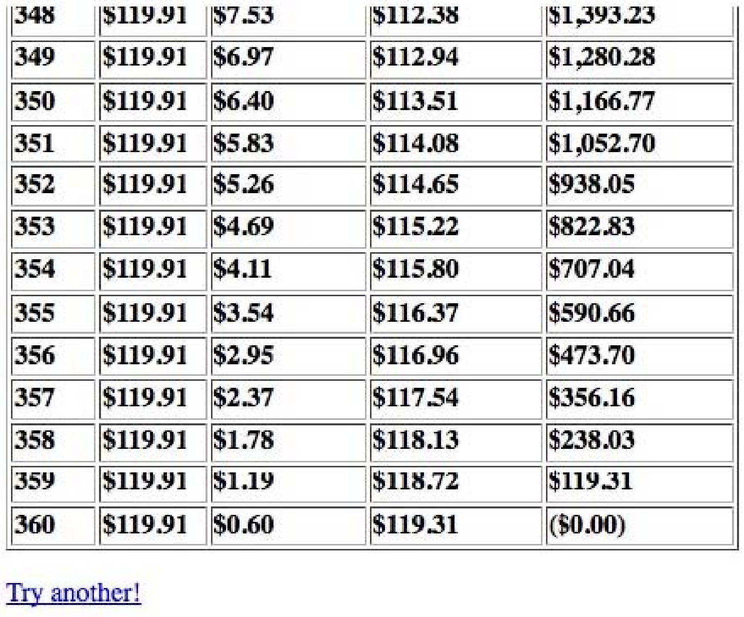
Loan Term: 360 months.

Loan Rate: 0.50% (monthly rate)

Month	Payment	Interest Paid	Principal Paid	Ending Balance
1	$119.91	$100.00	$19.91	$19,980.09
2	$119.91	$99.90	$20.01	$19,960.08
3	$119.91	$99.80	$20.11	$19,939.97
4	$119.91	$99.70	$20.21	$19,919.76
5	$119.91	$99.60	$20.31	$19,899.45
6	$119.91	$99.50	$20.41	$19,879.04
7	$119.91	$99.40	$20.51	$19,858.52
8	$119.91	$99.29	$20.62	$19,837.90
9	$119.91	$99.19	$20.72	$19,817.18
10	$119.91	$99.09	$20.82	$19,796.36
11	$119.91	$98.98	$20.93	$19,775.43
12	$119.91	$98.88	$21.03	$19,754.40
13	$119.91	$98.77	$21.14	$19,733.26
14	$119.91	$98.67	$21.24	$19,712.02
15	$119.91	$98.56	$21.35	$19,690.67
16	$119.91	$98.45	$21.46	$19,669.21

Figure 1: Beginnning of Loan Amortization Table

Figure 2 shows the last rows of values in the amortization table. notice that for correct output the table will end with a zero balance during the last term. Of course, here we are assuming a constant payment every month with no early payments allowed.



348	$119.91	$7.53	$112.38	$1,393.23
349	$119.91	$6.97	$112.94	$1,280.28
350	$119.91	$6.40	$113.51	$1,166.77
351	$119.91	$5.83	$114.08	$1,052.70
352	$119.91	$5.26	$114.65	$938.05
353	$119.91	$4.69	$115.22	$822.83
354	$119.91	$4.11	$115.80	$707.04
355	$119.91	$3.54	$116.37	$590.66
356	$119.91	$2.95	$116.96	$473.70
357	$119.91	$2.37	$117.54	$356.16
358	$119.91	$1.78	$118.13	$238.03
359	$119.91	$1.19	$118.72	$119.31
360	$119.91	$0.60	$119.31	($0.00)

Try another!

Figure 2: End of Loan Amortization Table

Required Components:

Loan.java: This defines a loan object. The Loan object should have principal, monthly rate, and total term as fields. It should have at least two constructors, a getter/setter pair for each instance variable, and a method that allows one to get a monthly payment. Here the Total Term is the total number of months in the life of the loan and the Monthly Rate is in decimal format (for example 1% = 0.01). You can calculate a monthly payment for a loan using:

$$\text{Monthly Payment} = \text{Principal} * \text{Monthly Rate} / (1 - (1 + \text{Monthly Rate})^{-\text{Total Term}}))$$

Amortization.java: This component will provide the amortization table portion of the output shown in Figures 1 and 2 when the doAmortization method is called. The Amortization class will include a Loan object as an instance variable. It will also have at least two constructors, a getter/setter pair for the instance variable, and a doAmortization method that will return a String containing the html based amortization table.

Some other relationships that you will need to complete the view are:

monthly payment to interest = monthly interest rate * current balance

monthly payment to balance = monthly payment - monthly payment to interest

current balance (after payment) = current balance (before payment) - monthly payment to balance

index.jsp: an html form page that allows the user to enter the initial loan information. Use textboxes and buttons. When the form is submitted, the doAmortization.jsp will be called to create the loan amortization output. doAmortization.jsp: a jsp page that will:

1. Get the data from the request.
2. Create a Loan and an Amortization objects
3. Use the Loan and Amortization object to calculate the amortization table
4. Get values from Amortization object and incorporate it into the HTML for response to the client.

What to Turn in:

The Eclipse project folder stored inside the assigned folder (see above).
