**CMSC 203**

**Fall 2015**

**Assignment #1**

Concepts tested by this program:

Set up and run java, javac, and an IDE (Eclipse) on your computer

Create a class

**main** method

variables

strings

mathematical operators

assignment operator

input/output

Eclipse IDE

## Specifications

Develop a **Loan Calculator** (as a Java **standalone application**) to calculate and display the monthly payment for a loan.  The user will be asked to enter the **loan amount**, the **interest rate**, and the **number of payments**.  The formula used for such a calculation is:

For example, for a loan amount of $225,000 with the APR 10% and 360 payments, the monthly payment is:

Payment, loan amount, APR, 12.0, and 1.0 are real numbers; whereas N=number of payments is an integer. If you wish to use other formulas, please feel free to use them.

Java Hints:

System.out.print("2 to the power of 3 is: ");

System.out.println(Math.pow(2.0, 3.0));   // Displays: 2 to the power of 3 is: 8.0

***Application Requirements***

* Ask the user for the loan amount, interest rate and number of payments.
* Remind the user the numeric values must be greater than 0.
* Display a header for your application.
* Use the Scanner class for input, and use System.out.print or System.out.println for output.

***Assignment Requirements***

* Write pseudo-code in English showing the steps to solve the problem (detailed enough for someone else to write the code based on this algorithm, but not including “setup” like import statements or variable declarions).
* Prepare a test table with expected results and actual results, in the following format:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Number | Input Variable Values | | | Expected Results (manually calculated) | Actual Results (from running the program) |
| Loan Amount | APR | Number of Payments |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |

Use as many tests as you need to check normal cases (like the above example) and boundary conditions (like 1 payment, or 0% APR).

* Run the application twice, and provide screen shots of both runs.
* Run it once using command-line Java (javac to compile, and java to run). Your screen shot should show both compilation and execution. If your computer cannot find the javac or java commands, see the lecture slides explaining how to set the path for Java.
* Run it again using an Integrated Development Environment (IDE). The one used in class will be Eclipse. Provide a screen shot of the console output. See the lecture slides explaining how to download and install Eclipse.
* Turn in a Word or PDF document with pseudo-code and test results, screen shots of runs, and the java code you wrote (NOT the .class file) according to the following format.

You will be uploading two .zip files for each assignment. The first .zip file will contain all the files required for the assignment (design document, test results, screen shot of runs, and source code and will be named: LastNameFirstName\_AssignmentX\_Complete. The second .zip file will only contain the .java files and will be named: LastNameFirstName\_AssignmentX\_Moss. This .zip will not have any folders in it – only .java files.

Here’s an example for Assignment 1:

AlexanderRobert\_Assignment1\_Complete.zip [a compressed file containing the following]

PseudoCode\_and\_TestTable.docx

CmdLine\_output.jpg

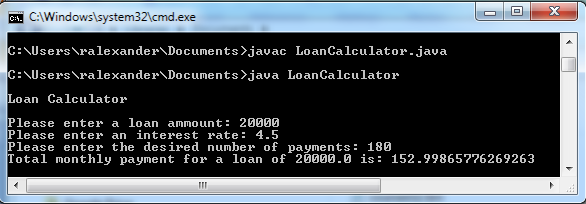
Eclipse\_console\_output.jpg

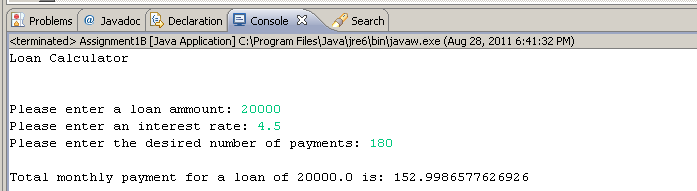
File1.java

AlexanderRobert\_Assignment1\_Moss.zip [a compressed file containing only the following]

File1.java

Example screen shots:





NAME\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Assignment 1 Grading Sheet

**EXTERNAL DOCUMENTATION** 24 (total)

CheckList for Assignment 1 is included and completed 1 pt \_\_\_\_\_

Algorithm Design

List ordered steps in English in to solve the problem 8 pts \_\_\_\_\_

(detailed enough for someone else to write the code based on this algorithm)

Test Cases

Proper Format (Inputs, Expected Output, Runs Correctly) 2 pts \_\_\_\_\_

4 test cases with valid data (unique input data for each case) 6 pts \_\_\_\_\_

Assumptions (if any) 2 pts \_\_\_\_\_

Lessons Learned 5 pt \_\_\_\_\_

In 3+ paragraphs, highlight your lessons learned and learning experience from working on this project. How did you do? What have you learned? What did you struggle with? How will you approach your next project differently?

**PROGRAMMING** 76 (total)

Programming Style

Consistent indentation is used 4 pts \_\_\_\_\_

Appropriate identifiers are used 4 pts \_\_\_\_\_

Internal class documentation (within source code)

Description of what class does 6 pts \_\_\_\_\_

Author’s Name 2 pts \_\_\_\_\_

Appropriate comments within each section of the code 8 pts \_\_\_\_\_

Java file compiles and runs 14 pts \_\_\_\_\_

Accurate output from additional test cases 20 pts \_\_\_\_\_

Program user interface

Clear to user how data is to be entered 6 pts \_\_\_\_\_

Output is easy to understand 6 pts \_\_\_\_\_

Program Details

Remind the user the APR must be greater than 0 4 pts \_\_\_\_\_

Prints application header 2 pts \_\_\_\_\_

TOTAL POINTS \_\_\_\_\_ / 100

Comments: