Problem Set 1
Due: 11AM, Friday September 17, 2004

Loan Calculator / Movie & Game Rental Store (0) [100 points]

Introduction

You just got your bachelor's degree from MIT and are very interested in pursuing a graduate degree as a next step. Unfortunately, you realized how expensive graduate program is and can’t really afford it right now. So what do you do? You have decided to go out make some money and come back to school. After spending a few days thinking about the right business venture, you have decided to open a Movie & Game Rental Store.

From Problem Set 2 through Problem Set 5, you will develop a Java program that manages customers, movies, games, and all the transactions. Through each problem set, your program will be evolved to have better functionality using new and advanced concepts introduced each week. Your final goal, which will be the solution for Problem Set 5, is to develop a GUI (Graphical User Interface) program that is similar to commercial applications you can find at any video rental store.

So what is the first order of business? You need to get a loan to kick start your business. After surfing the Web for hours and not finding a decent loan calculator, you have decided to develop your own loan calculator.

Structure

First, your loan calculator should ask user for his/her maximum monthly budget. Then, it asks for loan related inputs, such as loan amount ($), annual interest rate (%), and total number of months to repay the loan. Based on these inputs, loan calculator finds user’s monthly payment as well as total payment and total interest over the life of loan. Finally, it analyzes output, by comparing monthly payment to user's maximum budget, and tells whether he/she can afford to take the loan.

Once the analysis is done, user should be able to retry with different inputs but with the same maximum budget. Also, for users who want to change the value for maximum monthly budget, your program should be able to start over. For better understanding of the algorithm, see the flow chart below.
User input

In this problem set, your program will use the class JOptionPane, which is in the javax.swing package that comes with the standard Java distribution, and its method showInputDialog() to take inputs from users. This class and method will be covered in more detail later, but for now, follow these steps to implement the functionality required for this problem set.

Step 1. Import JOptionPane class
- You need to add the following line to the top of your Java source code file.
  - import javax.swing.JOptionPane;

Step 2. Define a String variable
- Inside the main() method of your class, define a String variable first.
  - String input;

Step 3. Take user input
- By using the following line of code, your program generates a pop-up window for input.
- Note that value returned by the showInputDialog() method of JOptionPane is represented in Java as a String objects even if the characters typed by the user are digits.
  - input = JOptionPane.showInputDialog("Enter your age");
Step 4. Convert a String object into a number
- Using the code below, your program converts a String object into a number.
- For instance, if user inputs “22” in Step 3, what you get is a String object, “22”. So with the following code, you convert a String “22” into a number 22.
- int myNumber1 = Integer.parseInt(input); OR double myNumber1 = Double.parseDouble(input);

Step 5. Repeat Step 3 and 4
- For additional inputs, repeat Step 3 and 4. You need one variable for each input.

Step 6. Terminate the program
- Every time your program uses JOptionPane in a non-GUI application, it needs to be terminated properly by adding the following line at the end of main() method.
- System.exit(0);

Example. Here is an example of taking two inputs from user

```java
import javax.swing.JOptionPane;   // Step 1
public class Example
{
    public static void main(String[] args)
    {
        String input;   // Step 2
        input = JOptionPane.showInputDialog("Your age"); // Step 3
        int age = Integer.parseInt(input); // Step 4
        input = JOptionPane.showInputDialog("Your ID"); // Step 3
        double id = Double.parseDouble(input); // Step 4
        System.exit(0); // Step 6
    }
}
```

Loan Calculator

Loan calculator takes three inputs: loan amount (C), annual interest rate % (R), and total number of months to repay the loan (N). Interest rate, R%, is always a yearly figure. However, in most loan situations, it is compounded monthly. In this calculator, the monthly payment (P) is calculated by the following formula where \( r = \frac{R}{1200} \):

\[
P = \frac{C \times r \times (1+r)^N}{(1+r)^N-1}
\]
Once you compute the monthly payment, you should be able to find the total payment and total interest.

Assignment

Develop a Java program for loan calculator.

1. Create a public Java class. Name it ProblemSet1.java. This class should contain a main() method, in which all of your code should be.
2. Ask for maximum monthly budget using a pop-up dialog box created by the JOptionPane class.
3. Convert the String returned by the showInputDialog() method of JOptionPane class to a numerical data type such as int or double.
4. Take all the other inputs using the JOptionPane class, converting each to a numerical data type.
5. Find monthly payment, total payment, and total interest.
6. Analyze monthly payment.
7. Print all the inputs and outputs (see sample output below).
8. Ask user to try again: If yes, go to 4. If no, go to 9.
9. Ask user to start over: If yes, go to 2. If no, go to 10.
10. Terminate the program

Note from the flow chart on page 2 that your program should have two loops for 'retry' and 'start over'.

Sample Output

Input:
Maximum budget: $400
Loan amount: $20000
Interest rate (%): 7.5
Number of months: 60

Output:
<Budget>
Maximum Budget: $400.0

<Input>
Loan Amount: $20000.0
Interest Rate: 7.5%
Number of Months: 60

<Summary>
Monthly Payment: $400.75897191247054
Total Payment: $24045.53831474823
Total Interest: $4045.5383147482316

<Budget Analysis>
Monthly Payment Over Maximum Budget: Try Again
Turn In

- Turn in **electronic** copies of **all source code** (.java files). No printed copies are required.

- Place a **comment** with your full name, Stellar username, tutorial section, TA’s name, and assignment number at the beginning of all .java files in your solution.

- Remember to **comment your code**. Points will be taken off for insufficient comments.

- Place all of the files in your solution into a **single zip file**. Submit this single zip file on the 1.00 Web site under the appropriate section and problem set number. For directions, see **HowTo: Submit Problem Set** on the 1.00 Web site.

- Your solution is **due at 11AM**. Your uploaded files should have a time stamp of no later than 11AM on the due date.

- **Do not** turn in compiled byte code (.class files) or backup source code (.java~ files).

Penalties

- **30% off** If you turn in your problem set after 11AM on Friday but **before 11AM on the following Monday**.

- **No Credit** If you turn in your problem set **after 11AM on the following Monday**.