Homework #5
Amortization
ISDS 372, Prof. Robin Burke
Assigned: 10/16/01
Due: 11/1/01

Objective
Write a Java application that computes and displays an amortization table for a fixed interest, fixed period loan. Use flowcharting to develop your design.

Activity
- Create flowcharts that describe how the program will operate. As we have done in class, you should have several levels of flowcharts from high-level ones that show the operation of the whole system to low-level ones that show the operation of an individual method. The flowcharts must be turned in with the assignment.

- Your application will take three arguments (principal, annual interest rate, and # of months) from the command line. It should compute the amortization table as a two-dimensional array: one row for each month, and each row containing four double values: the interest part of the payment, the principal part of the payment, the total payment and the principal remaining after payment.

- Your application should round to the closest cent. Use the following method to do so:
  ```java
  private static double roundToCents (double value)
  {
      return Math.floor ((value*100)+0.5)/100;
  }
  ```

- Currency and other number formatting should be performed using the methods from the Formatter class, included below and found on-line in the course folder and the course web site.

Submission
Students should

Turn in at the start of class a hardcopy of the code of your class file with a cover page clearly indicating the number and name of the assignment and the student’s name and ID #. Also, turn in finished versions of your flowcharts for the whole application. They can be hand-drawn or created in a graphics program such as Visio.

Before class time, submit a folder containing the complete JBuilder project for the Java classes to the on-line course Drop Box for homework #5. This folder is accessible as a shared volume on lab and classroom machines at \Doctor\Assignments\Burke\ISDS 372\HW5\. You must copy the entire folder at once: files and folders placed on the server cannot be modified. Your folder should be named with your last name, the last four digits of your student it and the assignment number. For example: Burke1234_HW5. If you make a mistake and have to submit the folder again, add a letter to the end. The system will not permit you to overwrite your first submission. I will grade the most recent folder (submitted up to classtime). You do not need to submit electronic version of the flowcharts.

Assessment
This assignment will be assessed on the completeness of the solution to the problem and the quality of the flowcharts. I will be looking for logical decomposition of the problem into methods. Partial solutions will be given partial credit.
Hints and Notes

1. You may recall from your finance courses how to calculate the payment for an amortized loan. The payment is fixed, but the proportion of principal increases over the life of the loan. The formula for the payment is

$$ PMT = \frac{PV}{\sum_{i=1}^{n} \frac{1}{(1+i)^i}} $$

$PV$ = the principal, $n$ = the number of months, $i$ = periodic interest rate (annual interest rate / 12).

2. For each line in the table, interest paid = $i \times$ principal, principal pd = payment – interest paid, balance = previous principal – principal pd. The last payment will typically be less than the full payment amount due to rounding.

3. Sample output for command line arguments set to 1000 0.065 12.

<table>
<thead>
<tr>
<th>Interest pd.</th>
<th>Principal pd.</th>
<th>Payment</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>$5.42$</td>
<td>$80.88$</td>
<td>$86.30$</td>
<td>$919.12$</td>
</tr>
<tr>
<td>$4.98$</td>
<td>$81.32$</td>
<td>$86.30$</td>
<td>$837.80$</td>
</tr>
<tr>
<td>$4.54$</td>
<td>$81.76$</td>
<td>$86.30$</td>
<td>$756.04$</td>
</tr>
<tr>
<td>$4.10$</td>
<td>$82.20$</td>
<td>$86.30$</td>
<td>$673.84$</td>
</tr>
<tr>
<td>$3.65$</td>
<td>$82.65$</td>
<td>$86.30$</td>
<td>$591.19$</td>
</tr>
<tr>
<td>$3.20$</td>
<td>$83.10$</td>
<td>$86.30$</td>
<td>$508.09$</td>
</tr>
<tr>
<td>$2.75$</td>
<td>$83.55$</td>
<td>$86.30$</td>
<td>$424.54$</td>
</tr>
<tr>
<td>$2.30$</td>
<td>$84.00$</td>
<td>$86.30$</td>
<td>$340.54$</td>
</tr>
<tr>
<td>$1.84$</td>
<td>$84.46$</td>
<td>$86.30$</td>
<td>$256.08$</td>
</tr>
<tr>
<td>$1.39$</td>
<td>$84.91$</td>
<td>$86.30$</td>
<td>$171.17$</td>
</tr>
<tr>
<td>$0.93$</td>
<td>$85.37$</td>
<td>$86.30$</td>
<td>$85.80$</td>
</tr>
<tr>
<td>$0.46$</td>
<td>$85.80$</td>
<td>$86.26$</td>
<td>$0.00$</td>
</tr>
</tbody>
</table>

4. Use the following class to format numbers:

```java
import java.text.NumberFormat;

public class Formatter {
    static final NumberFormat sfmtPercent = NumberFormat.getPercentInstance();
    static final NumberFormat sfmtCurrency = NumberFormat.getCurrencyInstance();

    public static String formatPercent (double value) {
        sfmtPercent.setMinimumFractionDigits(1);
        return sfmtPercent.format(value);
    }

    public static String formatCurrency (double value) {
        return sfmtCurrency.format(value);
    }
}
```