Simple Financial Planning Program

Consider a simple program that computes loan and retirement account amortizations. Design and implement the simple program, following generalization, abstraction, encapsulation, and other OO principles. Look for opportunities to apply patterns, like the adaptor and temple method pattern. Then, test all components using Path and Input Validation testing techniques.

# Loan Accounts

For loan accounts, the system should

* Gather information the following information
  + Principal amount (loan amount): *P*
  + Total number of payments or periods: *n*
  + Interest rate per period: *r*
  + Loan process fee: f
* Validate the input parameters
  + P >= $1000
  + n >= 2
  + r >= 0 .0008
  + f >= 0
* Compute payment amount per period: A
* Display payment amount and fee amounts
* Compute an amortization
* Display an amortization schedule, total interest and total fees, something like

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Period | Starting Principle Balance | Payment | Amount paid against Principle | Interest Paid | Fee | New Principle Balance |
| 1 | $2535 | $155 | $130 | $20 | $5 | $2405 |
| 2 | $2405 | $155 | $155 | $19 | $5 | $2274 |
| 3 | $2274 | $155 | $132 | $18 | $5 | $2142 |
| … |  |  |  |  |  |  |

Totals $2535 $520 $300

# Retirement Accounts

For a retirement account, assume that a fixed amount will be fixed deposited on a fixed schedule (like monthly) and that it has fixed interest rate. The system should do the following:

* Gather information the following information
  + Starting Balance : *B*
  + Total number of deposits or periods: *n*
  + Periodic Deposit amount : *A*
  + Interest rate per period: *r*
* Validate the input parameters
  + B >= 0
  + n >= 1
  + A >= $100
  + r >= 0
* Compute an amortization
* Display amortization schedule and an ending balance, in table similar to the loan account, but with column headers that are meaningful to the retirement account

# Design Hints

Think about the following issues:

* What is the same between the algorithms between the types of accounts? What is different?
* Can the requirements to adjusted or restated to increase the similarities, without losing anything important or creating inconsistencies relative to the original requirement?
* Are there any existing data structures we can reuse?
* How can we adapt those data structures to easier to use for this problem?
* What should the user interest look like?
* How will the user interest appear different for the two types of accounts?