

CS108 Assignment 2: Method Implementation
Due Date: 17 February 2005 @ Midnight

Learning Objectives

- Develop a class with convenience methods to perform logical operations and financial calculations
- Implement an interactive text-based user-interface for an application
- Implement formatted output
- Implement an application that uses multiple (2) classes
- Use a constructor to initialize member variables

Problem Statement

It's never too early to start planning for your retirement – especially if you want to retire early. In investments, it is most important to have time on your side. Let's define retirement, for the purposes of this exercise: to retire means that you do not need to work to support your basic needs.

In this assignment, you will create a retirement wealth calculator, which will help determine (a) how much money you will need to create a retirement annuity (monthly income for life), and (b) how much money you need to save each year during your working career to be able to reach that goal.

Example

Let's suppose you are 21 years old right now, you want to retire at age 50, and you expect to live until age 120. This means you will have a working career of about 29 years, and a retired life of 70 years. To retire, you figure you will need \$2500 a month to cover your living expenses. Suppose further that you can earn 5% (net of inflation) on stock-market investments over your lifetime investment horizon.

Mathematically, there are 2 investment calculations at play here:

- (1) The present value (e.g. in today's dollars) of a fixed annuity beginning in the future, for the years between retirement (e.g. 50) and death age (e.g. 120).
- (2) The equivalent-annual-cash-flows (e.g. annual investments) you will need to make between today and retirement age to have enough money saved up for the rest of your life.

The formula for the present value of a delayed annuity (PV) is given by:

$PV = \frac{c}{r} \left[1 - \frac{1}{(1+r)^N} \right] \frac{1}{(1+r)^T}$	<i>PV</i> is the present value of the annuity <i>c</i> is the annual payment you will receive in the future <i>r</i> is the rate you earn on the investment <i>N</i> is the number of years you will receive the annuity payments <i>T</i> is the number of years until you start receiving annuity payments
---	--

The formula for equivalent-annual-cash-flows (EACF) is given by:

$EACF = \frac{PV(r)}{\left[1 - \frac{1}{(1+r)^N} \right]}$	<i>EACF</i> is the annual investment you will make until you retire <i>PV</i> is the present value required to fund your retirement annuity <i>r</i> is the rate you earn on the investment <i>N</i> is the number of years you will make investments prior to retiring
---	--

(continued)

Requirements

Your application will take the form of 2 classes:

- The *FinancialCalculator* class, which will implement the financial calculations.
 - o This class will have member variables for all of the variables required for the financial calculations. All member variables will be initialized via the constructor.
 - o Accessor (get) and mutator (set) methods to get and set the several variables required in the calculations.
 - o A *clear* method, which will reset all numeric values to 0.
 - o Two convenience methods: *calculatePV* and *calculateEACF*, which will implement the PV and EACF calculations, respectively. Before performing any calculations, check to make sure the inputs are not zero or negative.

- The *RetirementPlanner* class, which will implement the interactive user-interface.
 - o This class will have a single member variable named *calc*, which is a reference to an instance of the *FinancialCalculator* class, and which will be initialized via the constructor.
 - o The *promptUserForInput* method will ask the user a series of questions, collect each of the inputs, set the inputs on the *FinancialCalculator* object, and call the *calculate* methods.
 - o The *displayResults* method will call the *getPresentValue()* and *getEACF()* methods of the *FinancialCalculator* object, test the return values to see if they are non-zero, and print out a formatted display of the results.
 - o The *main* method will create the reference to the *RetirementPlanner* object, and call the *promptUserForInput* and *displayResults*.

Deliverables

- The java code file(s) (*.java).
- A short executive summary, which will serve to introduce the application.

Grading Criteria

- | | |
|---|-----|
| - Correct syntax, no compile errors, good formatting, follow naming conventions, appropriate comments in the code | 20% |
| - Accessor, mutator, and constructor methods work correctly | 10% |
| - Implementation of financial equations/calculations* | 10% |
| - User interface methods to collect input | 20% |
| - Prints correct output to the screen | 20% |
| - Executive summary document | 20% |

* If you get totally stuck on these equations, I'll give them to you.

Questions?

Bring them to class, or email me, azs@bu.edu.