It’s never too early to plan for retirement, or at the very least, understand *how* to plan for retirement. Your assignment is to determine how much you’ll need to save each month to have a steady stream of predictable income after you’re done working.

We’re not stuffing our mattresses with cash to save for retirement- we’re investing our money. Investing implies that there’s a compounding factor when determining the value of the money tomorrow that is saved today. Money is compounded over time- the longer it is invested, the greater the compounding.

Part of your retirement income was planned during FDRs administration. Social Security (aka FICA or SSI) is a payroll tax split between employer and employee. Social Security payment schedules (how much is paid back) is based on the level of tax paid and age of retirement. For the year 2014, payroll taxes are levied on the first $117,000 of income. If you earned the maximum income, and retired at age 65, you’ll receive $2,642 in monthly payments for the rest of your life. Even if you’re an optimist, DO NOT plan on receiving more than the maximum amount from Uncle Sam.

Look at the following table:

|  |  |
| --- | --- |
| **Save Each Month** | $ 554.13 |
| **Years To Work** | 40 |
| **Annual Return** | 7% |
|  |  |
| **What you need saved** | $ (1,454,485.55) |
|  |  |
| **Years Retired** | 20 |
| **Annual Return** | 2% |
| **Required Income** | $ 10,000.00 |
| **Monthly SSI** | $ 2,642.00 |

The values in yellow are variables. The values in green are calculations. According to my calculations, if I save $554.13 for the next 40 years at 7% interest, I’ll be able to draw the difference between $10,000 and $2642 for the 20 years at 2% interest.

**Assignment:**

Create an original Java program (your own work) that will prompt the user for the values in yellow. Ask how many years they plan to work, expected average return on investment, how long they want to draw and their required income and expected SSI income. The program should respond with the first value- the amount the user will have to save each month.

**Hints:**

There are two key accounting formulas you should to calculate the green values. Don’t worry, it’s not difficult.

If this program is done right, it’s a couple dozen lines of code, including comments and prompts. If you’re writing thousands of lines at 2am… stop, then come see me.

**Rules:**

Interest Rates. Use a range of 0-20% for annual return when in investment mode. Use a range of 0-3% for annual return when in payback mode. This is real life- risk is usually not taken while in payback mode.

Make it simple. In real life, we’d be balancing portfolios based on time decay (how much time is left, moving to safer investments, etc.). Don’t worry about that… Assume an average rate of return.

Deliverables:

* Simple Java Project built on JDK version 1.8.
* One package, one class that will use Scanner object to read input from the user, calculate, and display the result(s).

Grading – general guidelines/rubric

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Exemplary** | **Developing** | **Oh, come on!** |
| **Timeliness**  (25%) | Completed on time.  25 points | More than 1 day, less than one week late.  10 points. | More than 1 week late, less than 2 weeks late.  More than 2 weeks late – no submission possible.  0 points |
| **Knowledge of Content**  (40%) | Lab completed the bulleted deliverables, all functionality implemented, program(s) works as it should  40 points | Missed one deliverable  Example:   * Used JDK 1.7 instead of JDK 1.8 * Missed JUnit test case(s)   20-35 points | Missed more than one deliverable  0 – 20 points |
| **Coding- Design or Runtime errors**  (35%) | No errors, program compiles and executes as expected  35 points | No more than two errors  20-35 points | More than two errors  0-20 points |