Employee Monthly Pay Slip Generator

ReadMe

Chola Kallepalli
DEVELOPER - srvarma7@gmail.com

Table of Contents

What is the application for?

This program is a console application that allows to generate monthly pay slip for an employee based on their annual salary related to its tax rates.

How does the program work?

After running the application using the Xcode IDE (Integrated development environment). The applications welcome's the user and displays the instructions to follow for generating monthly payslip.

The user simply has to type their input to the console is a specific manner. For example, if a user wants to generate monthly payslip for Mary Song who makes 60000 a year as salary. So, the user has to enter – `GenerateMonthlyPayslip “Mary Song” 60000`

Notice: - As you can observe the first field **GenerateMonthlyPayslip** is the one we want to generate (hence calling it as keyword throughout the application). The second field within the double quotes is the **employee's name**. And lastly, the field at the end is the **employee's salary per annum**.

The program checks if the entered data is valid or not. If valid, it calculates and displays the monthly payslip for Mary Song. If not valid, the application displays the error message based on the issue then asks user to re-enter the details.

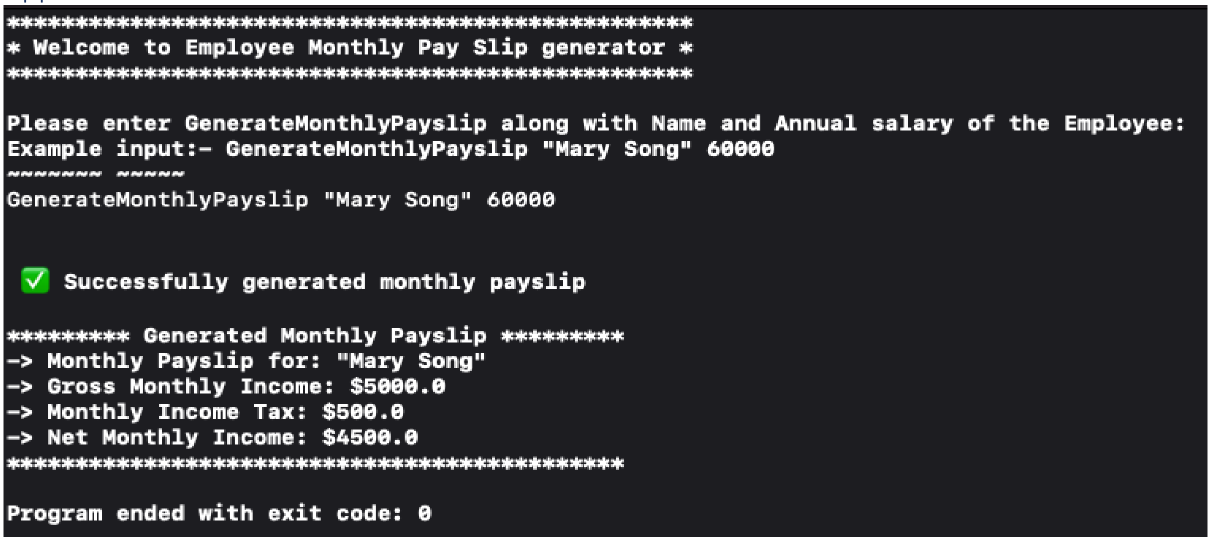
Example input

```
GenerateMonthlyPayslip “Mary Song” 60000
```

Example output

```
********* Generated Monthly Payslip *********
-> Monthly Payslip for: "Mary Song"
-> Gross Monthly Income: $5000.0
-> Monthly Income Tax: $500.0
-> Net Monthly Income: $4500.0
*****************************************
```

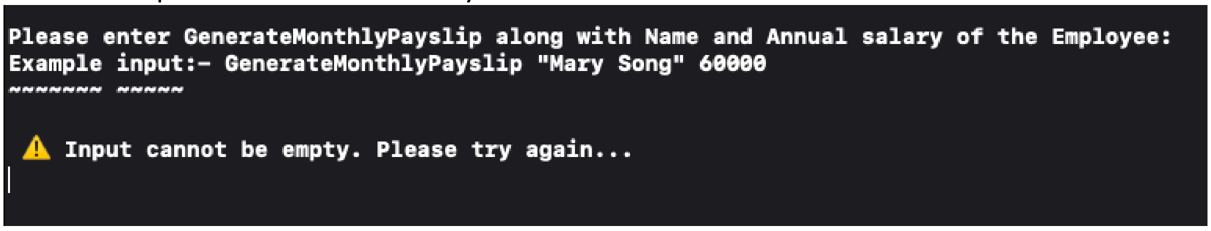
Application demo



```
****************************************************
* Welcome to Employee Monthly Pay Slip generator *
****************************************************

Please enter GenerateMonthlyPayslip along with Name and Annual salary of the Employee:
Example input:- GenerateMonthlyPayslip "Mary Song" 60000
~~~~~~~ ~~~~~
GenerateMonthlyPayslip "Mary Song" 60000


 ✅ Successfully generated monthly payslip

********* Generated Monthly Payslip *********
-> Monthly Payslip for: "Mary Song"
-> Gross Monthly Income: $5000.0
-> Monthly Income Tax: $500.0
-> Net Monthly Income: $4500.0
*********************************************

Program ended with exit code: 0
```

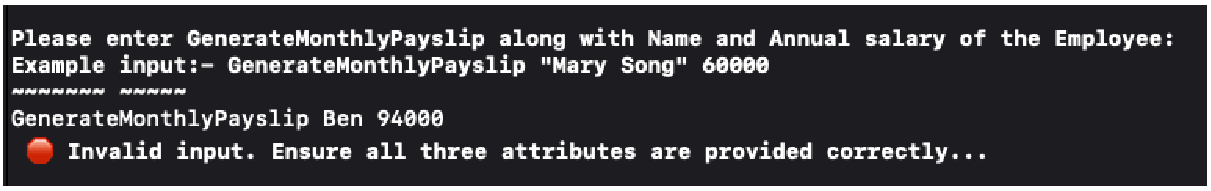
The above is a sample demonstration when the input meets the requirements.

A few examples of invalid data entry -

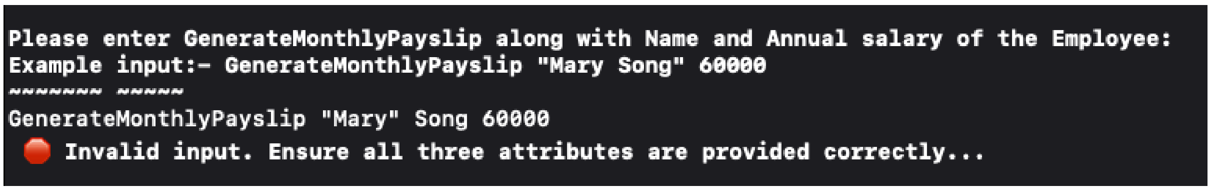


```
Please enter GenerateMonthlyPayslip along with Name and Annual salary of the Employee:
Example input:- GenerateMonthlyPayslip "Mary Song" 60000
~~~~~~~ ~~~~~

 ⚠️ Input cannot be empty. Please try again...
|
```

The above is a sample demonstration when the input is empty (nothing is entered to console).



```
Please enter GenerateMonthlyPayslip along with Name and Annual salary of the Employee:
Example input:- GenerateMonthlyPayslip "Mary Song" 60000
~~~~~~~ ~~~~~
GenerateMonthlyPayslip Ben 94000
 ⛔ Invalid input. Ensure all three attributes are provided correctly...
```



```
Please enter GenerateMonthlyPayslip along with Name and Annual salary of the Employee:
Example input:- GenerateMonthlyPayslip "Mary Song" 60000
~~~~~~~ ~~~~~
GenerateMonthlyPayslip "Mary" Song 60000
 ⛔ Invalid input. Ensure all three attributes are provided correctly...
```

The above are the sample demonstration when employee name entered incorrectly.

Warning: - When testing the application, it is essential to type the input in to the console rather than copying and pasting. If not done properly, the application is designed to force crash on purpose as we don't want to show wrong output that is especially related to financials.

Development environment

Environment – Mac
OS – macOS Big Sur
Version - 11.3.1

IDE – Xcode
Language – Swift
Version – 5.4

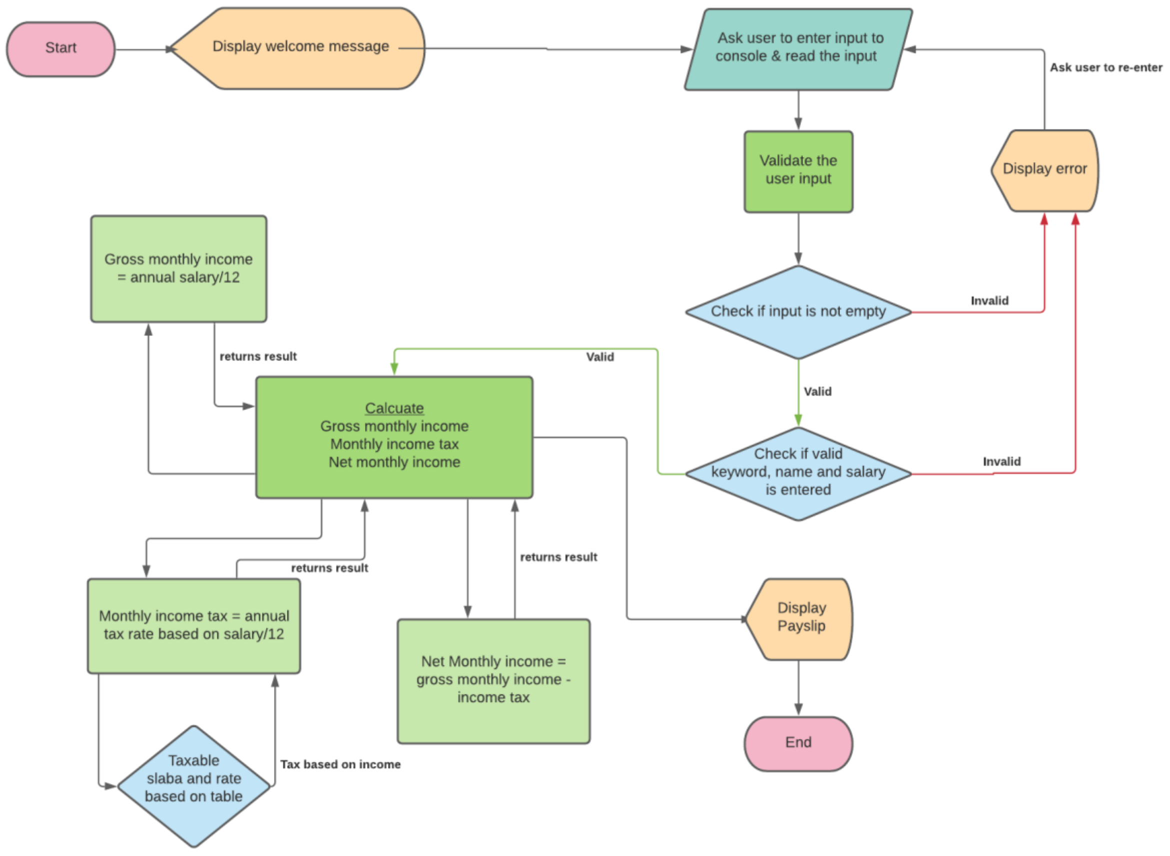
Designing the solution

Approach

When finding a solution for the problem it is always important to first understand the requirement and think of the different possible ways to achieve the goal.

As always, I tried to understand the requirement before beginning to work for a solution. After understanding the problem and how the tax is being calculated for different salary slabs (ranges), started working in creating a very high-level flowchart diagram.

Flowchart


Start
Display welcome message
Ask user to enter input to console & read the input
Ask user to re-enter
Validate the user input
Display error
Check if input is not empty
Invalid
Valid
Check if valid keyword, name and salary is entered
Invalid
Valid
Gross monthly income = annual salary/12
returns result
Calcuate
Gross monthly income
Monthly income tax
Net monthly income
returns result
returns result
Monthly income tax = annual tax rate based on salary/12
Net Monthly income = gross monthly income - income tax
Taxable slaba and rate based on table
Tax based on income
Display Payslip
End

Initial high-level flowchart

The program has been slightly modified during the development for improvements and based on the functionalities.

Ideology

Creating the skeleton of the application

First created the abstract classes for the application using protocol to define the required functionalities. Using the defined protocol, we can make the class confirm to protocol for autogenerating the helper methods in the codebase.

Distinguishing UI and program logic

The application is designed such a way that a particular class does its specific things rather than just populating with functions or variables as needed. For instance, the Display class is responsible for displaying/outputting the response or feedback on the console. It is not advised to have a functionality to take/read input from console. This helps us to achieve encapsulation.

Tax calculation

There are a number of ways to do calculate the tax amount based on the table provided in the requirement.
The process I have designed would take O(n) time complexity. It also adjusts to work if the tax slabs are changed in the Constants.

Pre-defined variables from Tax table
Tax rate – [0, 0.1, 0.2, 0.3, 0.4]
Tax slab base values – [0, 20000, 40000, 80000, 180000]

Process

1) Save the salary to a new variable for the use of calculating remaining salary.
2) Get the number of tax slabs available and store it as index. This index is used to loop through and get corresponding tax slab base value as well as tax rate.
3) Check if the salary is greater than 0 and the index is in valid range. If condition not met **skip to 4.**
    a. Get the current tax base slab value
    b. Check if the remaining salary is greater than current tax base slab. If condition not met **skip to c.**
        i. Subtract the remaining salary with current tax base slab to get the amount that is greater than tax base slab.
        ii. Calculate the tax for the obtained value using the current tax rate with the help of index.
        iii. Now add the tax amount to yearly income tax
        iv. Remove the taxed amount from the remaining salary.
    c. Regards less of the condition decrease the index value by 1.
4) Divide the obtained value by 12 to get income per month.

Error handling

Error handling is one of the important aspects while programming. It is useful to provide feedback to user about the issue and act accordingly. The error handling is taken care by catching the exceptions and displaying appropriate messages to the user in an informative way.

Design decision

The application is built using protocol-oriented mechanism. One powerful feature of Swift is its ability to extend protocols. Protocol-oriented programming takes that feature and help to craft the application architecture around it so that the first thigh you do is sketch out one or more protocols rather than get straight into concrete types. It is no different from other traditional programming languages where inheritance is more common.

It also follows the MVVM architecture. Model–View–ViewModel (MVVM) is a software architectural pattern that facilitates the separation of the development of the graphical user interface (the view) – be it via a console or GUI code – from the development of the business logic or back-end logic (the model) so that the view is not dependent on any specific model platform.

How to run the project

To run the project please follow the below steps.
It is advised to run the application on a Mac OS device.

Step 1	If you have Xcode installed on your computer, please skip to **Step 5** else follow the steps below.
Step 2	Open App Store on your mac. Search for Xcode and click download/get. The App Store may or may not prompt you to enter your Apple credentials. If you have issues downloading Xcode through Xcode follow Step 3.
Step 3	Please visit the link provided below to download Xcode from website. Link: - https://developer.apple.com/xcode/
Step 4	Then extract the downloaded file to Applications folder.
Step 5	Hurray!!! Your Xcode IDE is ready to run the project. Now, navigate to project file and open EmployeeMonthlyPayslip.xcodeproj using Xcode.

Running the code

Step 6	To run the application, simply press cmd (Command) + R together on your keyboard or click on play icon located at top right of the IDE.

Running the tests

Step 7	To run the test cases, press cmd + U.

Use the project navigator on the left to browse through the code files.
Should you have any issues, please reach me at srvarma7@gmail.com.

Assumptions

- The code is written and documented in a way to make the developer understand the process and programming language who is a complete beginner.
- While extracting the data from console provided by user, the name field is extracted without double quotes (“”) at the beginning and the ending.
- When printing the generated payslip, the name is wrapped in double quotes.

- Preconditions are used in few parts of the code to stop the program when something unanticipated is happening. For example, to go to the step 5 there are few conditions to be met in steps 1 to 4. So, in short step 5 cannot be achieved without following the previous steps.

Trade-offs

- The salary of the employee is always displayed as a number with decimals. From the example output, the numbers in the output looks like an integer type. But the output representing number is represented as a decimal number with precision two (Double type).
  Reason – Traditionally, the values used to represent financial data is always a decimal number.
  Example – Invoice, bank transactions, receipt, etc...

- When entering input to the console, please do not copy paste the values. Always write the input manually.
  Reason – Unknow, for some reason Xcode doesn’t take the input properly when copy pasting values to console.