**Salary Breakdown C# Application**

Salary breakdown application takes two input from user:

1. Gross Salary package figures
2. Pay frequency

The solution has been implemented using Interface, *ISalaryCaluclations* whichdeclares method *CalculateSalary(),* the interface has been implemented by class *SalaryCalculations* whichimplements the method *CalculateSalary(),* along withother member variables and internal methods.

The project contains following classes:

* ***Salary.cs***
* ***PayFrequency.cs***
* ***Superannuation.cs***
* ***TaxableIncome.cs***
* ***MedicareLevy.cs***
* ***BudgetReapirLevy.cs***
* ***IncomeTax.cs***
* ***NetIncome.cs***
* ***PayPacket.cs***
* ***SalaryCalculator.cs***
* ***Validation.cs***
* ***Exception Classes i.****e. (InputSalaryException.cs, CalculateSuperException.cs etc)*

**Salary class**: The Salary class contains lot of properties i.e. Salary Type, Gross Package, Superannuation, Medicare Levy, Budget Repair Levy, Income Tax, Taxable Income, Net Income, Pay Packets. The properties are updated accordingly once we have the calculated value for that property. It is easier to maintain the data through properties which can be easily retrieved or written back. To cater the future or past salary scenarios the class can be inherited as this is defined as a public class. Salary Type can be used to define the salary type i.e. Gross, Net, Commissions etc. for future references.

***PayFrequency****:* PayFrequency class is a Enum class specifying the frequency i.e. Weekly, fortnightly or Monthly.

***Superannuation:*** Superannuation classhas a read only member salary which provides Salary data to this class, the superannuation can be read or set using Superannuation property. For any future changes in Superannuation rates, the Superannuation percentage can be updated through this property. The other way to do it through Resource file in which we can maintain all static data and resource file can be updated once a while based on major changes.

It also contains CalculateSuperAmount method which calculates superannuation from gross salary and updates the “Superannuation” property of “grossSalary” object.

The class throws a CaculateSuperException if any exception occurs during Super calculations.

***TaxableIncome.cs:*** *TaxableIncome* class uses salary class object in the constructor and updates the “TaxableIncome” property in the Salary class object after calculations. The CalculateTaxableIncome method is used to calculate the Taxable Income. *CalculateTaxableIncomeException* exception is thrown if any exception occurs during processing.

***MedicareLevy.cs:*** *MedicareLevy* class references the Salary class object in the constructor and uses this object to update the properties of Salary class.

*CalculateMedicareLevy()* Method has been used to calculate the Medicare levy on the gross salary. Local properties have been used to get or set the percentage Medicare Levy. The idea behind that is, if in future Medicare Levy changes, it can be easily updated with the properties. The same can be achieved with the use of resource file in the project.

A custom exception *CalculateMedicareLevyException* has been thrown if any exception occurs during the calculations.

***BudgetReapirLevy.cs:*** BudgetRepairLevy class references the salary class object in the constructor and uses this to update the BudgetRepairLevy object of Salary class.

CalculateBudgetRepairLevy() method has been used to calculate the Budget Repair Levy. Local properties have been used to find the income range.

A custom exception CalculateBRLevyException has been thrown for exceptions occurred while calculating Budget Repair Levy.

***IncomeTax.cs:*** IncomeTax class references Salary class object in the Constructor Method. It uses local properties to set the income bracket ranges.

CalculateIncomeTax() method has been used to calculate the income tax figures and salary class object has been updated accordingly.

For any unknown exceptions, a custom exception CalculateIncomeTaxException has been thrown and handled.

***NetIncome.cs:*** NetIncome class is used to calculate the net income after all the deductions and super amount. It uses Constructor to pass on the salary class object which has been updated with Net income after all the calculations.

CalculateNetIncome() Method has been used to calculate the Net income on the Gross salary.

A custom exception CalculateNetIncomeException has been thrown for any unknown exceptions.

***PayPacket.cs:*** PayPacket class is used to calculate the total pay packet which is take home amount based on the specified pay frequency. The salary object is referenced through constructor method and Pay Packet property has been updated in the salary class object.

It has constant double variables mentioning the number of weeks, number of fortnights and number of months in the year.

CalculatePayPacket() method is used to calculate the Pay packet figures and salary object has been updated accordingly.

CalculatePayPacketException has been thrown for any unknown exceptions occurred in the processing.

***SalaryCalculator.cs:*** SalaryCalculations is one of the main classes of this project. It formulates the whole design of this application. It has lot of members and lot of functionality written in it. At start new objects of classes Salary, Super, Taxable Income, Medicare Levy, Budget Repair Levy, Income Tax, Net Income, Pay Packet, Validation classes have been instantiated.

RunApp() Method prints the input instructions to Console/User and calls different methods helping in calculating salary breakdowns.

ReadAndValidateSalary() method gets the salary input data from the console and passes the data to validation class for input validation.

ReadAndValidateFrequency() method gets the pay frequency input from the console and passes the data to validation class for pay frequency validations.

CalculateSalary() method calculates different components of the salary i.e. Super, medicare levy, tax etc.

PrintSalaryBreakdowns() prints the salary breakdowns back to the console in the formatted output.

***Validation.cs:*** Validation class checks the input data for correct input data.

ValidateSalary() method checks input data against Negative values, special characters, Alphabetic inputs, Zero Salary inputs, Infinity value salary input, Blank salary inputs etc. if the Validation is not passed then user has been asked to re-enter the Salary data unless correct data has been recorded.

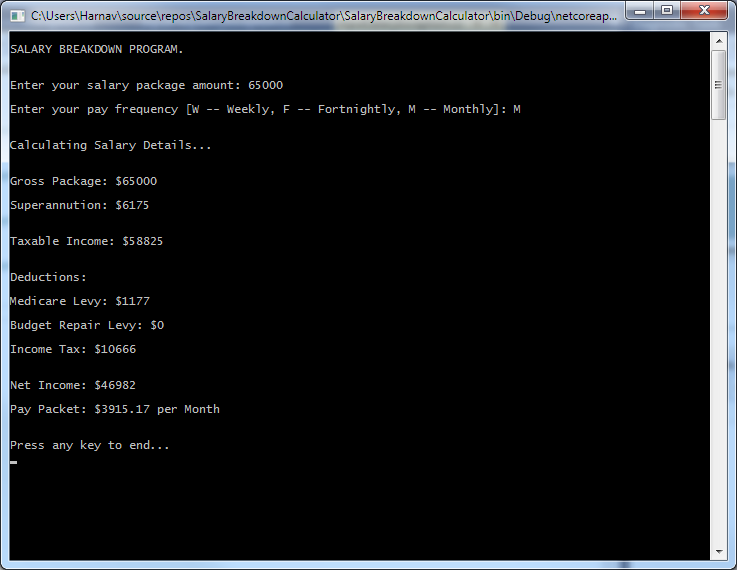
ValidateFrequencyInput() method checks input data for valid frequency options i.e. W, F and M (Week, Fortnight and Month). If any other input has been recorded then user has been asked to re-enter the correct option until correct data has not been recorded.

***Exception Classes:***Various Custom exception classes have been created to throw custom exceptions and let user be aware of any exceptions occurred in the program. Exception handlers help in smooth execution of the program.

Following classes have been created:

* *CalculateBRLevyException.cs*
* *CalculateIncomeTaxException.cs*
* *CalculateMedicareLevyException.cs*
* *CalculateNetIncomeException.cs*
* *CalculatePayPacketException.cs*
* *CalculateSuperException.cs*
* *CalculateTaxableIncomeException.cs*
* *InputFrequencyException.cs*
* *InputSalaryException.cs*

**Example output from Application:**



**Note:** The example output given in the Veritec Developer Interview code Assignment PDF document is incorrect as the Superannuation calculated at rate of 9.5% should be $6175 of $65000 Gross Salary where as it has been mentioned as $5,639.27.

**Disclaimer: I solemnly declare that the project work done is purely my own work and does not include any physical assistance from external sources. I am happy to provide further clarifications if required.**

**Tejpartap Singh.**