PORTFOLIO DOCUMENTATION

Object-oriented Programming

**Contract Manager Prototype**

By

Preethi Mandadi

(W9457747)

# Problem scenario:

A telecommunication company needs a prototype model to portrait the plans based on the several business types and frequency of service purchases.

Initial steps taken:

1. Designed the blocks based on the work and properties
2. Extracting different types of objects can be developed based on the concept
3. Also designed the utilities classes to support objects on regular transactions
4. Planned the services classes and main classes
5. Created the main objects that can be used as inheritance
6. Created the main class process so that services can be easily access and utilized in the program
7. Selected the type of datatypes should be used to store the list of the data.

The project needs to access with the local storage, and it should access the text files, information extracted from the files should be mapped with the objects in our class, on considering above all scenario’s the process is started.

All the programs start with the ContractManager class, and it has a service class ContractManagerServices that provides the all the functionalities to contract manager class

Created the utility class which supports the whole project with some basic functions.

Created the Company class that holds the information of the company plans and it performs the calculations of the plans based on the input

Created the Contract class, which is a basic type of contract object, and also created the Old contract class which represents the old contract objects.

# Implementation of menu options

## Enter New Contract

### Tools & techniques

* + - 1. This API creates the new contract into the contracts, rather than other API’s this option needs a high data entry process where user need to select the all the available options. This crates the record in the contacts.txt file
      2. Every input from the customer is validates before insert into the record
      3. Used the encapsulation techniques to validate the input

### Result

* + - 1. Programs perform as expected, the result is printing into a box and the record is saved to the contacts.txt
      2. Program options are working as expected, sometimes taking an extra space that creates the wrong input issue
      3. On providing the wrong selections throws the error

### Critique

* + - 1. Result is well presented
      2. More Object Oriented can be done
      3. Choose the exception handling while reading the inputs on wrong data inputs
      4. Selection of options are well planned

## Summary of Contracts

### Tools & techniques

* + - 1. This API shows the overall summary of the contacts of the year
      2. This API need a high computation rather that IO with the system
      3. Looping through the record and extraction the month and saving the records
      4. Used integer array datatype to save the month count
      5. Using a file reader to loop through the record of the file

### Result

* + - 1. Programs perform as expected, the result is printing into console that how many months has a contract and also displaying the analytics of the contracts
      2. Every functionality of this option is implemented successfully
      3. File handling errors may occur, and it is catched by the exception handlers

### Critique

* + - 1. Result of total contacts and contacts per month and average charge and high or unlimited packages data are presented accurately
      2. Usage of the preextracted records into a data structure collection may increase the performance.
      3. Searching optimization can be done by advanced methods.
      4. Choose the exception handling while reading the inputs on wrong data inputs

## Summary of contracts for selected month

### Tools & techniques

* + - 1. This API shows the overall summary of the contact of selected month
      2. This API need a computation
      3. Looping through the record and extraction the month and saving the records
      4. Declaring the pre values and increasing the count of it when record found of same month
      5. Using a file reader to loop through the record of the file
      6. Individual variables are used rather than array, values are added with utility object methods
      7. On reading the archive file records object inheritance is used to get the properties.
      8. Encapsulation is used

### Result

* + - 1. Programs perform as expected, the result is printing into console that how many records there per month and their average charges
      2. Every functionality of this option is implemented successfully
      3. File handling errors may occur, and it is catched by the exception handlers

### Critique

* + - 1. Result of total contacts and contacts in selected month and average charge and high or unlimited packages data are presented accurately
      2. For high records this may cause the time complexity, that may be reduced.
      3. Choose the exception handling while reading the inputs on wrong data inputs

## Find and Display Contract

### Tools & techniques

* + - 1. This API must find the record based on the name and Id
      2. Regex Pattern is used to search within the text of name and id
      3. Uses FileReader to read the file, and compare the text with id and name and prints the output as box
      4. New Contract object is created on finding the record and prints the summary with that object method

### Result

* + - 1. Programs perform as expected, the result is printing into console that how many records matches the name and id
      2. Creating the object and calling the method to print the record is implemented at best
      3. Space complexity can be reduced by crating the singleton object
      4. Every functionality of this option is implemented successfully
      5. File handling errors may occur, and it is catched by the exception handlers

### Critique

* + - 1. Searching is working perfectly as it was tested, regex search can be computed costly feature.
      2. Choose the exception handling while reading the inputs on wrong data inputs
      3. Flow of the code and validations are perfectly implemented
      4. View the record on creating new objects may cause the load, that can be optimized

## Modify existing contract

### Tools & techniques

* + - 1. This API must find the record based on the name and Id and also provide the options to display according to the number and allow the users to select a record and edit it
      2. Regex Pattern is used to search within the text of name and id
      3. ArrayList of string is used to store the id of the record and count number displays as it is.
      4. On selection of the option system again get the record and provides the option to edit the record as per plan conditions
      5. New Contract object is created on finding the record and prints the summary with that object method

### Result

* + - 1. Programs perform as expected, the result is printing into console that how many records matches the name and id with the count number
      2. Displaying the records count and validating with the count are projected correctly
      3. Space complexity can be reduced by crating the singleton object

### Critique

* + - 1. Searching is working perfectly as it was tested, regex search can be computed costly feature.
      2. Selecting the same record with the id and editing the same record are working flawlessly