DPIT121 – Lab Exercise 2

Due: Week 3 lab

In lab 2, you will continue on the same scenario from lab 1 to improve it. The improvement in term of software engineering is called Iteration. So you are working on another iteration of Car Insurance Provider Company System. Study week 2 lecture codes (Both bank and library examples) in depth before attempting this lab. Also watch Lab 2 Recorded Tutorial:

1) Add a new class **MyDate** with these attributes: 0.25 marks

- int year;
- int month;
- int day;

Add a field to your **InsurancePolicy** class to store the expiry date for the policy:

- MyDate expiryDate;

2) Add a new class **Address** with these attributes: 0.25 marks

- int streetNum;
- String street;
- String suburb;
- String city;

3) Add mutators (set methods), and assessors (get methods) to your **Car**, **MyDate**, **Address**, and **InsurancePolicy** classes.

4) Add these methods to your **InsurancePolicy** class: 1.5 marks: 0.3 marks for each method

- static void printPolicies(ArrayList< InsurancePolicy > policies) // prints a list of policies. You had this in your main ( ) in lab 1. Move it to this method. Study *printAccounts* in Bank lecture code or *printBooks* or *printUsers* in Library lecture code.

- static double calcTotalPayments (ArrayList< InsurancePolicy> policies, int flatRate) //calculates the total premium payments for a list of policies. You had this in your main ( ) in lab 1. Move it to this method. Study *calcTotalBalance* in Bank lecture code.

- void carPriceRise(double risePercent) // It has one parameter, a price rise in percent. The method increases the policy's car price by rise percent. (e.g., 0.1 as rise percentage means the car price to be increased by 10% i.e., $price = price \, x \, 1.1$).

Bad Design: You can call Car *setPrice* and *getPrice* methods for policy's car to do it as below:

car.setPrice(car.getPrice()(1+risePercent);*

Good Design: Add a new method *priceRise(double rise)* to Car and call this method in *carPriceRise*

- static void carPriceRiseAll(ArrayList< InsurancePolicy > policies, double risePercent) // calls the *carPriceRise* method for all the policies in a given list (in a for each loop). This is to increase the price of cars for all policies in a list.

- static ArrayList<InsurancePolicy> filterByCarModel (ArrayList<InsurancePolicy> policies, String carModel) // which filter a list of policies and creates a filtered list of policies, all with the given car model . See *filterByName* and *filterByAuthor* in Library example.

5- Write a class **User** with the following fields and methods: 1.5 marks

- private String name; // the name of the account holder
- private int userID; // the user ID/number
- private Address address; // you need to define the Address class as described
- ArrayList< InsurancePolicy> policies // list of all the Insurance Policies this user holds
- Constructor, mutators (set methods) and assessors (get methods) if necessary and not for all fields.
- boolean addPolicy (InsurancePolicy policy) // adds a policy, returns true if successful (when policyID is unique) and returns false if not. See *addBook()* and *addUser()* methods in Library example
- InsurancePolicy findPolicy (int policyID) // finds the policy and returns it. Returns null if policyID does not exist. See *findBook()* and *findUser()* methods in Library example
- void print() // prints all the information of this user including all the policies information
- String toString() // converts the user and his/her policies to String
- void printPolicies(int flatRate) // prints all the policies this user owns as well as the premium payment for each policy by calling the corresponding static method inside InsurancePolicy (add new one to print the list as well as the premium) or just call print and calcPremium methods inside InsurancePolicy in a loop.
- double calcTotalPremiums (int flatRate) // returns the total premium payments for this user by calling the corresponding static method inside InsurancePolicy.

- void carPriceRiseAll (double risePercent) // calls the corresponding static method inside InsurancePolicy to increase the car price for all the policies the user owns
- ArrayList< InsurancePolicy > filterByCarModel (String carModel) // filters the policies and returns a list of policies with the car model containing the given carModel by calling the corresponding static method inside InsurancePolicy.

6- Add this test code to your main: 1.5 marks: 0.1 marks for each test item excluding creating the policies

- Create few third-party and comprehensive polices and one user in your main.

- Add the policies to the user by using *addPolicy( )*

- call the print method for the user (note that *user.print ( )* also prints all the policies)

- print the user by using *toString()* (Note that toString includes all the policies' details in addition to user information)

- Find a policy by using *findPolicy()* for a given policy ID when the ID is not valid and show an error message: "Policy has not been found"

- Find a policy by using *findPolicy()* with a given policy ID (valid) and save it in a policy object to be used for following steps.

    - print this policy, call carPriceRise with 0.1 rise for this policy and print it again

    - change the policyHolderName of this policy to "Robert" by using setPolicyHolderName (String newName)

    - change the car model of this policy to "Toyota Camry 2018" by using *Policy.SetCarModel(String model)* which calls *Car.setModel(String model)*

- change the city of the user to "Wollongong" by using *User.SetCity(String city)* which calls *Address.setCity(String city)*

- ask the customer to enter the information for a new address (by using Scanner) and change the address of the user by using *setAddress(address)* and print the user after change.

- print the total premium payments for all policies this user owns

- add 10% to the price of cars for all the policies this user owns

- print the total premium payments for all policies this user owns again

- ask the customer to enter a carModel then call filterByCarModel method for the user and store the filtered list.

- print the filtered list by calling the static method inside InsurancePolicy