

## DPIT121 – Lab Exercise 3

**Due: Week 4 lab**

In lab 3, you will continue on another iteration for the same scenario from lab 1 and 2.

**1) Add these methods to your `InsurancePolicy` class: 0.25 marks**

- static `ArrayList<InsurancePolicy> filterByExpiryDate (ArrayList<InsurancePolicy> policies, MyDate date)` // which filter a list of policies and creates a filtered list of policies that are expired by the given date. To do this add a new method `Boolean isExpired( MyDate expiryDate)` to your `MyDate` class, which compares the given date with the object and returns false if the date is before that given expiryDate or true it is after that date.

**2) Add these methods to your `User` class: 0.75 marks: 0.25 marks for each method**

- boolean `createThirdPartyPolicy( String policyHolderName, int id, Car car, int numberOfClaims, MyDate expiryDate, String comments )` // creates a Third-Party Policy and adds it to the list of the user's policies, returns false if the id is not unique. Create an object from `ThirdPartyPolicy` and call `addPolicy` method to add it to the list
- boolean `createComprehensivePolicy( String policyHolderName, int id, Car car, int numberOfClaims, MyDate expiryDate, int driverAge, int level)` // creates a Comprehensive Policy and adds it to the list of the user's policies, returns false if the id is not unique. Create an object from `ComprehensivePolicy` and call `addPolicy` method to add it to the list
- `ArrayList<InsurancePolicy> filterByExpiryDate (MyDate date)` // filters the policies and returns a list of policies with the expiry date before the given date by calling the corresponding static method inside `InsurancePolicy`

3) Write a class **InsuranceCompany** with the following fields and methods: 2 marks. 0.1 mark for each method

- String name; ✓
- private ArrayList<User>users // list of all the users having a policy with the company ✓
- private String adminUsername ✓
- private String adminPassword; ✓
- private int flatRate; ✓
- Constructors, mutators (set methods) and assessors (get methods) if necessary ✓
- boolean validateAdmin( String username, String password) // returns true if username and password matches the admin login details ✓
- boolean addUser(User user) // adds the user to users list if userID is unique, if not returns false ✓
- Boolean addUser(String name, int userID, Address address) // creates and adds the User to users list if userID is unique, if not returns false. Create a user object and reuse the addUser(User user) method ✓
- User findUser(int userID) // finds the user with the given ID or returns null if user does not exist ✓
- boolean addPolicy (int userID, InsurancePolicy policy) // finds the user with the given userID by using findUser method and adds the policy to the user, unsuccessful if userID does not exist or policy is not unique ✓
- InsurancePolicy findPolicy (int userID ,int policyID) // finds the insurance policy for the given userID and returns it. Returns null if userID does not exist or policyID does not exist for the given user ✓
- void printPolicies(int userID) // prints the user information and all the policies for the given userID ✓
- void print() // prints all the users and for each user all the policies by calling User. PrintPolicies(int flatRate) ✓

- `String toString()` // converts the whole object to string ( including all the users and their policies). Hint: call `toString()` for users in a loop and concatenate them
- `boolean createThirdPartyPolicy(int userID, String policyHolderName, int id, Car car, int numberOfClaims, MyDate expiryDate, String comments)` // finds the user with the given userID (by calling `findUser`) and calls the `createThirdPartyPolicy` for that user. Returns false if the user does not exist or if `User.createThirdPartyPolicy` returns false ✓ Change comments for construct
- `boolean createComprehensivePolicy( int userID, String policyHolderName, int id, Car car, int numberOfClaims, MyDate expiryDate, int driverAge, int level )` // finds the user with the given userID (by calling `findUser`) and calls the `createComprehensivePolicy` for that user. Returns false if the user does not exist or if `User.createComprehensivePolicy` returns false ✓
- `double calcTotalPayments(int userID)` // returns the total premium payments for the given user ✓
- `double calcTotalPayments ()` // returns the total premium payments for all the users in the company ✓
- `boolean carPriceRise (int userID, double risePercent)` // calls `carPriceRiseAll` method for the given user. Returns false if user cannot be found ✓
- `void carPriceRise(double risePercent)` // Raise the price of all cars for all users in the company ✓
- `ArrayList<InsurancePolicy> allPolicies ()` // returns a list of all the policies in the company across all users ✓
- `ArrayList<InsurancePolicy> filterByCarModel (int userID, String carModel)` // find the user by calling `findUser` and calls `filterByCarModel` for the given user ✓
- `ArrayList<InsurancePolicy> filterByExpiryDate (int userID, MyDate date)` // find the user by calling `findUser` and calls `filterByExpiryDate` for the given user ✓
- `ArrayList<InsurancePolicy> filterByCarModel (String carModel)` // filters all the policies in the company by carModel across all users. Iterate over a loop for all users and for each user call the `filterByCarModel` method and add all the results together for a global list including all users. ✓
- `ArrayList<InsurancePolicy> filterByExpiryDate (MyDate date)` // filters all the policies in the company by ExpiryDate across all users. The same as above ✓

4) Add this test code to your **main**: 2 marks

- 1- You have one user from lab 2. Create few more users and more policies and one insuranceCompany in your main ✓
- 2- Login to the insuranceCompany once successful and once not successful with proper prompting ✓
- 3- Add users to the insuranceCompany by using both versions of addUser() and at least once not successful with prompting ✓✓
- 4- Add several polices to the users by calling addPolicy (int userID, InsurancePolicy policy), at least once not successful with wrong userID and once with duplicate policy with prompting ✓✓
- 5- Add several polices to some users by calling createThirdPartyPolicy() and createComprehensivePolicy() , at least once not successful with wrong userID and once *Add more* with duplicate policy ID with prompting ✓✓
- 6- Ask customer to enter a userID and print the user and all of his policies by using methods inside insuranceCompany ✓
- 7- Ask customer to enter a userID and policyID and find a policy with the given policyID for that userID by calling findPolicy (int userID ,int policyID) and then print the policy. ✓
- 8- Print all the users inside the insuranceCompany ✓
- 9- Raise the price of cars for all users and policies by 10% and print the users again ✓
- 10- Print the total premium payments for a given userID by calling calcTotalPayments (int userID) ✓
- 11- Print the total premium payments for all users in the company ✓
- 12- Call allPolicies() for the insuranceCompany and store it in an ArrayList and print the list by using InsurancePolicy.printPolicies() ✓

- 13- For a given userID and expiry date call filterByExpiryDate (int userID, MyDate date), store the filtered list and print the list by using InsurancePolicy.printPolicies() ✓
- 14- For a given car model call insuranceCompany.filterByCarModel (String carModel) ✓ and print the filtered list
- 15- Ask user to enter a date (year, month, and day) and call filterByExpiryDate (MyDate date) and print the filtered list ✓
- 16- Find a user with the given ID (valid) and save it in a user object. Ask user to provide a new address and change the current address for the given user ✓