***Documentation for Insurance Premium Calculator System***

**1. Design Choices**

**Object-Oriented Design: We designed this project using the principles of Object-Oriented Programming (OOP) to make it flexible and easy to add new features. Each main part of the system—like Customer, Policy, and Premium Calculator—is represented by a class, which makes it easier to work with each part separately.**

**Using Inheritance: We created a Policy base class with common features and built specific types of insurance (like Health Insurance, Auto Insurance, and Life Insurance) as subclasses. This lets each insurance type have its own premium calculation method, making it easy to manage different types of policies without repeating code.**

**Organizing Data Safely: Each class has its own data stored privately, meaning that the information can only be accessed through specific methods in that class. This keeps the data safe and ensures it’s used correctly.**

**Using Enumerations for Conditions: For health insurance, we used something called an Enum (Medical Condition), which defines different medical conditions that impact the premium amount. This makes it easy to add new conditions or adjust the premium for each condition in the future.**

**Type Hints and Comments: We added type hints (for example, age: int) and comments to make the code easy to understand for future developers, helping them know what type of data to use and what each part of the code does.**

**2. Challenges**

**Flexible Premium Calculation: Each type of insurance needs a unique formula for calculating premiums. For example, car insurance premiums depend on driving history and age, while health insurance considers medical conditions. To handle this, each policy type has its own calculate premium method, making it easy to customize the formula for each type of insurance.**

**Expanding to Add New Policies or Conditions: Since we wanted the system to be flexible, it was designed to support adding new types of insurance or adjusting conditions easily. By using inheritance (the Policy base class), new policies can be added by simply creating a new subclass of Policy.**

**Validating Data: We needed to ensure that customer and policy information was entered correctly to avoid errors. The Premium Calculator class helps validate that all necessary data is included when calculating premiums, making sure the inputs are correct.**

**3. Meeting the Requirements**

**Managing Customer Information: The Customer class holds information about the customer, like ID, name, age, and address. It also has a way to add multiple insurance policies to a customer’s record, so we can keep track of all policies they have.**

**Choosing Policies and Calculating Premiums: Each policy subclass (Health Insurance, Auto Insurance, and Life Insurance) has its own premium formula, based on factors that matter for that type of insurance. For example, health insurance uses age and medical conditions, while auto insurance looks at driving history. This meets the requirement to calculate premiums based on different factors.**

**Generating a Quote: The Premium Calculator class has a method called generate quote that uses customer and policy information to calculate and display a premium quote. This lets customers get an estimated price based on their specific data and insurance needs.**

**Easy to Expand: By organizing the code into classes and using inheritance, the system is designed to be flexible. New types of insurance can be added, or changes to premium calculations can be made without changing the core parts of the code, making it easy to maintain and expand.**

**4. UML Class Diagram**

**The UML diagram helps visualize the relationships between the parts of the system. Here’s a breakdown:**

**Classes: Key classes are Customer, Policy, Health Insurance, Auto Insurance, Life Insurance, and Premium Calculator.**

**Attributes: Each class has its own specific attributes, like customer\_id, name, age, and address for Customer, and policy type and base premium for Policy.**

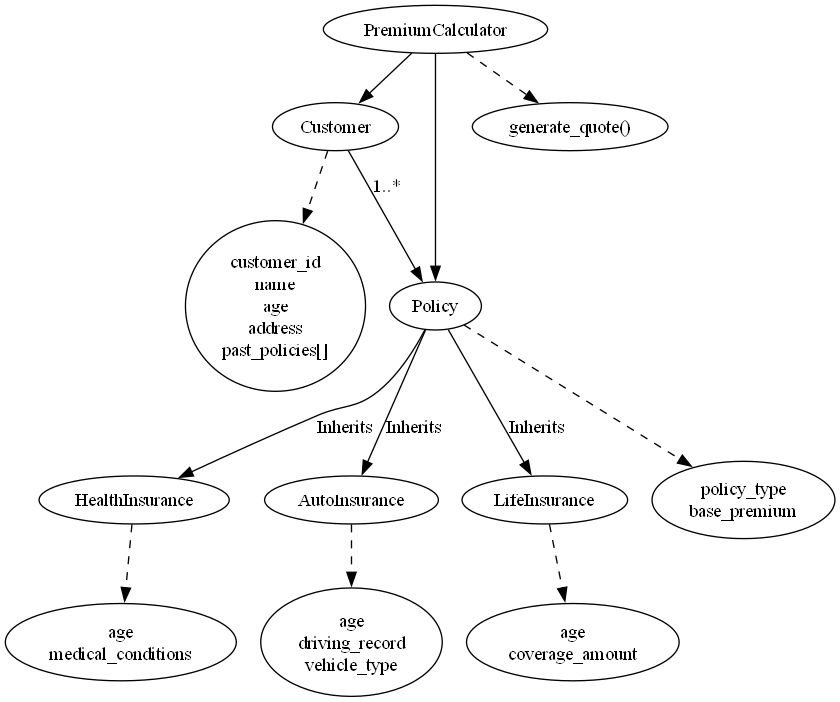
**Methods: Each policy subclass has a calculate premium method to compute the premium based on relevant data.**

**Relationships:**

**Customer can have multiple Policy objects (represented as 1..\*), meaning each customer can hold multiple policies.**

**Premium Calculator connects to both Customer and Policy to calculate and display the quote.**

**Health Insurance, Auto Insurance, and Life Insurance inherit from the Policy class, meaning they get some basic features from Policy but also add their own details.**

****

5. Conclusion

This system is designed to be easy to expand and maintain. Each part of the system works together to allow customers to get premium quotes for different types of insurance based on their personal information. The structure makes it easy to add new types of insurance in the future or adjust premium calculations without major changes to the code. The use of Object-Oriented Programming principles like inheritance and encapsulation helps keep the code organized and flexible.

Mahmoud Khatib