70083 ExerciseTypes.CW4

Car Loans (UML)

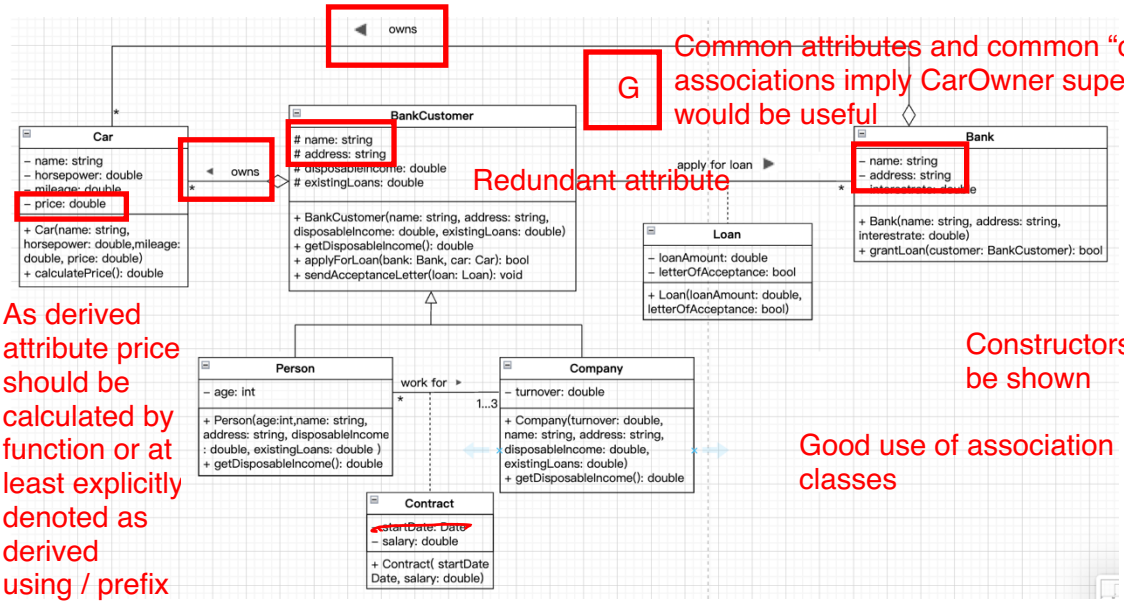
Submitters

sf23	Shihan Fu
zh3423	Zhuofan Huang

14/20 Good effort

Report for Car Loans UML design

group member: Zhuofan Huang, Shihan Fu



According to the description of the scenario, we designed a simplified financial system, including 'BankCustomer', 'Contract', 'Car', 'Loan', and 'Bank', which are intended to model interactions between individuals, companies, banks, and loans. Below is a report describing the main design decisions and justifications for the design:

1. Class Hierarchy:

- The core classes in the design are 'BankCustomer', 'Car', 'Loan', and 'Bank'.
- 'BankCustomer' is an abstract class which stores the common attributes of individuals, which can be refined to 'Person' and 'Company'.
- 'Car' represents information about a vehicle, including its number plate, horsepower, and mileage.
- 'Loan' represents a loan with attributes such as amount of loan. It is associated with a 'BankCustomer' and a 'Bank'.
- 'Bank' is responsible for granting loans to its customers based on their financial eligibility.
- 'Contract' represents a relationship between a person and a company

2. Relationships:

- 'BankCustomer' can apply for multiple loans from bank and a bank can grant multiple loans from customers. In the many-to-many associations, we use 'Loan' to position the property from both ends.
- 'Company' has a one-to-many relationship with 'Person' because a company can have multiple employees. And a 'Person' can have up to 3 'Company'.

3. Constructors and Member Functions:

- Each class has a constructor to initialize its attributes.

- ‘getDisposableIncome’ in ‘BankCustomer’ calculates the disposable income and should be overwrote in subclasses according to various calculation logics.

- ‘calculatePrice’ in ‘Car’ calculates the price of the car based on horsepower and mileage.

- ‘grantLoan’ methods in ‘Bank’ associate loans with a ‘BankCustomer’. Its return value is a bool, indicating whether the loan is granted based on disposable income and amount for loan.

- ‘sendAcceptanceLetter’ is triggered when a bank grants one loan.

The implementation of the codes of the design can be found: *https://github.com/Randi-f/70083PPP/tree/main/CW4-UML*

Could not access this. Probably using gitlab.doc.ic.ac.uk is a better idea.

4. Design Justifications:

- The design separates concern effectively. Each class has a specific role and responsibility.

- Encapsulation is maintained by keeping data members private and providing methods for interacting with the objects.

- The design reduces redundancy to build one superclass to store common attributes for a ‘BankCustomer’.

- The ‘getDisposableIncome’ method overwritten in ‘Person’ and ‘Company’ ensures that their financial status is evaluated independently.

- ‘Bank’ is responsible for assessing loan eligibility and creating loan objects, following the Single Responsibility Principle.

5. Potential Improvements:

- The code lacks proper error handling and validation for cases like dividing by zero, which should be addressed for robustness.

- The design could benefit from more documentation and comments to enhance code readability and maintainability.

- The design might need further consideration for handling multiple loans for the same entity and loan management.

In conclusion, the provided design represents a basic financial system with clear relationships and responsibilities among the classes. It separates concerns and allows for flexibility in handling loans for both individuals and companies. However, it could benefit from additional error handling and documentation for a more robust and maintainable implementation.