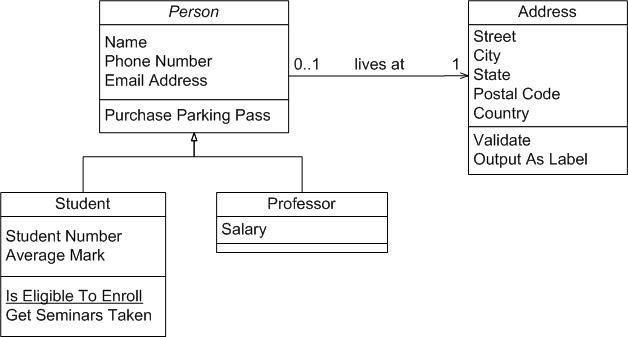
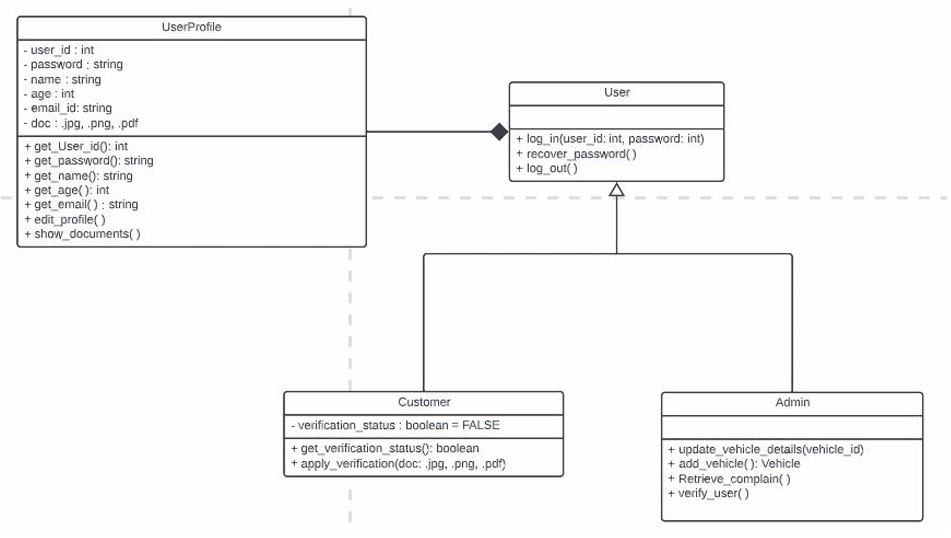
**UTS QUESTIONS**

**OBJECT-BASED PROGRAMMING PRACTICUM**

1. Identify the following diagram class, make complete improvements and in accordance with the rules for writing the diagram class.



1. Create a diagram class that uses multilevel inheritance and create the program code!
2. Please identify the class diagram by providing an explanation of the concept of inheritance, the relationship between classes and the following system flow, create a program code from the following class diagram!



**---- Good Luck ----**

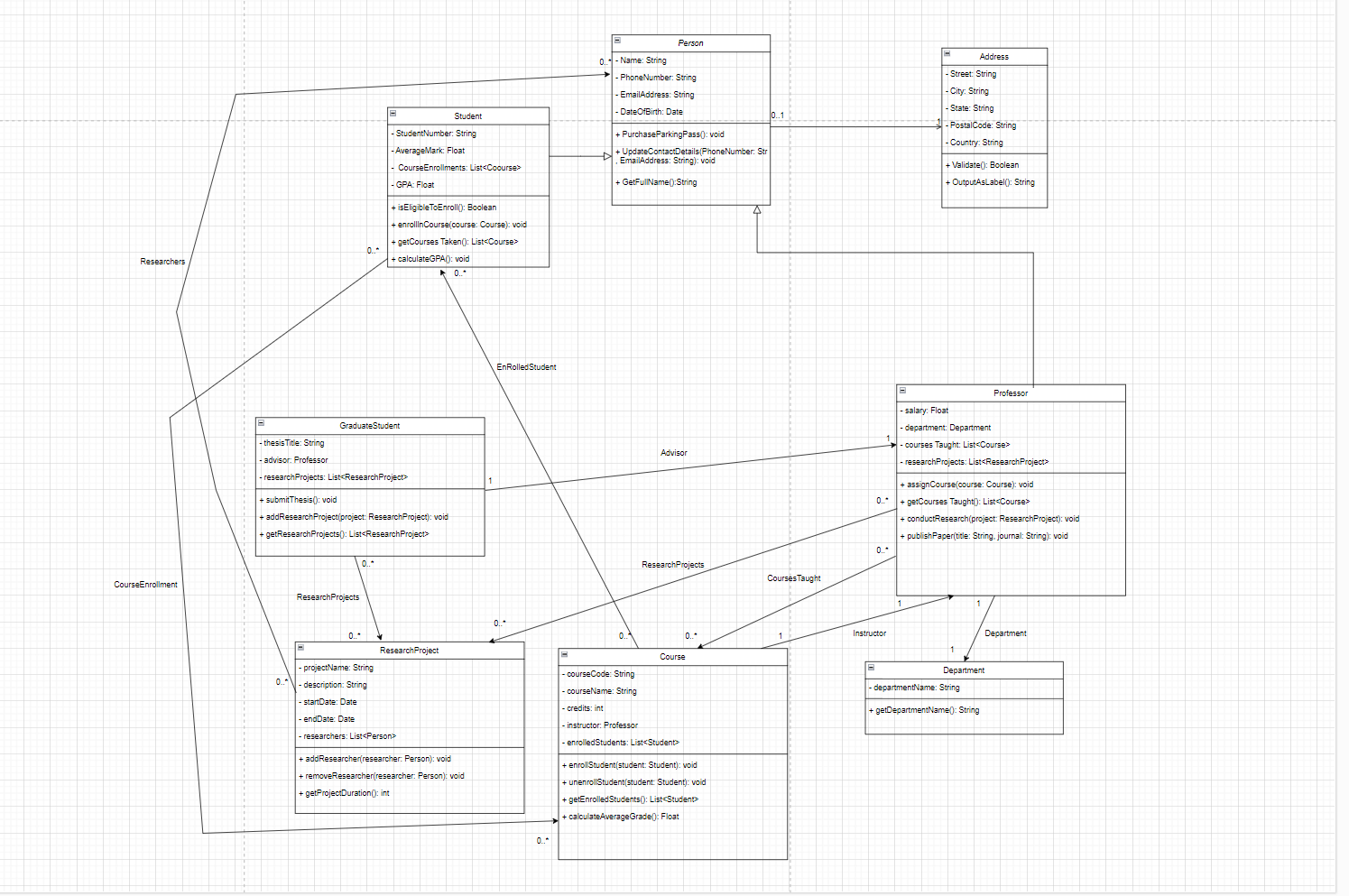
**AnsWer**

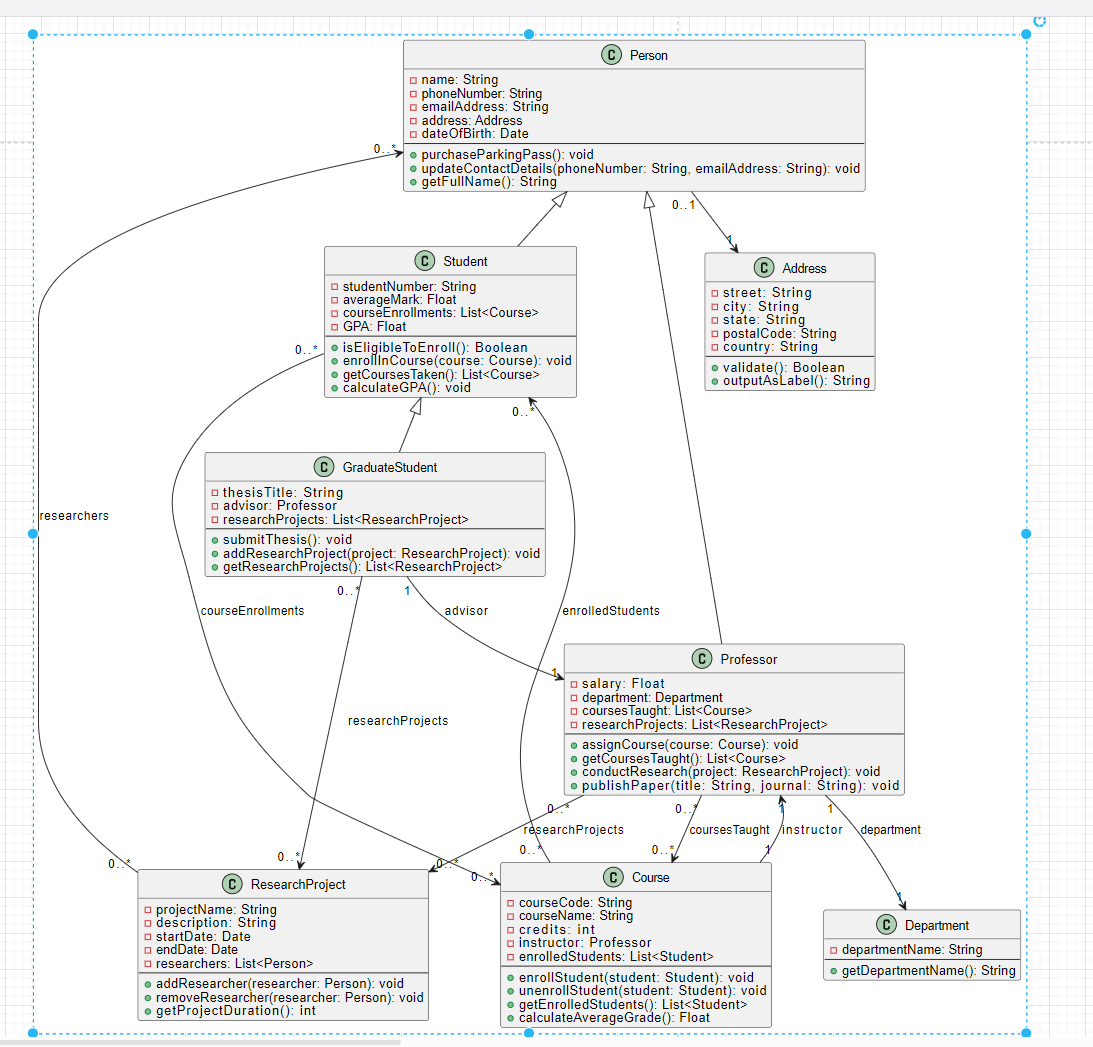
Link Github : <https://github.com/safrizalrahman46/UTS_PBO_SAFRIZALRAHMAN-SEMESTER3.git>

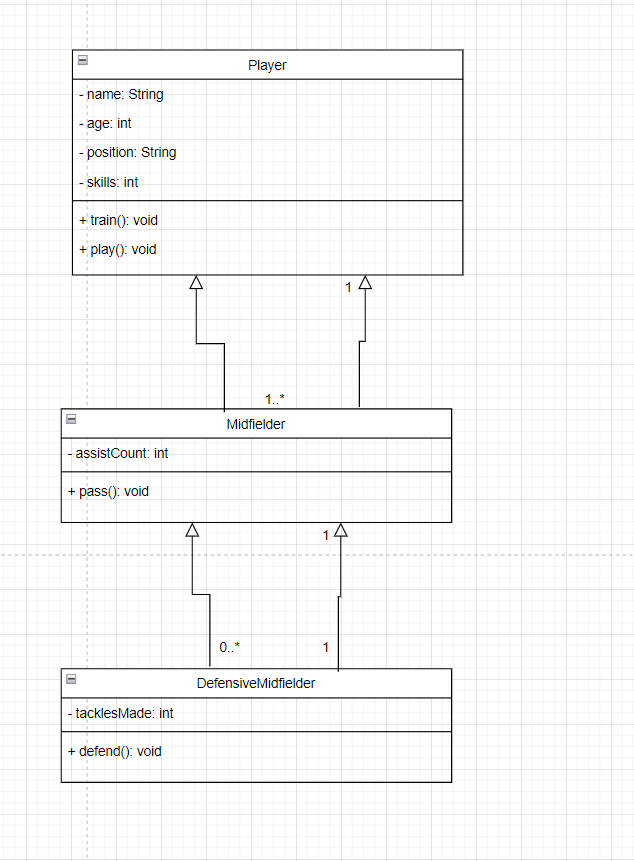
Link Drive File Class Diagram :

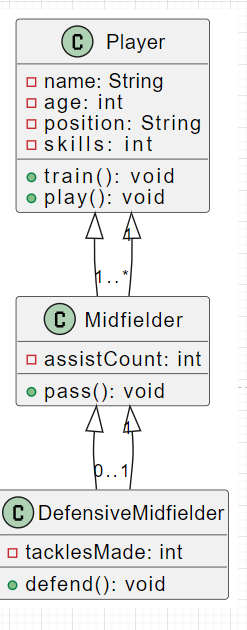
[https://app.diagrams.net/#G1FMGxTby-FuFdYJW3jodTg-PsBbUDPY6h#%7B%22pageId%22%3A%22C5RBs43oDa-KdzZeNtuy%22%7D](https://app.diagrams.net/#G1FMGxTby-FuFdYJW3jodTg-PsBbUDPY6h)

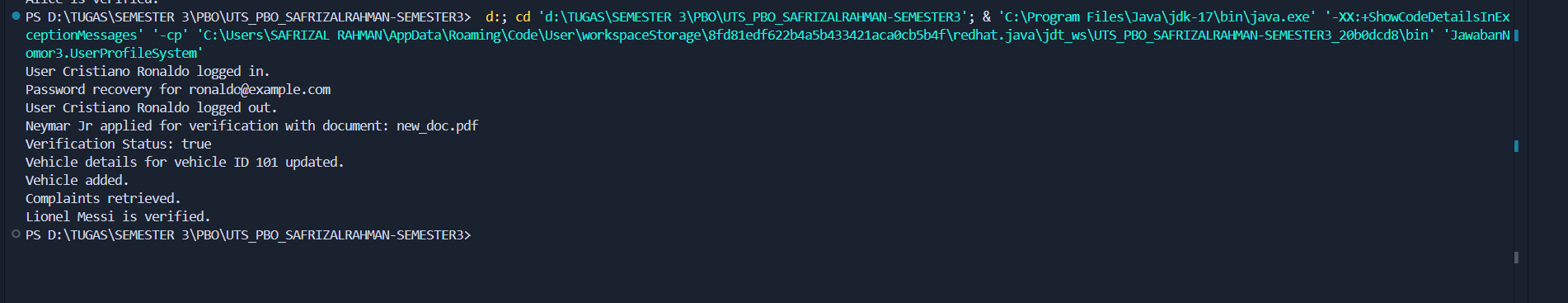
NOMOR 1

****

****

2. 



3. 

Class Diagram Overview

**Explanation:**

1. **UserProfile**: The base class with common attributes and methods.
2. **User**: Inherits from UserProfile and adds methods for user-specific actions like login, recover password, and logout.
3. **Customer**: Inherits from UserProfile and includes methods specific to customer actions, like applying for verification.
4. **Admin**: Inherits from UserProfile and includes methods for administrative actions like updating vehicle details, adding vehicles, retrieving complaints, and verifying users.

Inheritance Concept

Inheritance is a fundamental principle of object-oriented programming (OOP) that allows a new class, known as a subclass, to inherit attributes and methods from an existing class, referred to as a superclass. This mechanism promotes code reuse and establishes a hierarchical relationship among classes. By using inheritance, developers can create more specialized classes that build upon the functionalities of general ones.

Relationship Between Classes

In this system, the following classes are defined:

* **UserProfile Class**: This is the base class that contains common attributes and methods applicable to all user profiles.
* **User Class**: Inherits from UserProfile and includes methods specific to user actions, such as logging in, recovering passwords, and logging out.
* **Customer Class**: Also inherits from UserProfile, adding attributes and methods related to customer-specific actions like applying for verification.
* **Admin Class**: Inherits from UserProfile and includes methods for administrative tasks such as updating vehicle details and verifying users.

System Flow

The interaction among these classes can be summarized as follows:

* **UserProfile**: Serves as the foundational class, providing common functionalities to all user types.
* **User**: Extends UserProfile to incorporate login-related functionalities.
* **Customer**: Extends UserProfile to include functionalities related to user verification.
* **Admin**: Extends UserProfile to encompass administrative functionalities.

Summary

The inheritance structure allows for efficient organization of user-related functionalities while maintaining clarity in the relationships between different user types. Each subclass inherits the base attributes and methods from UserProfile, ensuring that common functionalities are centralized, which simplifies maintenance and enhances code reusability.