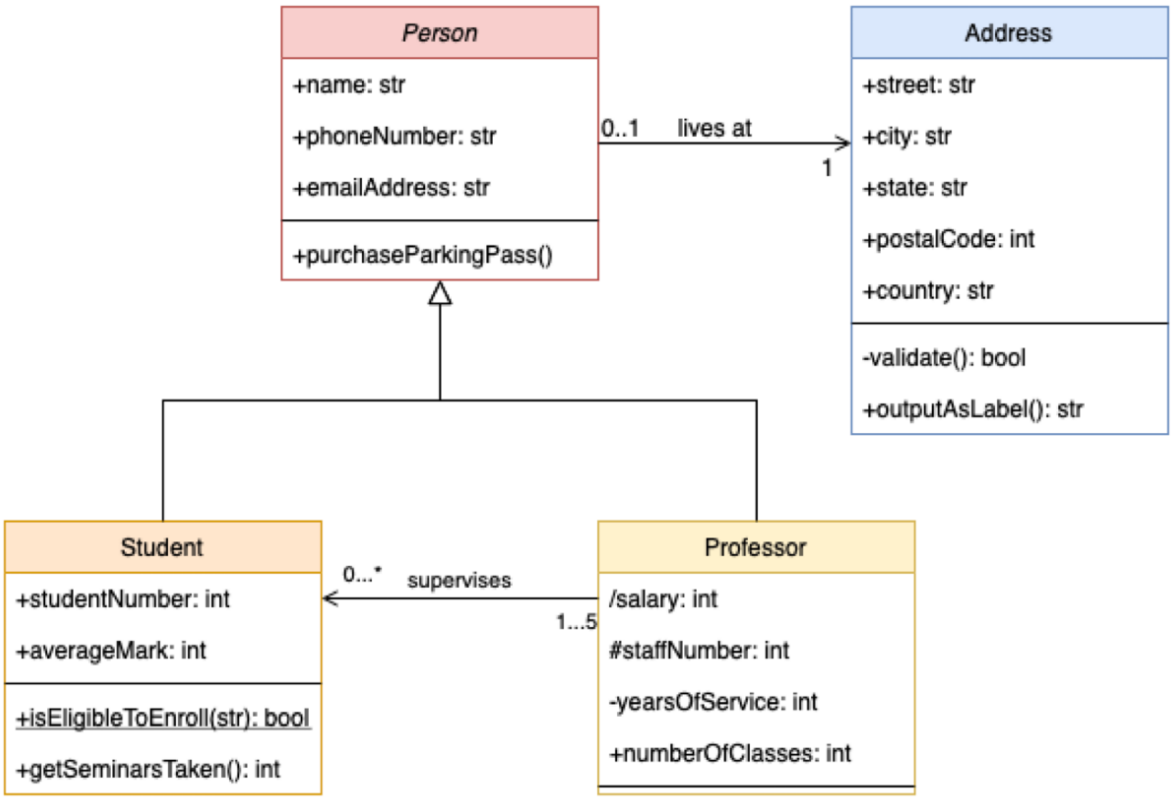
주제 및 내용: Consider the following class diagram for a UNIVERSITY management system



Based on the UML class diagram described, I can see how a university management system could be structured within a computer program. This structure reflects the hierarchy and associations among the different entities, such as "Person," "Student," "Professor," and "Address."

- **Person:** This class acts as a base class for Student and Professor. It likely contains attributes common to both students and professors such as name, phone number, email address, and an address (including street, city, state, postal code, and country). It also has a method **validate()** that likely checks if the entered data adheres to some set of rules (e.g., email format, phone number format, etc.) and an **outputAsLabel()** method that presumably returns a string representation of the person's information in a certain format.
- **Student:** This class inherits from Person and likely includes attributes specific to students such as student number and average mark. It also has methods like **isEligibleToEnroll()** that checks if a student meets the enrollment criteria (likely based on their average mark) and **getSeminarsTaken()** that retrieves the number of seminars a student has taken.
- **Professor:** This class also inherits from Person and likely has attributes specific to professors such as staff number, years of service, and the number of classes they teach.

- **Address:** This class likely models an address and has attributes such as street, city, state, postal code, and country.

✧ **Relationships:**

- **Person** has a relationship with **Address**. This is likely an association where a person has one address (e.g., lives at).
- **Professor** supervises 0 to 5 **Student**s. This could indicate a professor can advise multiple students but a student can only be advised by one professor (one-to-many relationship).

✧ **Structure of the Corresponding Program:**

The classes likely translate to real-world entities in the University Management System. The program would likely have separate modules/classes to manage each class (e.g., StudentManager, ProfessorManager) These modules would handle CRUD (Create, Read, Update, Delete) operations specific to their domain (e.g., StudentManager might have functionalities to add a new student, update a student's information, or delete a student record).