



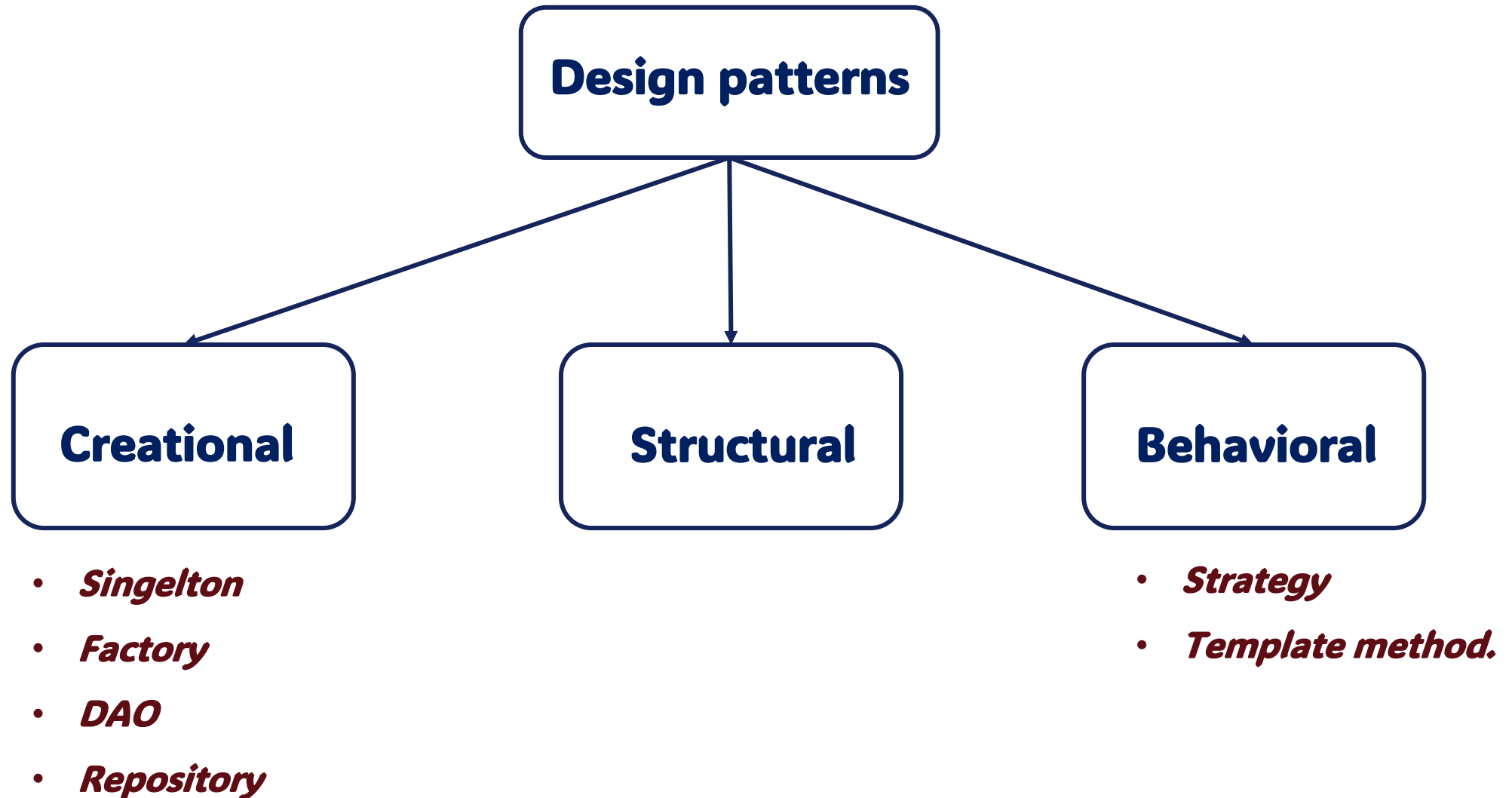
الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

Software system design – practical

Lecture 04 – Data Access Layer
Design patterns

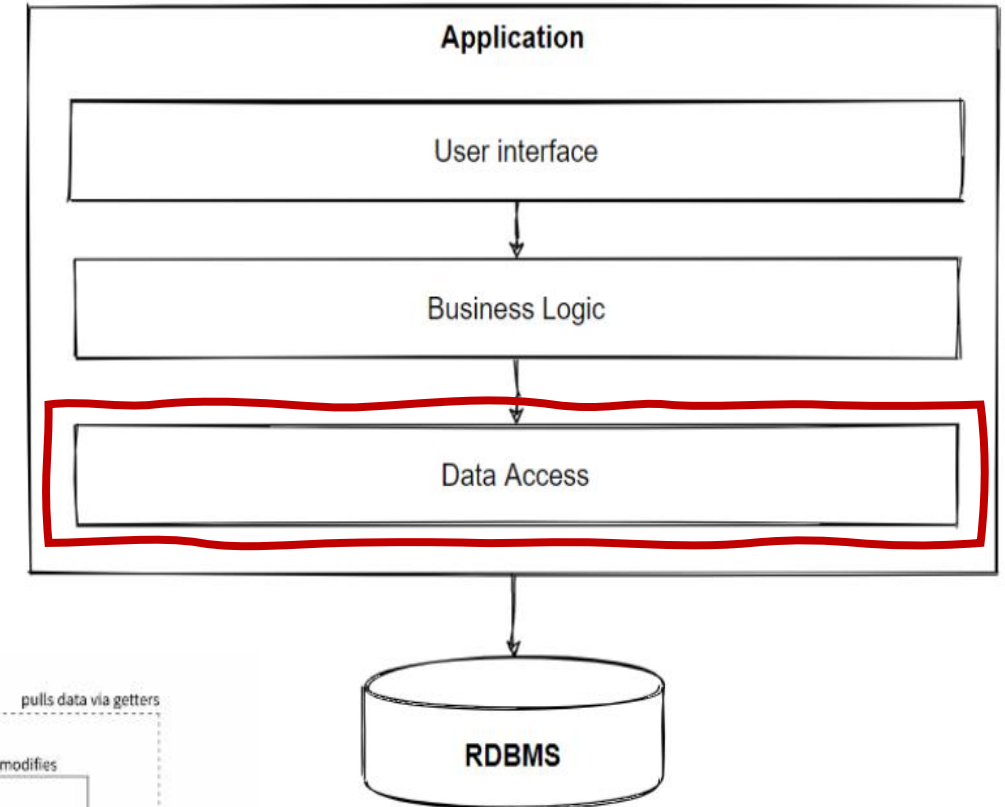
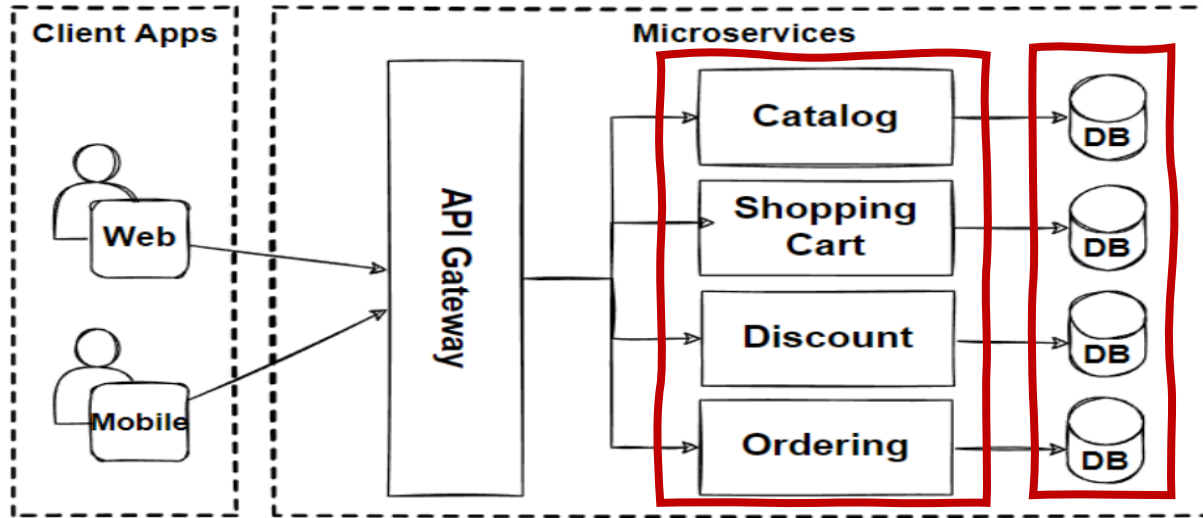
Eng. Raghad al-hossny

Design patterns:

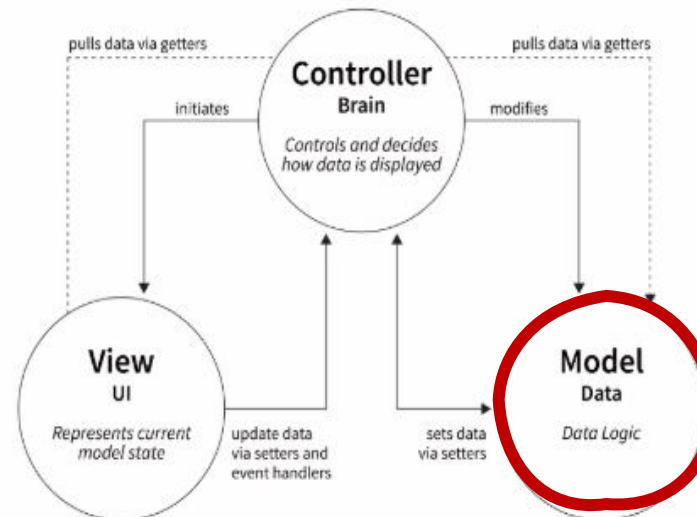


DAL design patterns

Data access (component):



In this course, we will focus on the detailed design of the DAL .

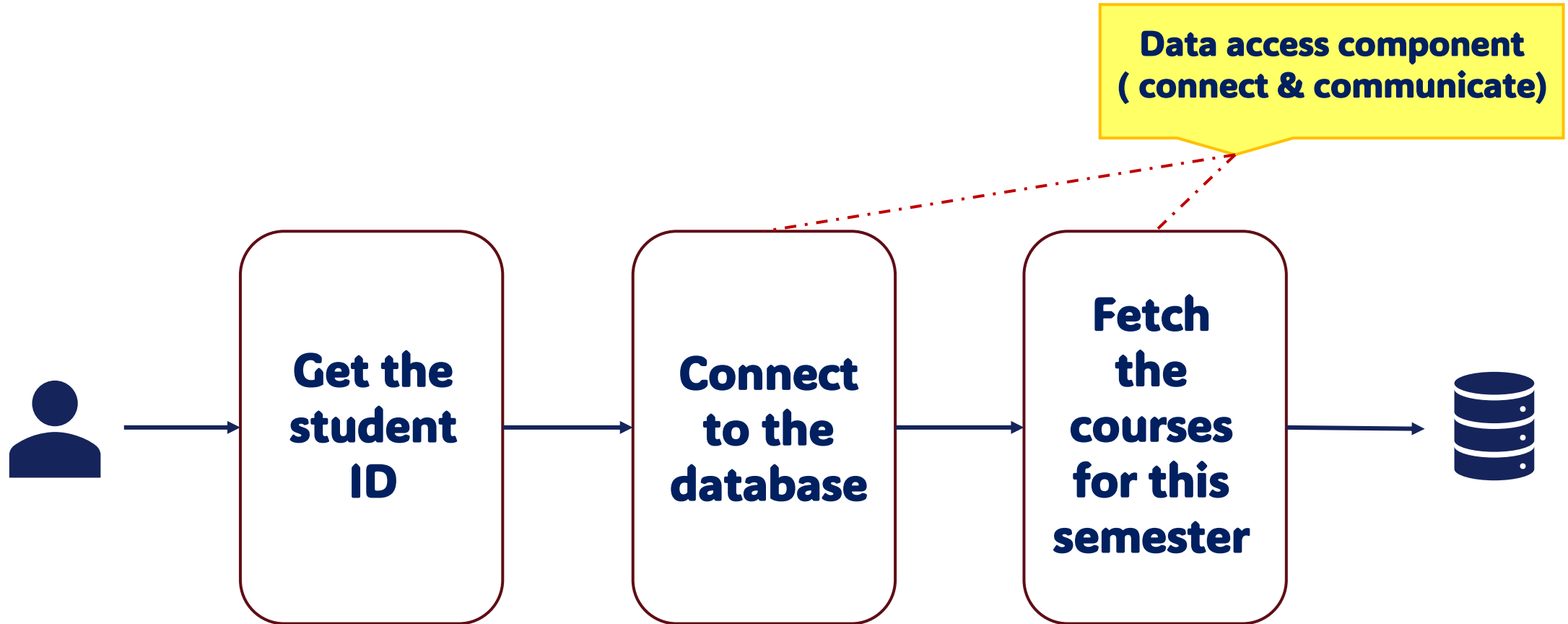


Problem – Student Courses :

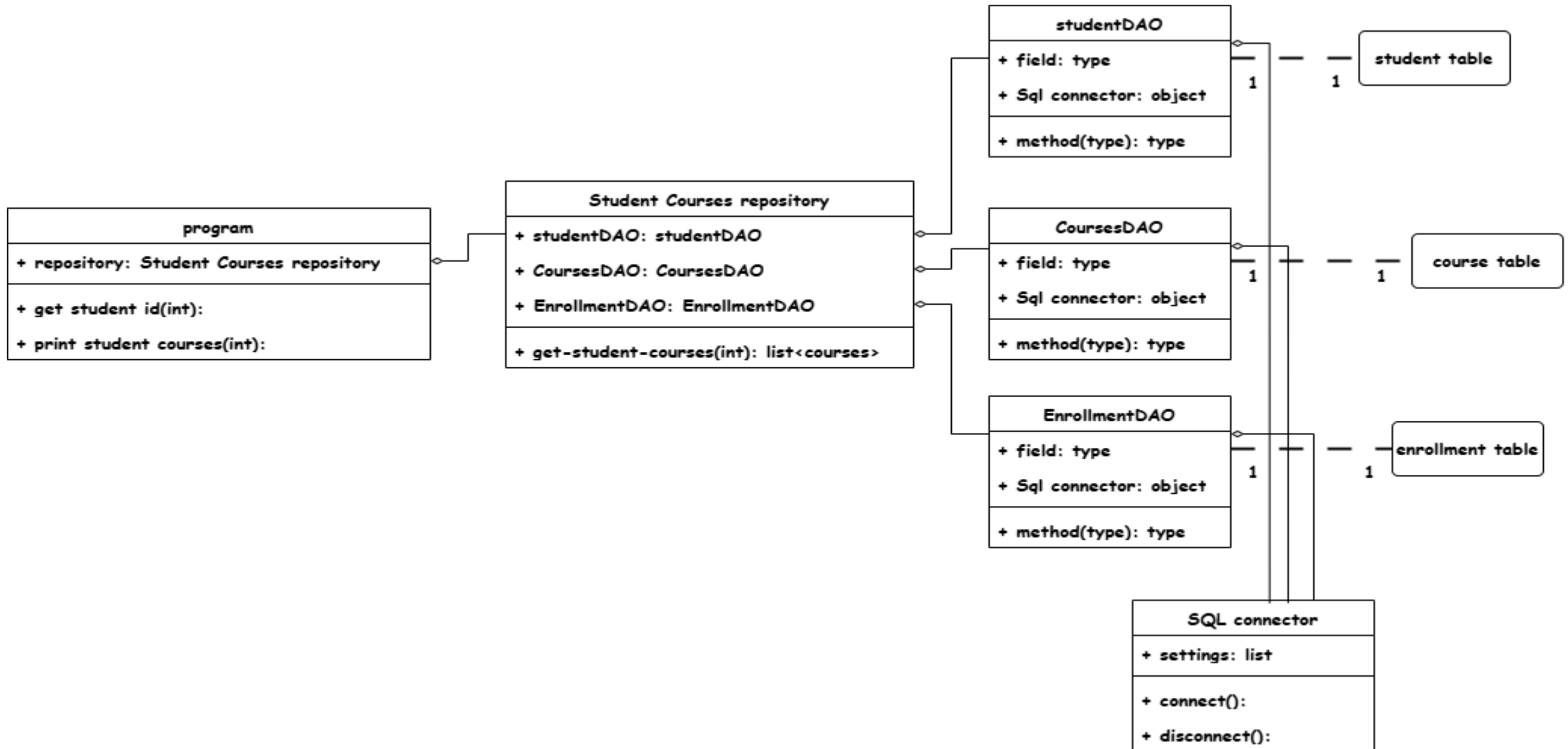
“Imagine we are a team of software developers tasked with designing and building a **lightweight console program** that allows a **user** (e.g., an admin or staff member) to **enter a student number**, and in return, it **fetches and displays the current enrolled courses** for that student by querying a **relational database** (e.g., PostgreSQL, MySQL, or SQL Server).”



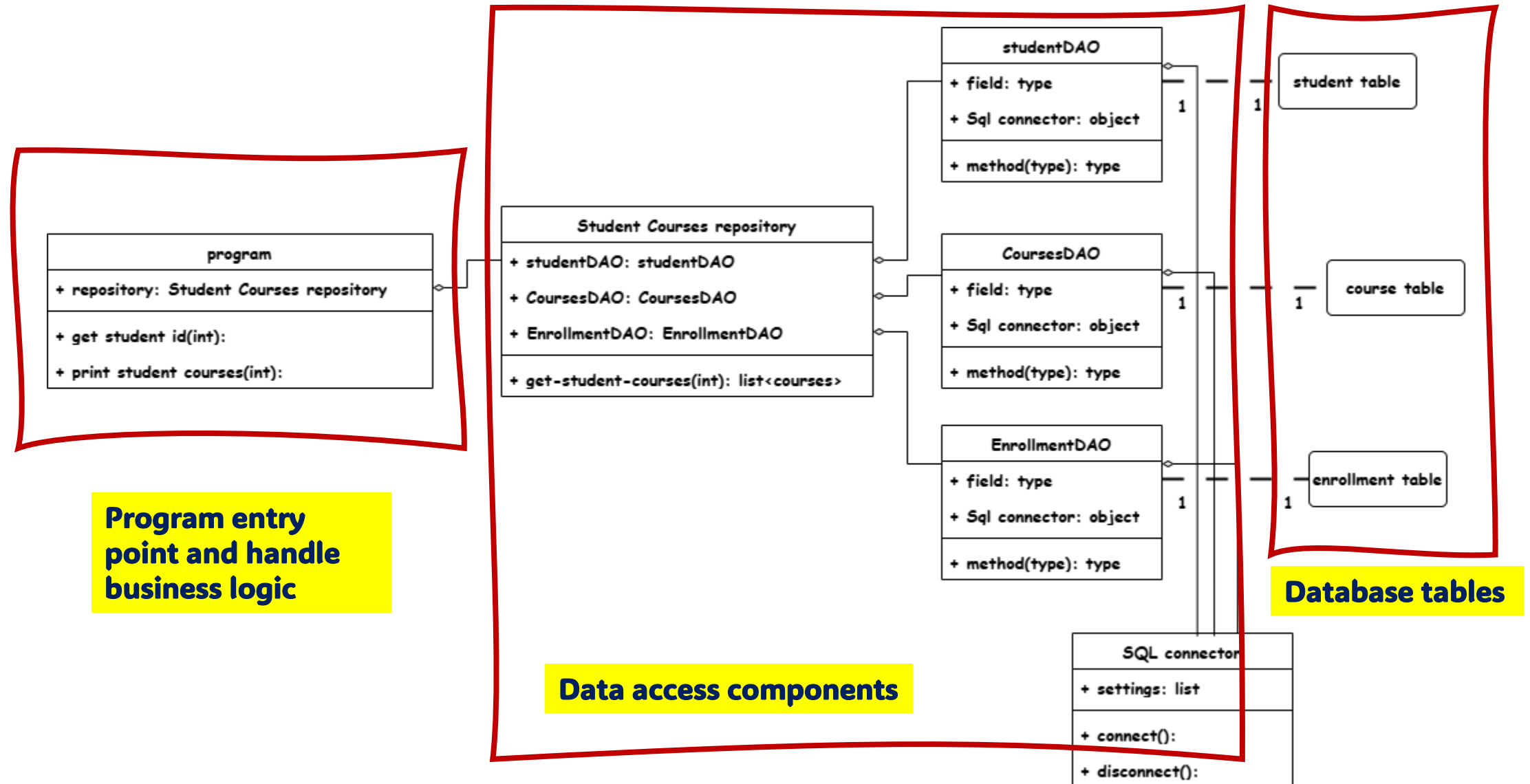
1- Understand the problem:



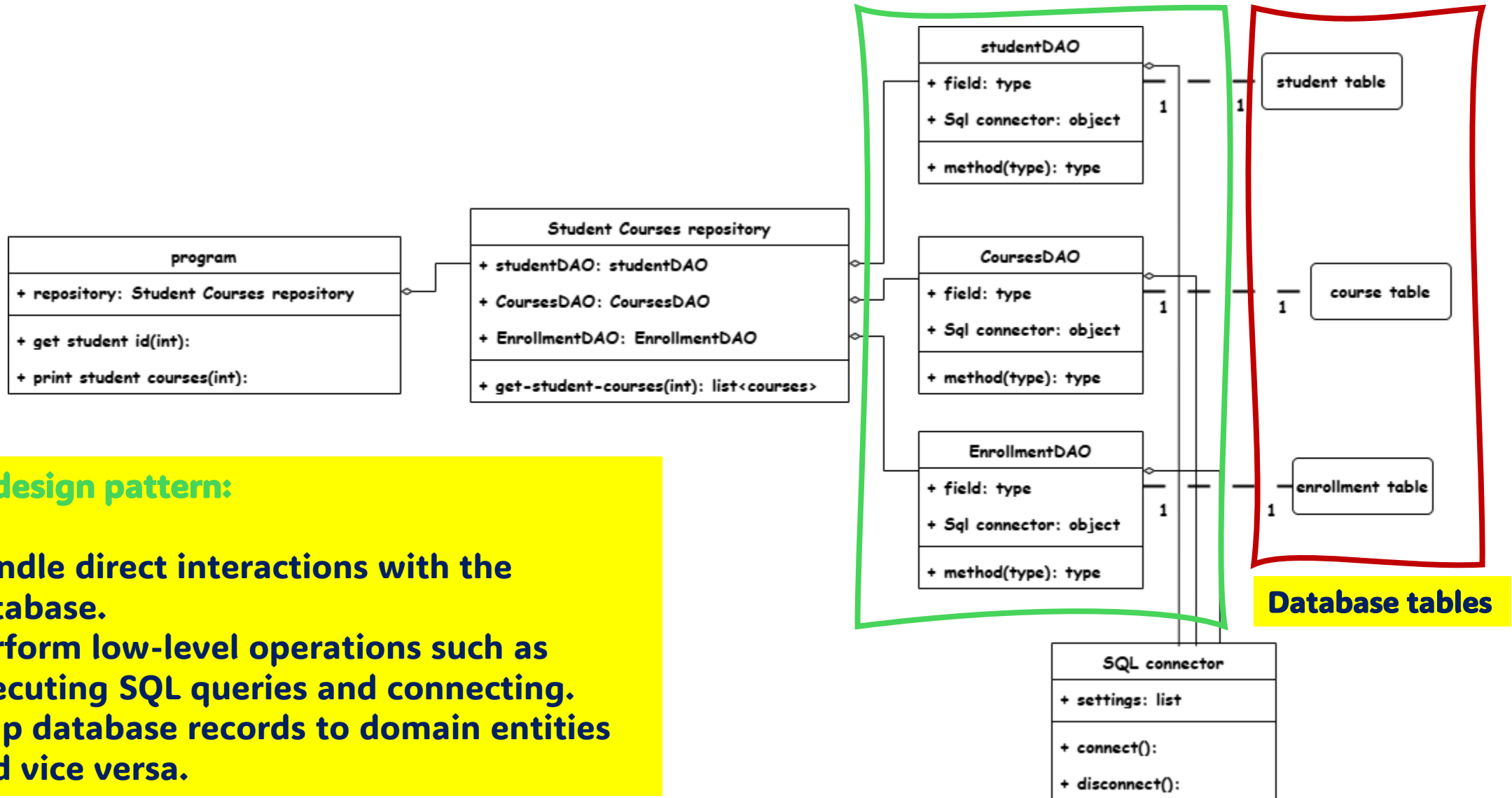
2- design the solution – class diagram:



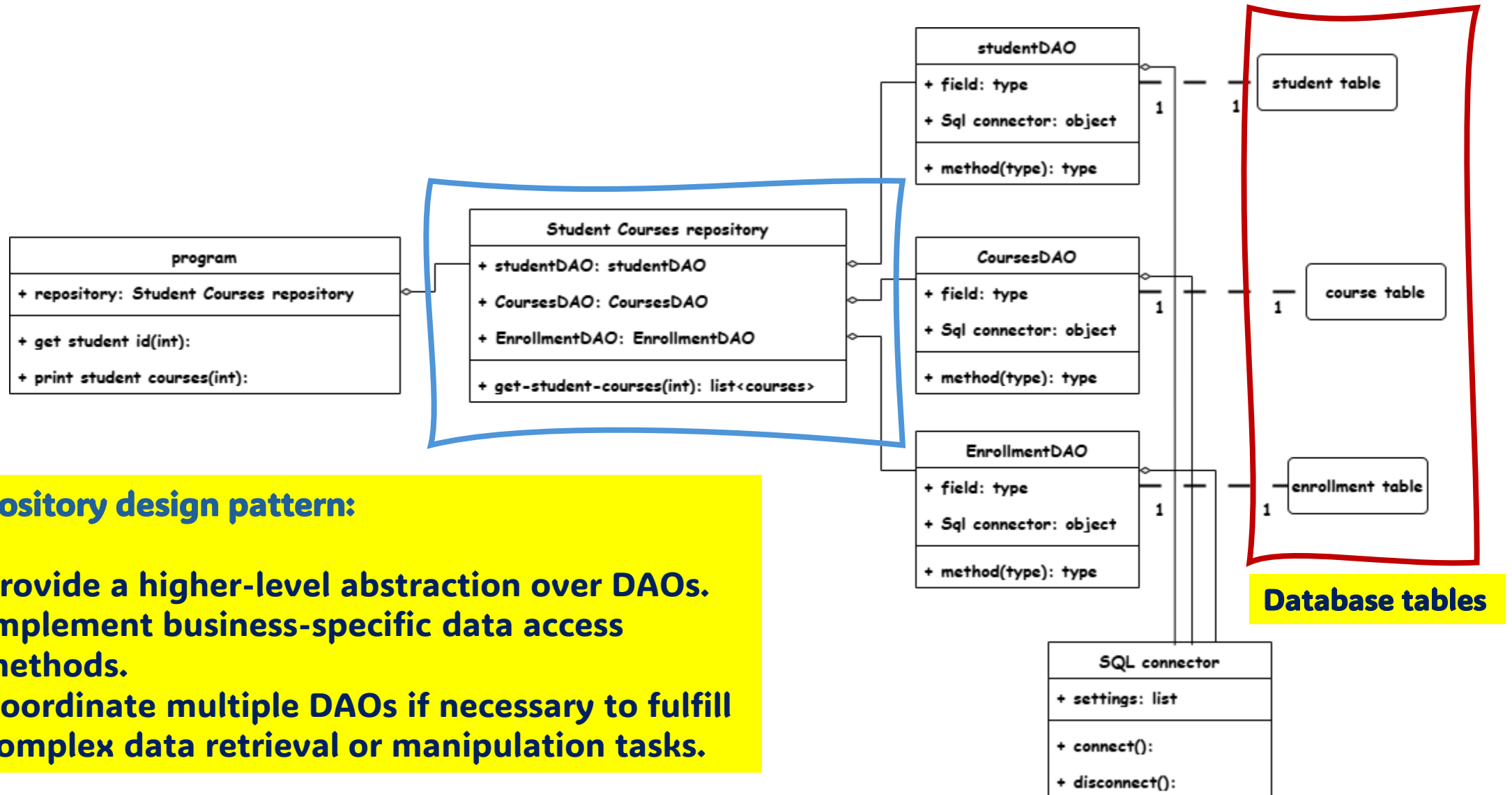
2- design the solution – class diagram:



2- design the solution – class diagram:



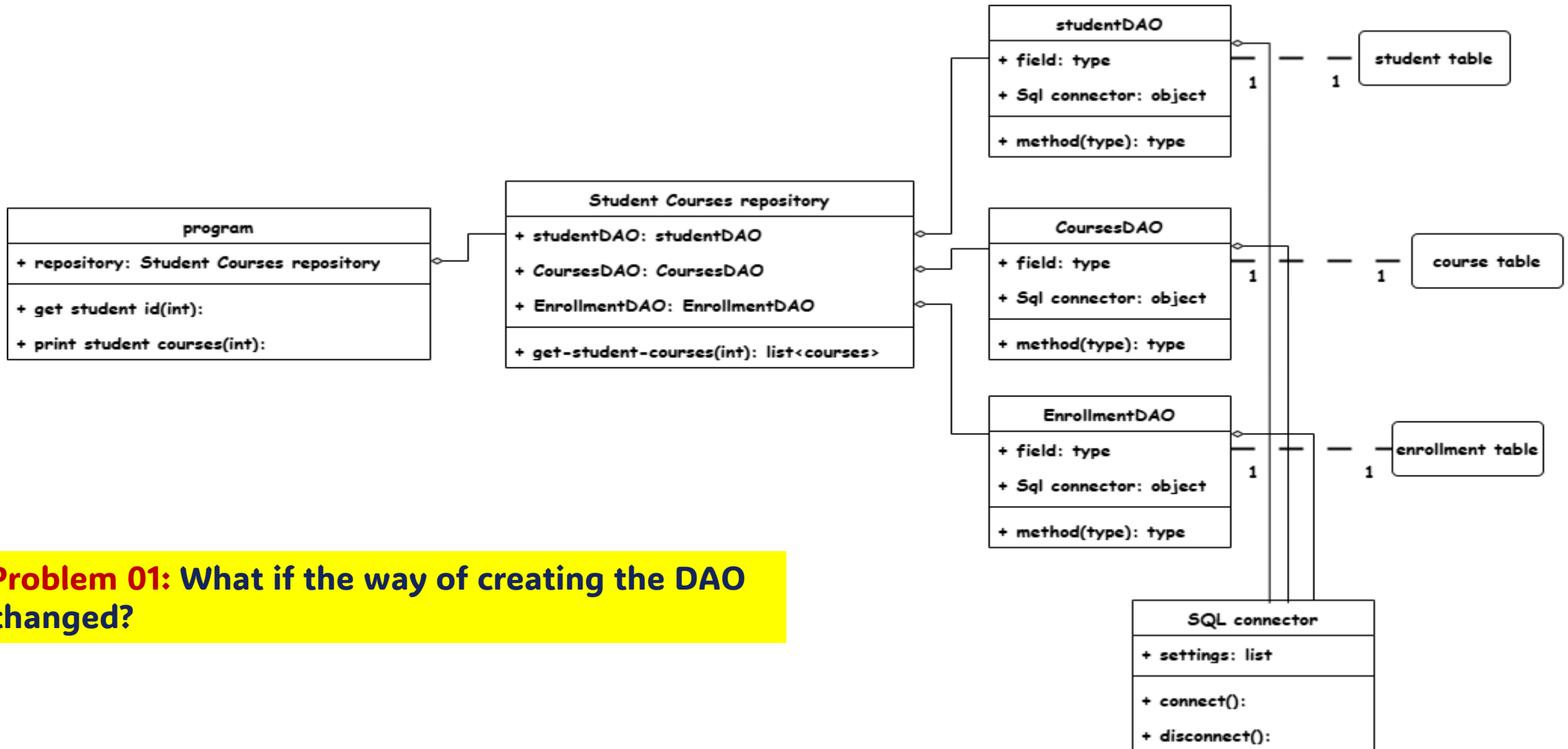
2- design the solution – class diagram:



Repository design pattern:

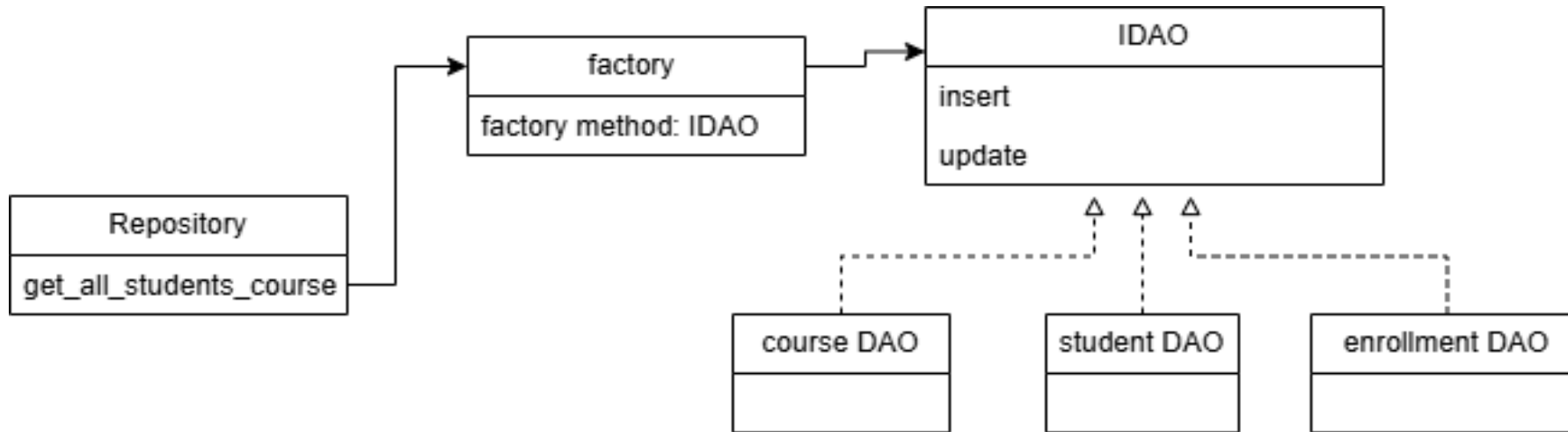
- Provide a higher-level abstraction over DAOs.
- Implement business-specific data access methods.
- Coordinate multiple DAOs if necessary to fulfill complex data retrieval or manipulation tasks.

How could we enhance this design?



Problem 01: What if the way of creating the DAO changed?

Using a factory method:



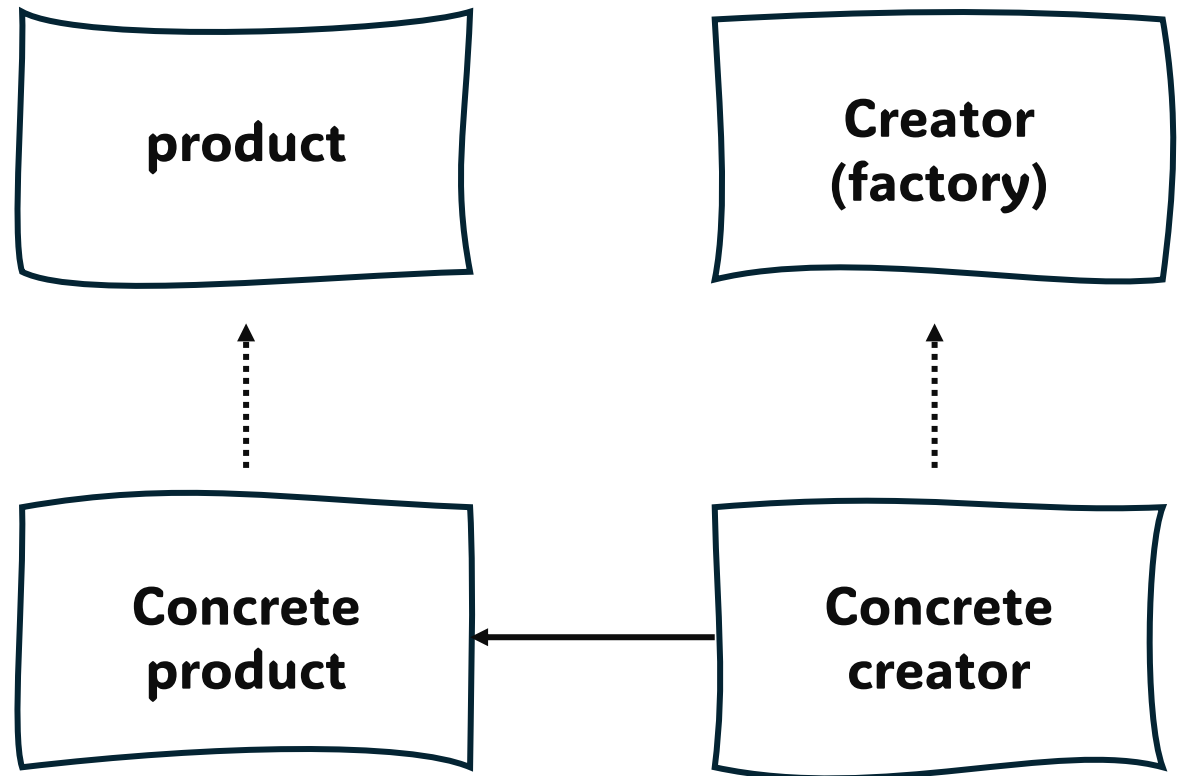
The Factory Method pattern suggests that you replace direct object construction calls (using the `new()` operator) with calls to a **special factory method**.

the objects are still created via the `new` operator, but it's being called from within the factory method.

Objects returned by a factory method are often referred to as **products**.

Using a factory method:

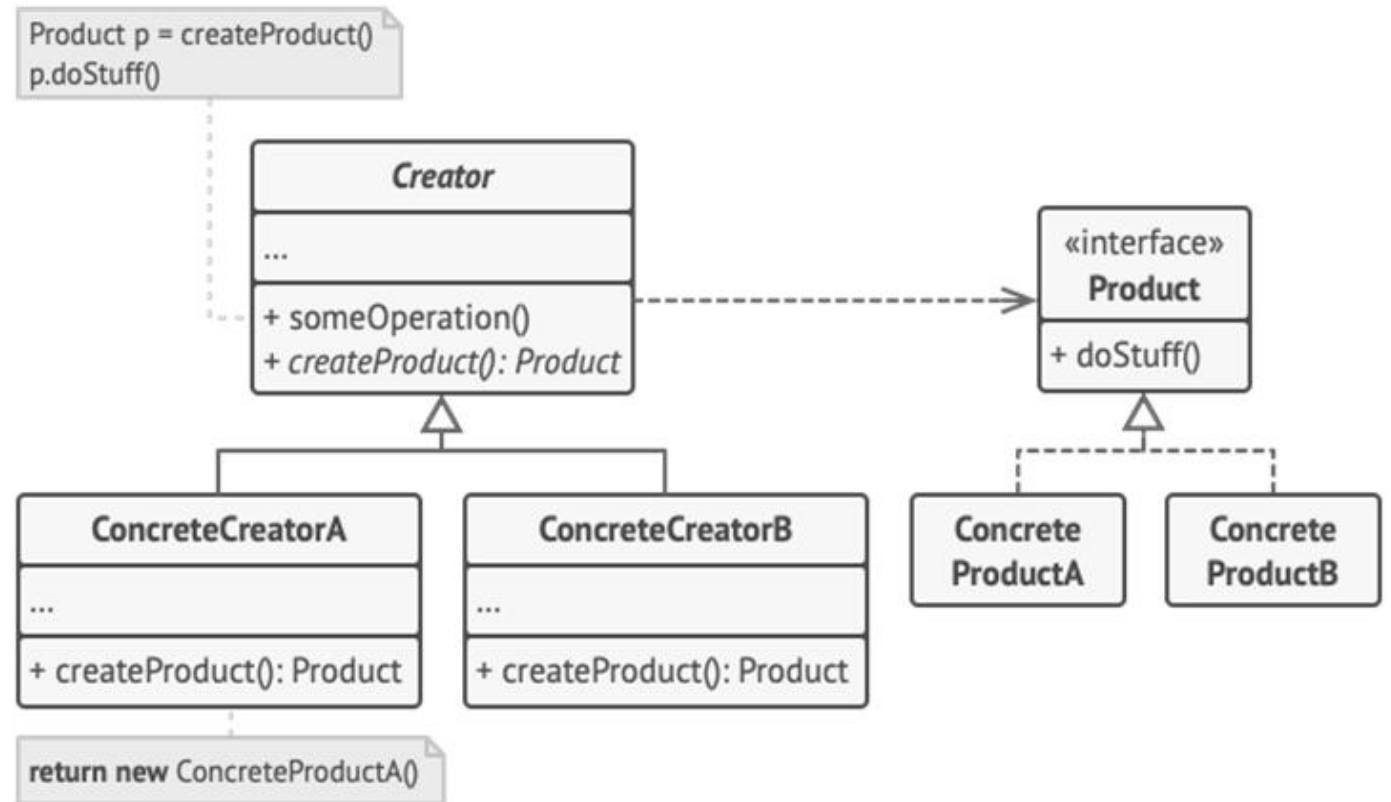
The code that uses the factory method (the client code) doesn't see a difference between the actual products returned by various subclasses. The client treats all the products as abstract Transport. The client knows that all transport objects are supposed to have the **deliver** method, but exactly how it works isn't important to the client.



Using a factory method:

Factory Method is a creational design pattern that provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created.

It is best practice to use the factory method design pattern in conjunction with the strategy design pattern.



Workshop

Problem – orders and products :

“Imagine we are a team of software developers tasked with designing and building a lightweight console program that allows a user (e.g., a salesperson) to enter an order ID, and in return, it fetches and displays the list of products included in that order, along with their quantities and total price, by querying a relational database (e.g., PostgreSQL, MySQL, or SQL Server).”

- 1. Design the database.**
- 2. Design the DAL.**
- 3. Build the database.**
- 4. Implement the DAL in Java.**

