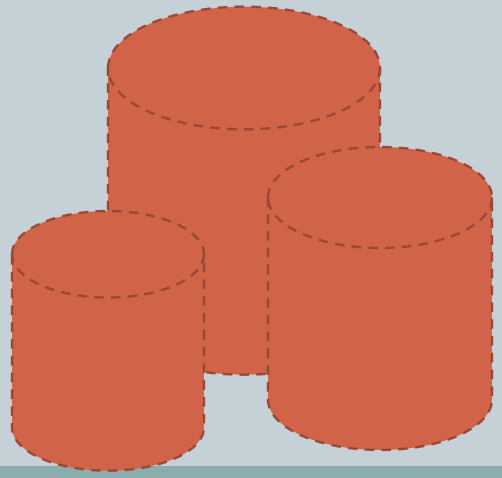
Data Access Patterns



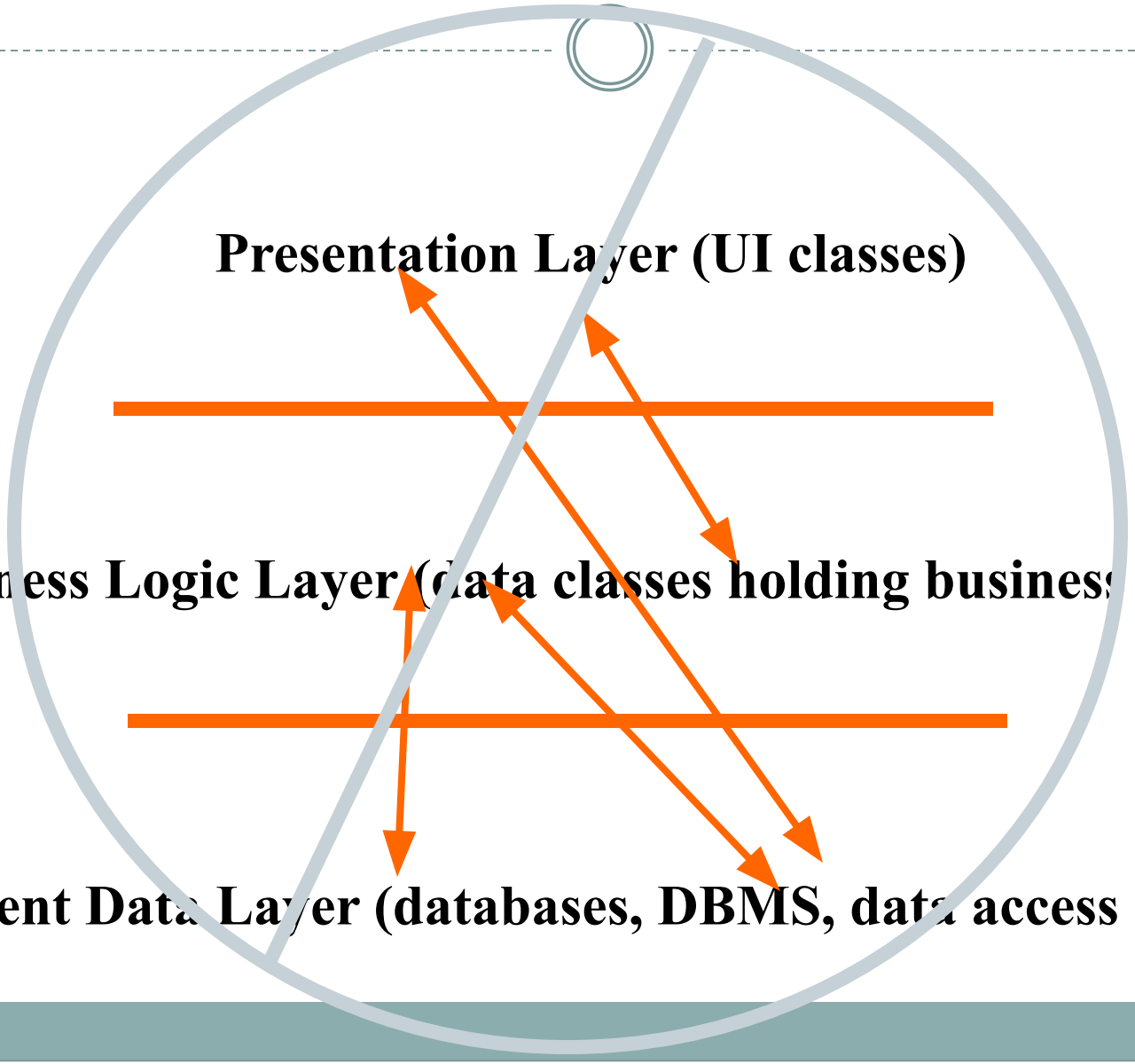
Three Tier Architecture

Presentation Layer (UI classes)

Business Logic Layer (data classes holding business logic)

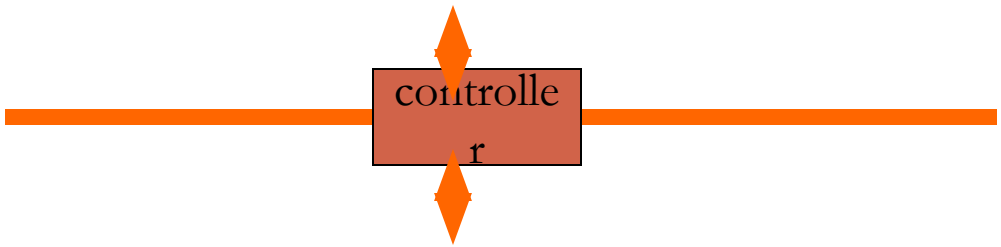
Persistent Data Layer (databases, DBMS, data access classes)

Three Tier Architecture

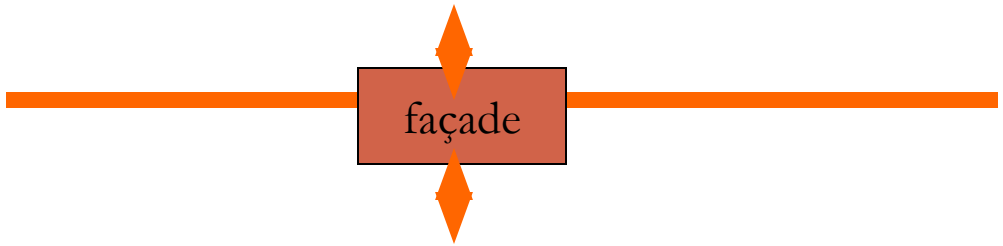


Three Tier Architecture

Presentation Layer (UI classes)



Business Logic Layer (data classes holding business logic)



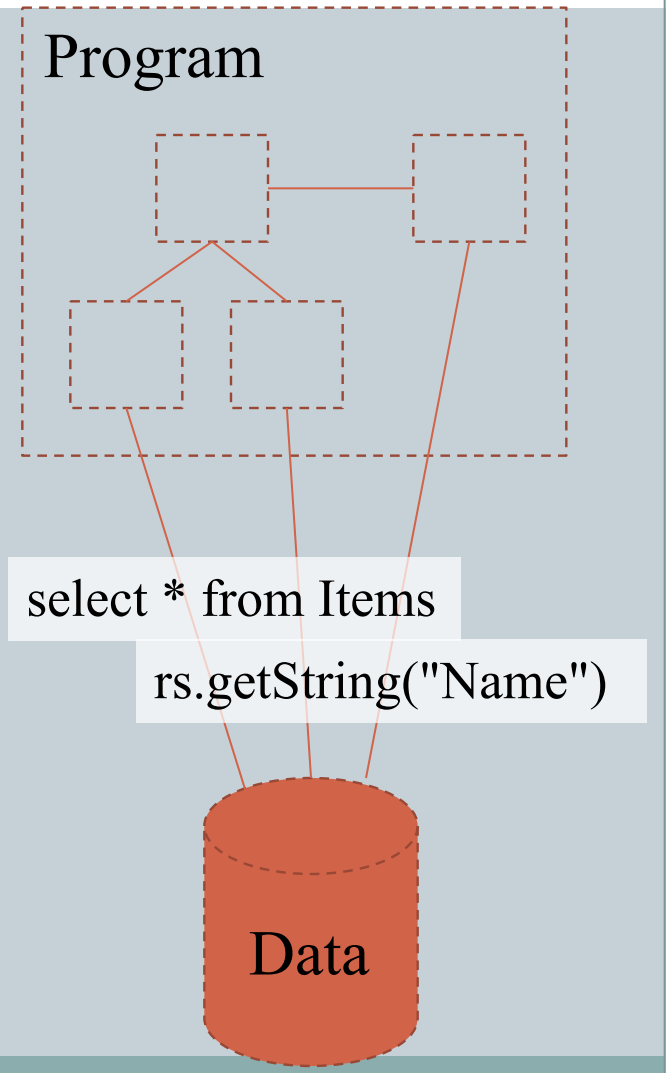
Persistent Data Layer (databases, DBMS, data access classes)

Facade

Шаблон фасад (англ. Facade)
— структурный шаблон проектирования, позволяющий скрыть сложность системы путём сведения всех возможных внешних вызовов к одному объекту, делегирующему их соответствующим объектам системы.

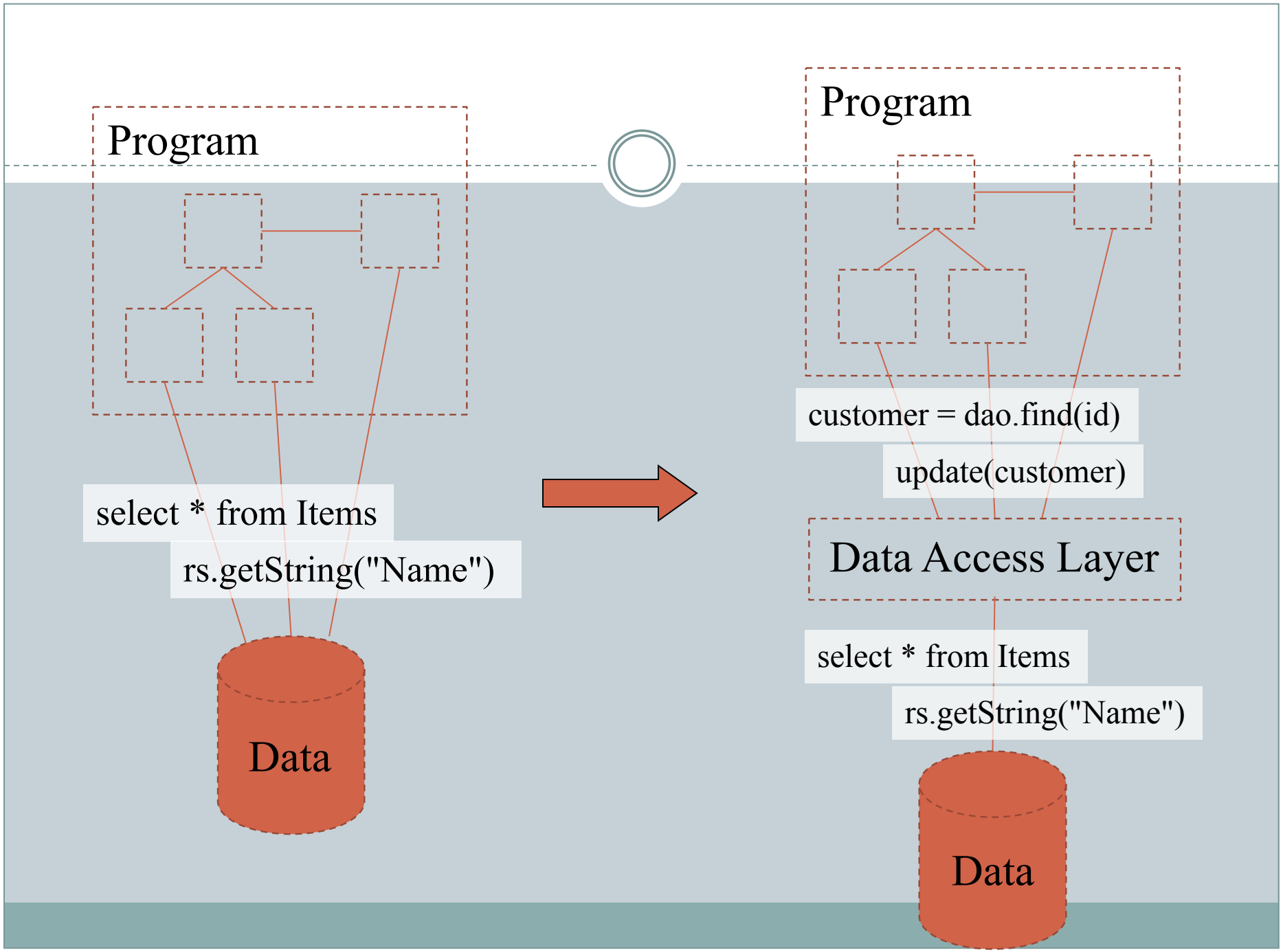
Motivation

- Most software systems require persistent data (i.e. data that persists between program executions).
- In general, distributing low-level data access logic throughout a program is <u>not</u> a good idea (design).



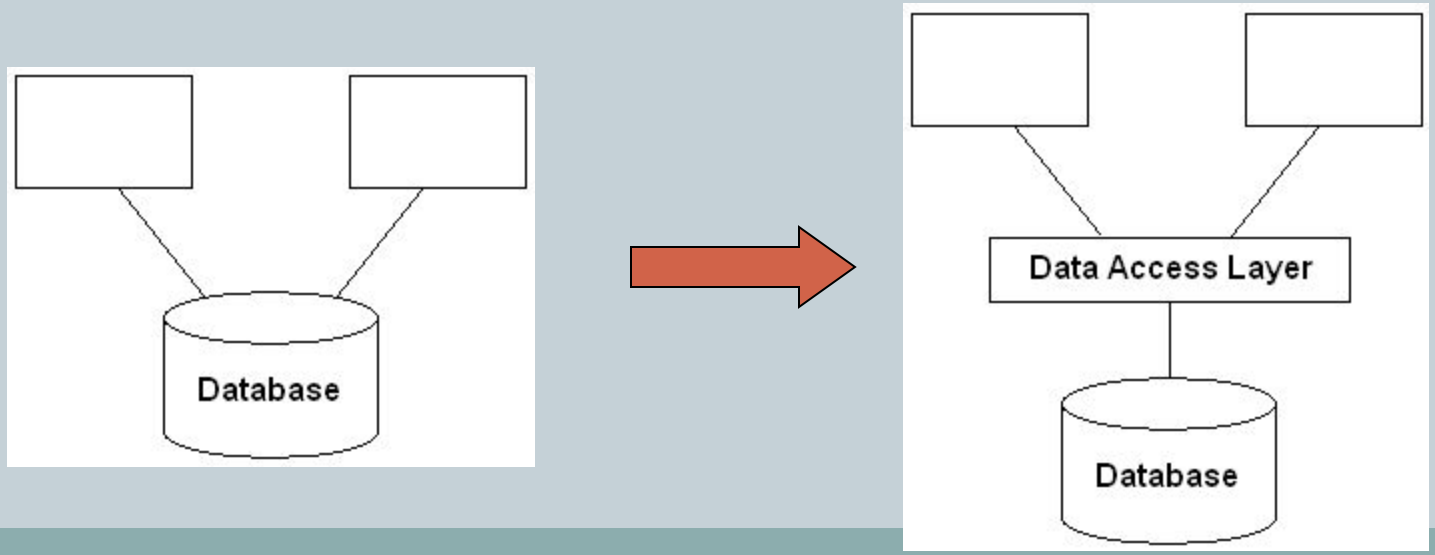
Data Access Layer

- A better design is one that includes a data access layer which encapsulates the details of the underlying persistence API.
- It abstracts the low-level details of persistent storage.
- It provides an interface that is usually a better match for the style of programming used in the domain logic. For example, the data access layer might provide an OO interface onto relational data.

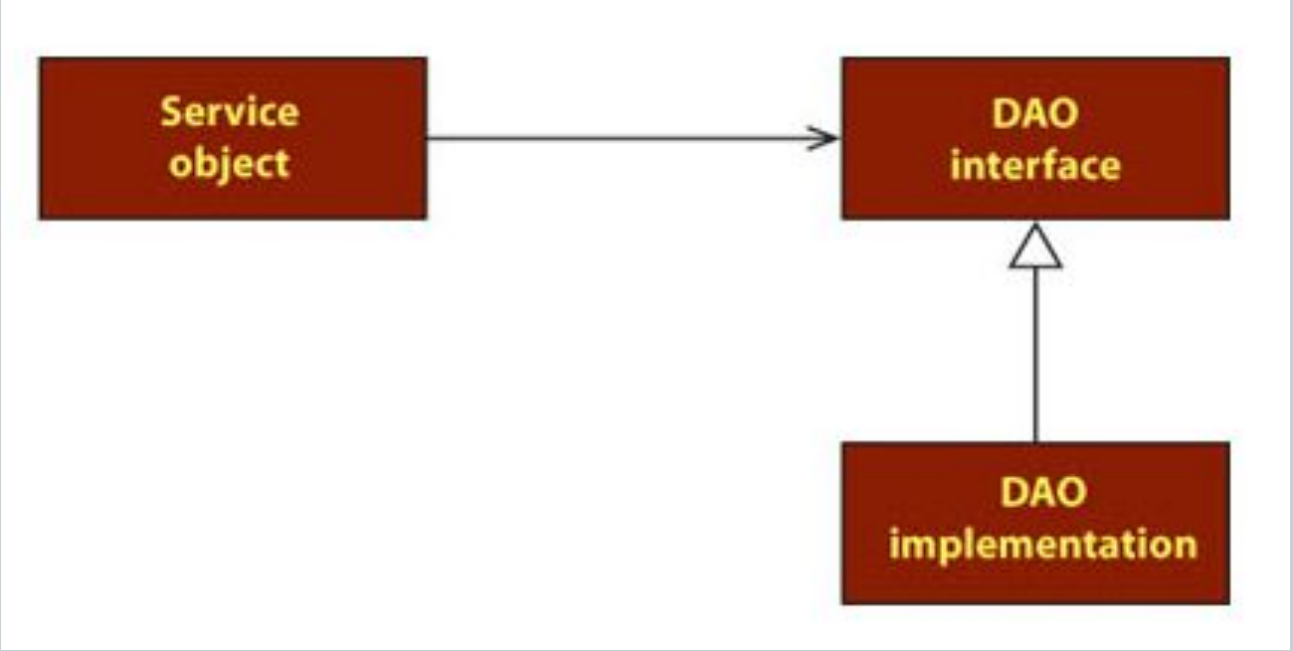

Program
select * from Items
rs.getString("Name")
Data
Program
customer = dao.find(id)
update(customer)
Data Access Layer
select * from Items
rs.getString("Name")
Data

Data Access Layer

- To avoid problems associated with mixing SQL and application logic, SQL statements and data base design details are often encapsulated in a data access layer
- The data access layer presents an interface that is convenient for application programs.



Program to an Interface; Not Implementation



Three Tier Architecture