Layered Architecture

Learning Objectives

At the end of this lesson, you will be able to:

- Business Delegate Pattern
- Data Access Object Pattern
- Transfer Object Pattern
- Iterator Pattern

What are Design Pattern?

Design patterns are programming constructs used by object-oriented programmers. They are programming language-independent strategies for solving common object-oriented design problems

Design patterns are well-proven solutions for solving specific problems and hence help improve the flexibility, reusability, and maintainability of our code. They provide a common language and shared understanding among developers, making it easier to communicate and collaborate on software projects.

History?

In 1994, four authors Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides wrote a book called “Design Patterns – Elements of Reusable Object-Oriented Software”. Gang of Four (GoF) has divided the book into two parts with the first part explaining the Pros and Cons of Object-Oriented Programming and the second part describing the Evolution Of 23 Classic Software Design Patterns.
.

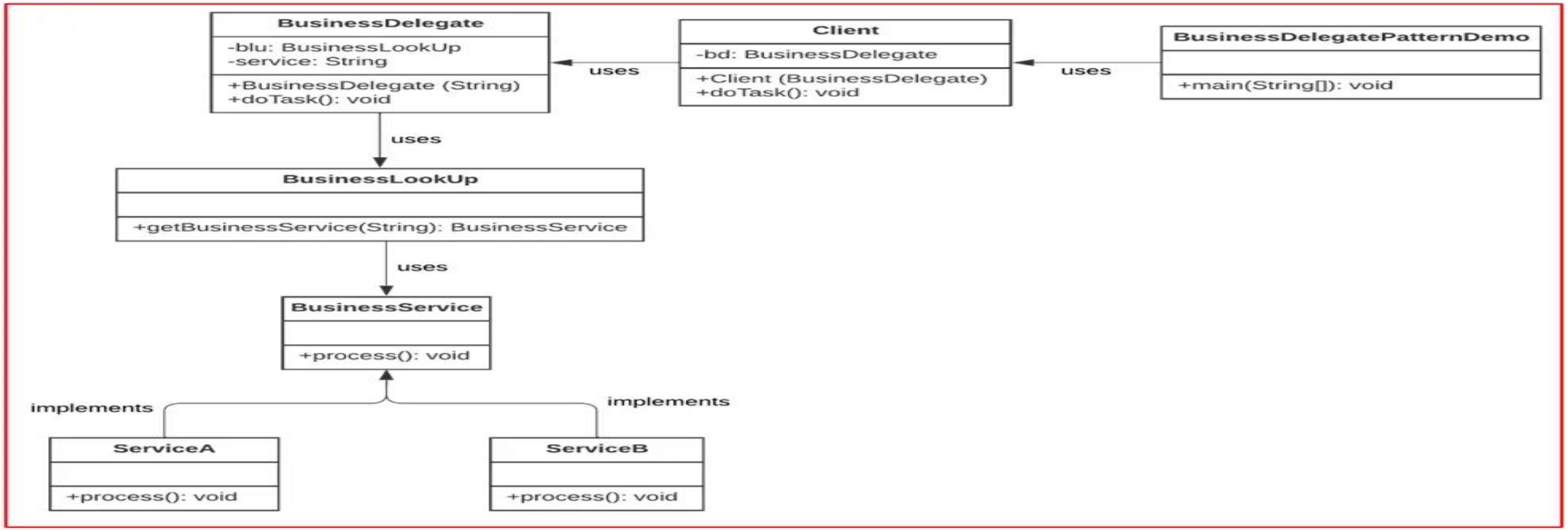
Types of Design Patterns?

- **Creational Design Patterns**: These provide a way to easily create objects while hiding the creation logic. This gives the program more flexibility in deciding which objects need to be created for a given use case.
- **Structural Design Patterns**: These work with class and object composition and use inheritance to write interfaces and obtain new functionality.
- **Behavioral Design Patterns**: These are concerned with communication between objects.

Business Delegate Pattern

The Business Delegate design pattern offers a solution to abstract away the complexities of interacting with multiple services, providing a centralized point of access.

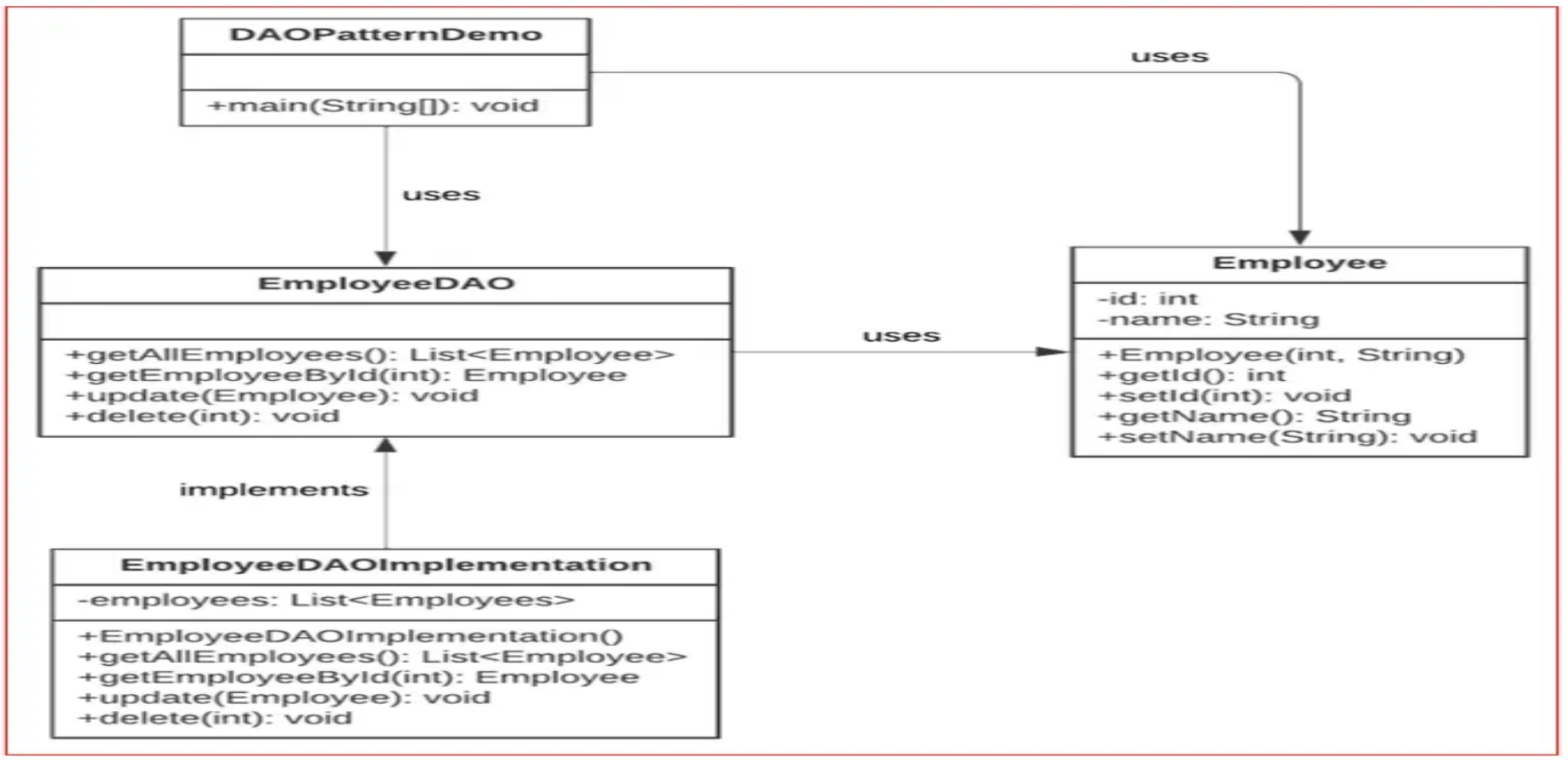
The Business Delegate design pattern separates the client's presentation logic from the complexities of accessing and interacting with multiple business services. It encapsulates the process of communication, error handling, and protocol-specific details, allowing the client to focus on the core business logic.



Data Access Object Pattern

Data Access Object (DAO) design pattern provides a structured approach to handling data access, abstracting away the underlying database details.

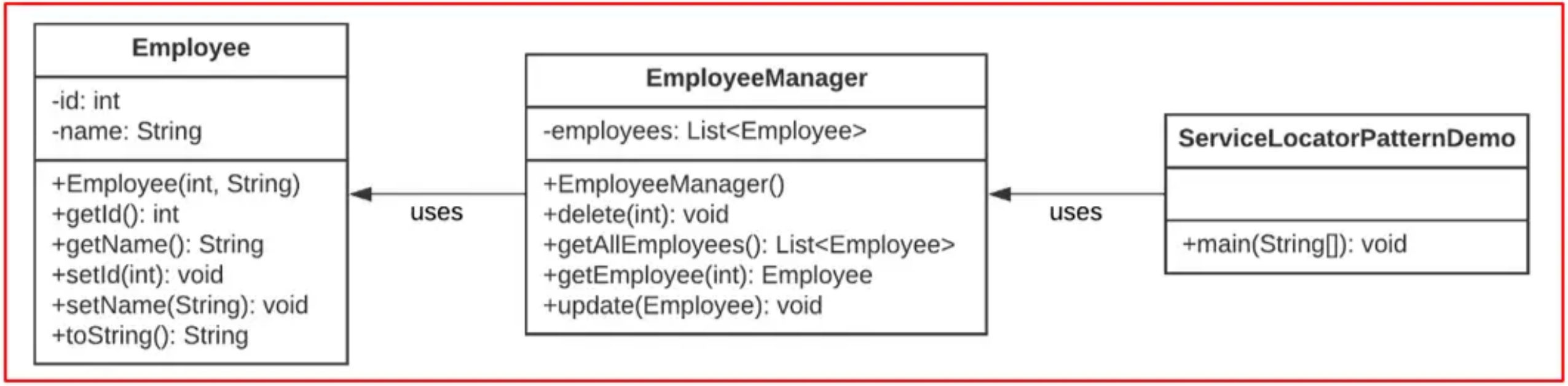
The DAO design pattern separates the business logic of an application from the data access layer, providing a consistent and abstract interface for interacting with a data source, such as a database. It encapsulates the data access operations, hiding the complexities of database connections, SQL queries, and result handling.



Transfer Object Pattern

Transfer Object Design Pattern provides an efficient and flexible way to transfer data between components or layers of an application. It encapsulates a set of related data fields into a single object, which can be easily serialized, transmitted, and reconstructed at the receiving end.

The Transfer Object design pattern aims to improve the performance and maintainability of data transfer operations by encapsulating related data fields into a single object. It is often used in scenarios where data needs to be transferred between remote components or layers of an application.`



Iterator Pattern

In software development, efficiently traversing and accessing the elements of a collection is a common requirement. The Iterator design pattern provides an elegant solution to this challenge by separating the traversal logic from the collection itself. By encapsulating iteration within a separate iterator object, the pattern enables uniform access to elements while maintaining encapsulation and abstraction.