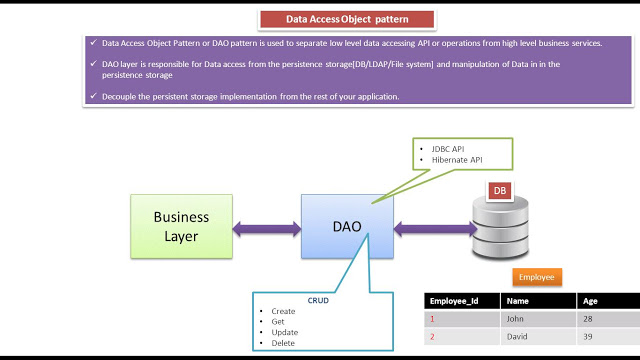
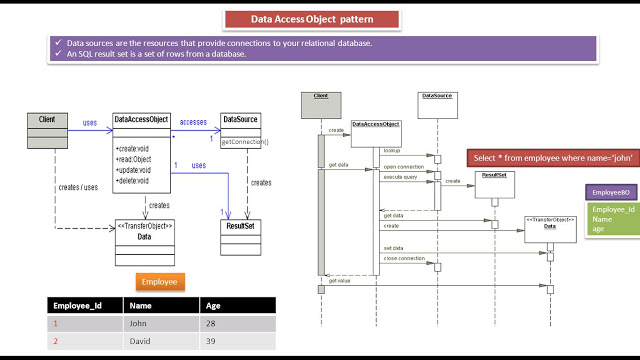
# DATA ACCESS OBJECTS, JAVA BEANS AND POJO CLASSES

# DATA ACCESS OBJECTS

In computer software, a **data access object** (**DAO**) is a pattern that provides an abstract [interface](https://en.wikipedia.org/wiki/Interface_(computer_science)) to some type of [database](https://en.wikipedia.org/wiki/Database) or other persistence mechanism. By mapping application calls to the persistence layer, the DAO provides some specific data operations without exposing details of the database. This isolation supports the [single responsibility principle](https://en.wikipedia.org/wiki/Single_responsibility_principle). It separates what data access the application needs, in terms of domain-specific objects and data types (the public interface of the DAO), from how these needs can be satisfied with a specific [DBMS](https://en.wikipedia.org/wiki/Database_Management_System), database schema, etc. (the implementation of the DAO).



The primary advantage of using data access objects is the relatively simple and rigorous separation between two important parts of an application that can but should not know anything of each other, and which can be expected to evolve frequently and independently. Changing business logic can rely on the same DAO interface, while changes to persistence logic do not affect DAO clients as long as the interface remains correctly implemented.

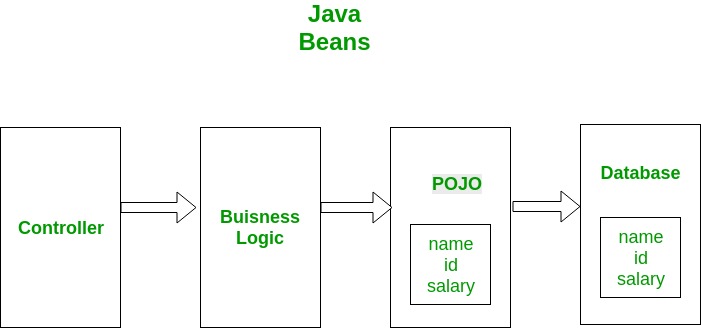


# JAVA BEANS

A JavaBean is a Java class that should follow the following conventions:

* It should have a no-arg constructor.
* It should be Serializable.
* It should provide methods to set and get the values of the properties, known as getter and setter methods.

A bean encapsulates many objects into one object so that we can access this object from multiple places. Moreover, it provides easy maintenance.  It is a reusable software component.



# POJO CLASSES

POJO stands for Plain Old Java Object. It is an ordinary Java object, not bound by any special restriction other than those forced by the Java Language Specification and not requiring any classpath. POJOs are used for increasing the readability and re-usability of a program.

**A POJO should not:**

1. Extend prespecified classes, Ex: public class GFG extends javax.servlet.http.HttpServlet { … } is **not** a POJO class.
2. Implement prespecified interfaces, Ex: public class Bar implements javax.ejb.EntityBean { … } is **not** a POJO class.
3. Contain prespecified annotations, Ex: @javax.persistence.Entity public class Baz { … } is **not** a POJO class.

# POJO V/S BEANS

**Beans are special type of Pojos. There are some restrictions on POJO to be a bean.**

1. All JavaBeans are POJOs but not all POJOs are JavaBeans.
2. Serializable i.e. they should implement Serializable interface. Still, some POJOs who don’t implement Serializable interface are called POJOs because Serializable is a marker interface and therefore not of much burden.
3. Fields should be private. This is to provide the complete control on fields.
4. Fields should have getters or setters or both.
5. A no-arg constructor should be there in a bean.
6. Fields are accessed only by constructor or getter setters.