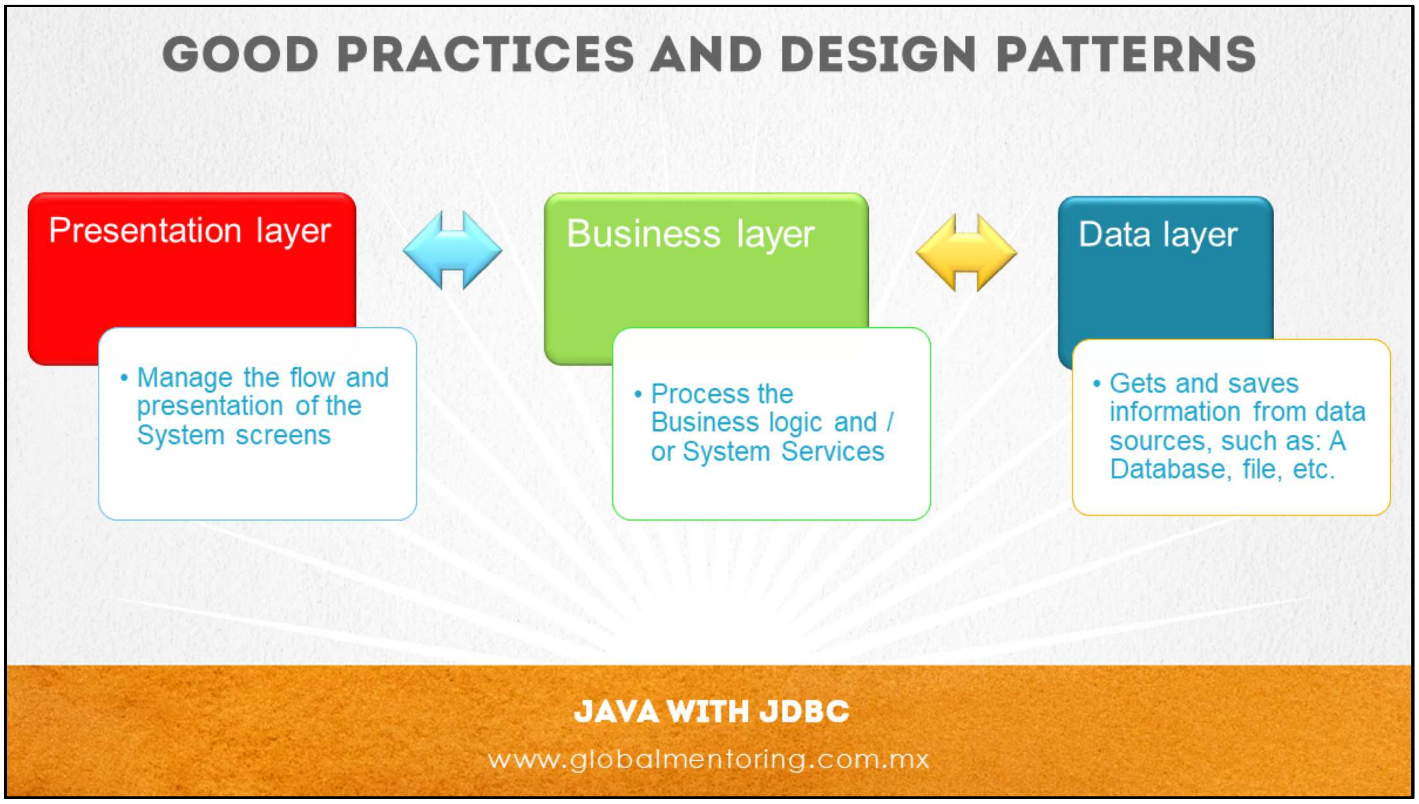
JAVA WITH JDBC
DATA LAYER WITH JDBC
By the expert: Ubaldo Acosta
Eng. Ubaldo Acosta
Global Mentoring
Experiencia y Conocimiento para tu vida
JAVA UNIVERSITY
JAVA WITH JDBC
www.globalmentoring.com.mx

Hello, Ubaldo Acosta greets you. Welcome again. I hope you're ready to start with this lesson.

We are going to study the topic of how to create a data access layer using the JDBC API.

Are you ready? OK let's go!



GOOD PRACTICES AND DESIGN PATTERNS

Presentation layer
• Manage the flow and presentation of the System screens
Business layer
• Process the Business logic and / or System Services
Data layer
• Gets and saves information from data sources, such as: A Database, file, etc.
JAVA WITH JDBC
www.globalmentoring.com.mx

We are going to review below the topic of good practices and the concept of design patterns.

In a Java Enterprise architecture it is common for the application to be divided into several layers and each one is responsible for performing specific tasks. Due to the complexity of the systems and the problems we face every day there is the concept of good practices and the concept of design patterns with the aim of reducing the mentioned problems.

Good practices include coding issues, division of responsibilities into logical layers, among other topics. In turn, the design patterns, as the name says, solve a problem that occurs recurrently (pattern) in the development of systems, particularly in object-oriented systems.

A fundamental issue in the design of systems is cohesion and coupling. This issue comes into play when we handle several logical layers since each layer is responsible for certain functionality.

As we can see in the figure, we have the presentation layer, the business layer and the data layer, each of these layers has certain responsibilities.

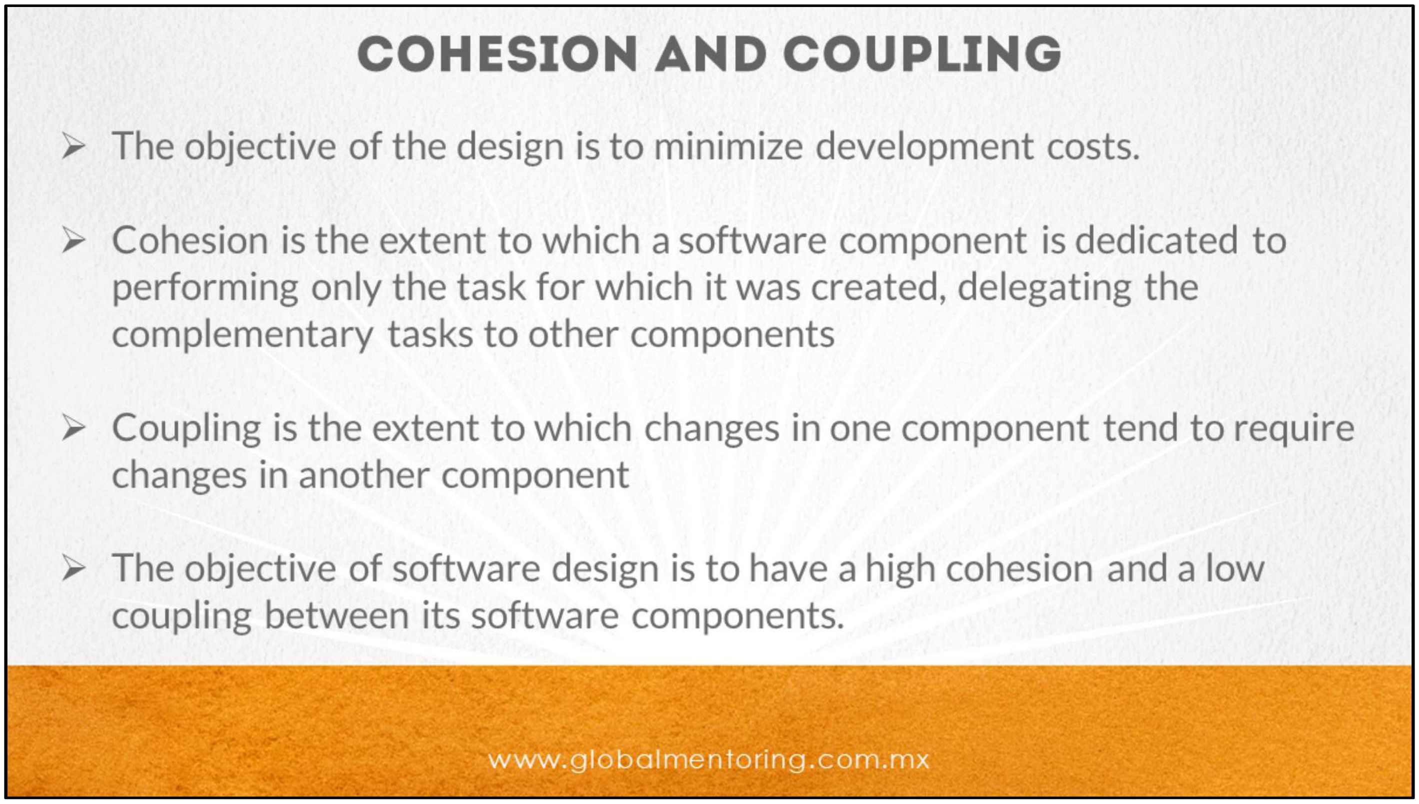
The Presentation Layer is responsible for managing the flow between the different screens of the system, as well as processing the user's data (forms) and displaying the information to it.

On the other hand, the Business Layer is responsible for processing the logic of the business and / or the services that our system must handle.

Finally, the Data Layer is responsible for communication with the database, files and other information sources.

The layers intercommunicate to process the information, that is, from a user making a request, until the system responds to the user of the new account.

Next we are going to describe some characteristics of each one of these subjects.



## COHESION AND COUPLING

- The objective of the design is to minimize development costs.
- Cohesion is the extent to which a software component is dedicated to performing only the task for which it was created, delegating the complementary tasks to other components
- Coupling is the extent to which changes in one component tend to require changes in another component
- The objective of software design is to have a high cohesion and a low coupling between its software components.

www.globalmentoring.com.mx

We will review the issue of cohesion and coupling, which play a central role in software design.

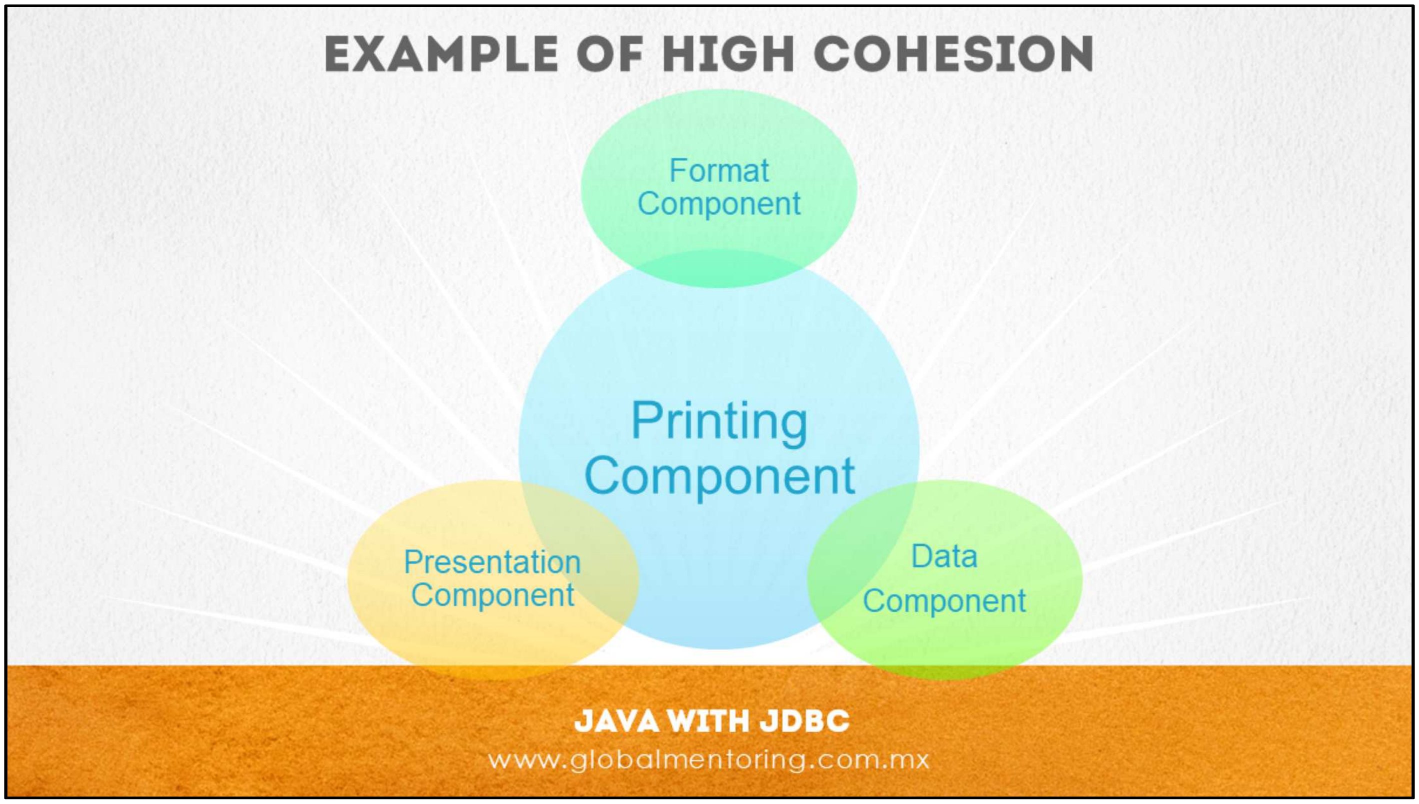
When designing our software modules we will surely require subsequent changes as the application needs change. So the design of our system can directly impact on the time and cost associated to make these changes, therefore we will review these concepts of cohesion and coupling.

We can understand cohesion as the extent to which a component is dedicated to carrying out only the task for which it was created, delegating the complementary tasks to other components.

The coupling on the other hand, measures the degree of dependence between two or more elements, these elements can be modules classes or any other software component.

The ultimate goal of software development is to create systems with low coupling and high cohesion, these are 2 characteristics that we must take into account when creating our applications.

Layered logical division in a Java Enterprise architecture (Java EE), introduces low coupling and high cohesion automatically, because it allows the number of relationships between each layer is the lowest possible and increases cohesion in each layer, because we have a division of responsibilities in a very specific and clear way.

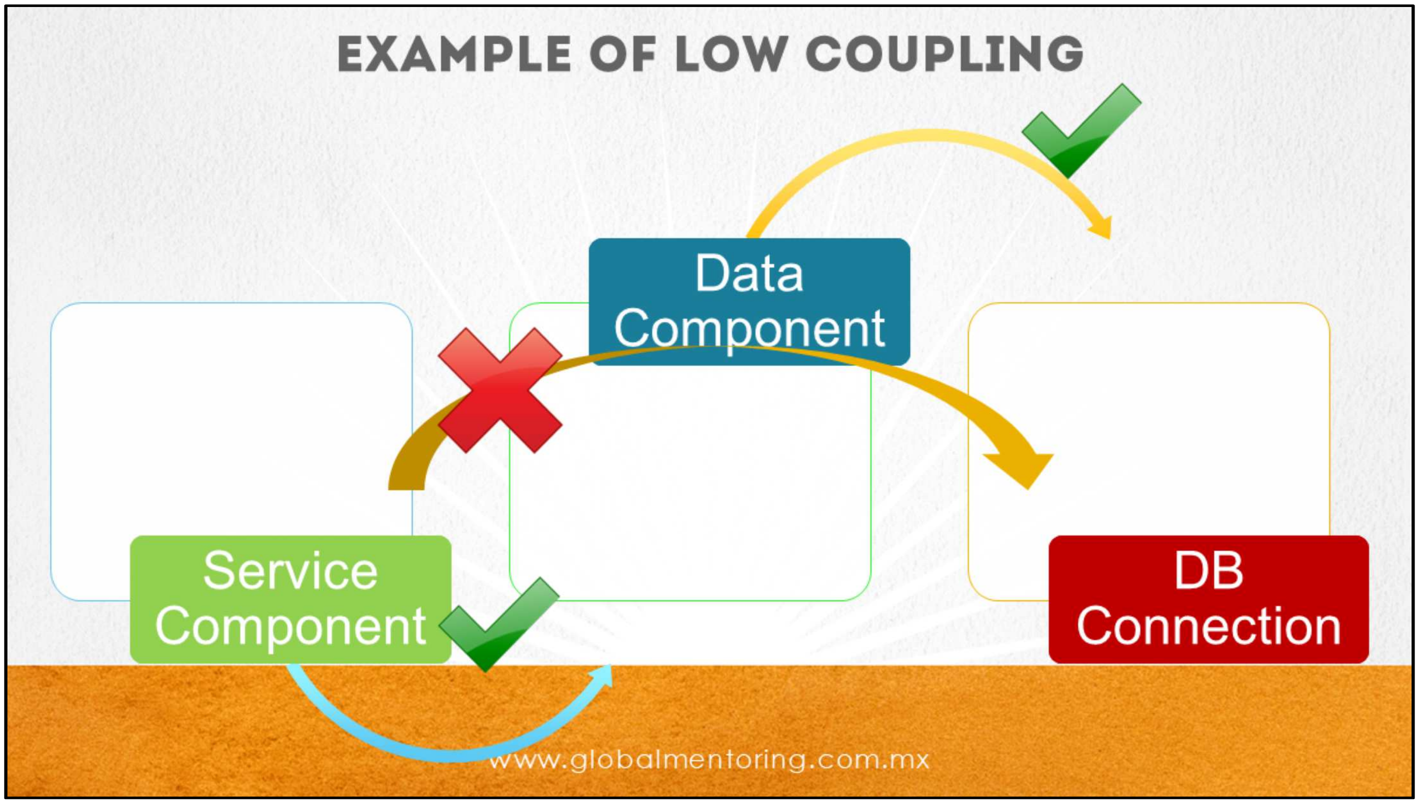


In this example, we can observe the printing component, which only has the responsibility to print certain information. To carry out the other tasks, it relies on other components, so it is said that we have a high cohesion, because each component is dedicated only to the task for which it was created.

To achieve high cohesion we will support components such as data components to obtain information (from any database or file), the component to present the information and a component to format the information selected by the user.

The printing component in the design of classes is not necessarily the most important design, since we can see that when printing we do not necessarily have to have all the tasks being executed by this component, but that we can rely on more components to finally perform the printing task. Some of these components can even be part of other applications and we can reuse code and components of other applications, as long as the components have few dependencies and the code has been designed with a high cohesion in mind.

This type of design allows changes to be easier to identify and perform, but generates smaller components, that is, with less code, which could increase the coupling and as we discussed this is not desirable. To correct this, you must manage a balance between the design of components and try not to be too small modules. We must also take care that the components are not too large, that is, that they do not have too much code, otherwise the errors will become increasingly difficult to solve and / or identify.

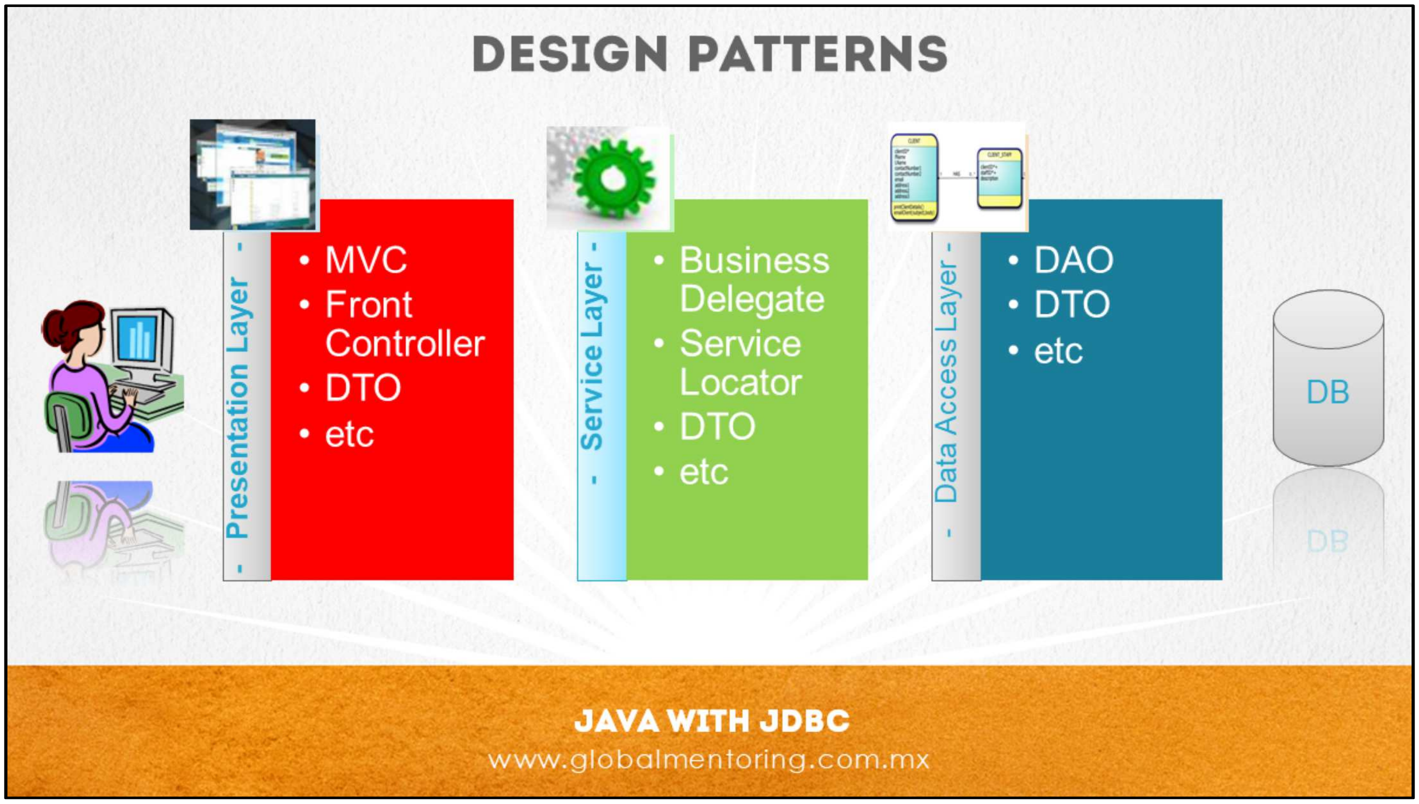


As we can see in the figure, we are showing components that present a flow for the extraction of information with a database.

We can observe the smallest number of possible relationships, since we are avoiding direct communication between the service layer and the layer that creates the connection to the database (under coupling).

If we had a relationship between the service component and the database connection, we would be generating more relationships than necessary and therefore generate greater dependence between each of the components.

One of the objectives in software design is to make the components reusable, so that the greater the dependency between each component, the more difficult it will be to take the module and adapt it for use in another application.



## DESIGN PATTERNS

- Presentation Layer -
  - MVC
  - Front Controller
  - DTO
  - etc
- Service Layer -
  - Business Delegate
  - Service Locator
  - DTO
  - etc
- Data Access Layer -
  - DAO
  - DTO
  - etc

JAVA WITH JDBC
www.globalmentoring.com.mx

Let's now review the theme of design patterns. A design pattern is a guide that can involve several classes and that in turn allows us to solve a problem that occurs repeatedly.

When we talk about the layers of a Java EE architecture, in this case the presentation layer, the service layer and the data access layer, each layer can have several design patterns and the presentation layer we can observe patterns like the MVC . This pattern means model-view-controller and its objective is to divide the responsibilities in these three areas, a model, a view and a controller. The Front Controller pattern allows us to provide a single entry to the user and therefore we can apply here several features such as some type of security, validations, among other validations.

The DTO (data transfer object) pattern represents an object of the domain of the problem, sometimes it can be an entity class, that is, a class that is persisted or saved in a database. This pattern appears in the three layers as we can see, because it is used to transfer an entity or a list of entities of a certain type between the different layers of the application, for example a user may be requesting a list of people, then the presentation layer processes the request and then requests the service layer to execute the method to find a list of people, later the services layer accesses the data layer so that we can retrieve the list of people and later we will return an arrangement of DTO types of Person type. The object DTO in this case is a person-type object and then we begin to return the information until we give a response to the user with the list of people who have requested.

In the business layer we have several patterns, such as the Business Delegate pattern, which handles the details of calling some service method and in turn we also have the Services Locater pattern, which is used by the Business Delegate pattern for locate the services if they are in a directory such as a JNDI directory (Java Naming and Directory Interface). We will see these patterns in more advanced courses, at the moment it is only important to know that there is a separation of responsibilities in each of the layers of a Java Enterprise application.

Finally, in the data layer we have the DAO (Data Access Object) pattern, this pattern is responsible for extracting and storing information in the database, and this is one of the patterns that we will use in this course.

It is worth mentioning that there is an extensive catalog of design patterns for Java, so if you require more information on this subject, you can consult: http://www.oracle.com/technetwork/java/patterns-139816.html

It is worth mentioning that the layers that we are showing are the most common that are used in the development of Java applications, but they are not the only ones and we could even have fewer layers, sometimes we only have a presentation layer that communicates with the database , although these types of practices are not recommended, unless they are prototypes. The type of final architecture will really depend on the problem we want to solve, but our recommendation is that we use at least these three layers so that the changes and maintenance in our systems are much simpler to perform.

ONLINE COURSE

JAVA WITH JDBC

By: Eng. Ubaldo Acosta

Global Mentoring

Experiencia y Conocimiento para tu vida

JAVA WITH JDBC

www.globalmentoring.com.mx