Java Design Pattern Interview Questions and Answers

Design patterns are generally sets of standardized practices used in the software development industry. Design Patterns represent the solutions given by the community to general problems faced in every-day tasks regarding software development.

Let's have a look at the most frequently asked **design pattern interview questions and answers**. These questions will help you with your coding interviews as well as competitive exams.

1) Categories Java Design patterns?

Based on problem analysis, we can categorize design patterns into the following categories.

Creational patterns

- Factory method/Template
- Abstract Factory
- Builder
- Prototype
- Singleton

Structural patterns

- Adapter
- Bridge
- Filter
- Composite
- Decorator

- Facade
- Flyweight
- Proxy

Behavioral patterns

- Interpreter
- Template method/ pattern
- Chain of responsibility
- Command pattern
- Iterator pattern
- Strategy pattern
- Visitor pattern

J2EE patterns

- MVC Pattern
- Data Access Object pattern
- Front controller pattern
- Intercepting filter pattern
- Transfer object pattern

2) Explain the advantages of Java design pattern?

- The Design Patterns are reusable in multiple projects.
- The Design Patterns provide a solution that helps to define the system architecture.
- The Design Patterns capture software engineering experiences.
- The Design Patterns provide transparency to the design of an application.
- They are testified and well-proved since they have been built upon the knowledge and experience of expert software developers.

3) What Is Gang of Four (GOF)?

In 1994, four authors Erich Gamma, Ralph Johnson, Richard Hel, and John Vlissides published a book titled **Design Patterns Elements of Reusable Object-Oriented Software.** This book introduced the concept of Design Pattern in Software development.

These four authors are known as Gang of Four GOF.

4) What are the Creational Patterns?

Creational design patterns are related to the way of creating objects. Creational design patterns are used when a decision is made at the time of instantiation of a class.

```
EmpRecord e1=new EmpRecord();
```

Since new keyword is used to create an object in Java, So, here we are creating the instance using the new keyword. In some cases, the nature of the object must be changed according to the nature of the program. In such cases, we should use the creational design patterns to provide a more general and flexible approach.

5) What Is Factory Pattern?

- It is the most used design pattern in Java.
- These design patterns belong to the Creational Pattern as this pattern provides one of the best ways to create an object.
- In the Factory pattern, we don't expose the creation logic to the client and refer the created object using a standard interface.
- Factory Pattern allows the sub-classes to choose the type of objects to create.
- The Factory Pattern is also known as **Virtual Constructor**.

6) What Is Abstract Factory Pattern?

Abstract Factory Pattern states that define an abstract class or interface for creating families of related objects but without specifying their concrete sub-classes. That means Abstract Factory allowed a class to return a factory of classes. That is why the Abstract Factory Pattern is one level higher than the Factory Pattern.

- Abstract Factory patterns work around superclasses, which creates other classes.
- The Abstract Factory Pattern comes under Creational Pattern as this pattern provides one of the best ways to create an object.

- In the Abstract Factory pattern, an interface is liable for creating a factory of related objects without explicitly identifying their classes.
- Each generated factory can give the objects according to the Factory pattern.

7) Explain Structural Patterns in Java?

Structural patterns are used to provide solutions and efficient standards regarding class compositions and object structures. They depend on the concept of inheritance and interfaces to allow multiple objects or classes to work together and form a single working whole.

Structural design patterns are responsible for how classes and objects can be composed to form larger structures.

8) Explain the Singleton pattern?

Singleton pattern in Java is a pattern which allows a single instance within an application. One good example of the singleton pattern is **java.lang.Runtime**.

Singleton Pattern states that define a class that has only one instance and provides a global point of access to it.

In other words, it is the responsibility of the class that only a single instance should be created, and all other classes can use a single object.

9) Describe in how many ways can you create a singleton pattern?

There are two ways of creating a Singleton pattern.

1. Early Instantiation

It is responsible for the creation of instance at load time.

2. Lazy Instantiation

It is responsible for the creation of instance when required.

10) What are the Adapter patterns?

Adapter pattern converts the interface of a class into another interface based on the requirement.

In other words, it let you convert the interface according to requirement while using the class service with a different interface.

It is also known as Wrapper.

11) Illustrate the uses of Adapter Patterns?

It is used in the following cases:

- When an object requires to utilize an existing class with an incompatible interface.
- In case we want to create a reusable class that collaborates with classes which don't have compatible interfaces.

12) Discuss the strategy to describe a design pattern?

The following points should need to be taken care to describe the design pattern.

- The Pattern name and classification.
- The Problem and solution.
- **Consequences:** Variation and language-dependent alternatives should also be addressed.
- **Uses:** Identify the uses in the real systems and its efficiency.

13) What is the decorator pattern in Java explain it with an example?

The decorator pattern is one of the popular Java design patterns. It is common because of its heavy usage in **java.io** (package). The Decorator Pattern uses composition in place of inheritance to extend the functionality of an object at runtime.

BufferedReader and **BufferedWriter** are some excellent examples of decorator pattern in Java.

14) Difference between Strategy and State design Pattern in Java?

This question is a commonly asked Java design pattern interview question as both Strategy and State pattern has the same structure. The UML class diagram of both patterns looks precisely the same, but their intent is different.

The state design pattern is used to manage and define the state of an object, while the Strategy pattern is used to describe a set of an interchangeable algorithm.

15) What are the advantages of Composite design Pattern in Java?

Composite design pattern allows clients to operate collectively on objects that may or may not represent a hierarchy of objects.

Advantage of composite design patterns is as follows.

- It describes the class hierarchy that contains primitive and complex objects.
- It makes it easy to add new kinds of the component.
- It facilitates with the flexibility of structure with a manageable class or interface.

16) Can you describe the uses of the composite pattern?

It is used in the following cases:

- When we want to represent a partial or full hierarchy of objects.
- In case we need to add the responsibilities dynamically to the individual object without affecting other objects.

17) What are Some Design Patterns which are used in the JDK library?

Some of the design patterns which are used in the JDK library are as follows.

- The decorator pattern is used by Wrapper classes.
- Singleton pattern is used by Calendar classes (Runtime).
- The Wrapper classes use factory pattern like Integer.valueOf.

- Event handling frameworks use observer pattern like swing, awt.

18) Mention advantage of Builder design pattern in Java?

Advantages of builder design patterns are as follows.

- It facilitates with a clear separation between the construction and representation of an object.
- It provides improved control over the construction process.
- The constructor parameter is reduced and is provided in highly readable method calls.
- In design Pattern, the object is always instantiated in a complete state.
- In the Builder design pattern, Immutable objects can be quickly built in the object building process.

19) Can you write Thread-safe Singleton in Java?

There are many ways to write a Thread-safe singleton in Java.

- Thread-safe Singleton can be written by writing singleton using double-checked locking.
- Another way is, by using static Singleton instance initialized during class loading.
- By using Java **enum** to create a thread-safe singleton, this is the most straightforward way.

20) Is it possible to create a clone of a singleton object?

Yes, it is possible to create a clone of a singleton object.

21) What is the proxy pattern, and what does it do?

The term Proxy stands for an object representing another object. The proxy pattern provides a substitute or placeholder for another purpose to control access to it.

According to Gangs of four, a Proxy Pattern "provides control for accessing the original object."

We can perform many security operations like hiding the information of the original object, on-demand loading, etc.

It is also called as placeholder or surrogates.

22) Explain some different type of proxies?

There are many cases where the proxy pattern is beneficial. Let's have a look at some different proxies.

Protection proxy

It controls access to the real subject based on some condition.

Virtual proxies

Virtual proxies are used to instantiate the expensive object. The proxy manages the lifetime of the real subject in the implementation.

It decides the need for the instance creation and when to reuse it. Virtual proxies optimize performance.

Caching proxies

Caching proxies are used to cache expensive calls to the real subject. There are many caching strategies that the proxy can use.

Some of them are read-through, write-through, cache-aside, and time-based. The caching proxies are used for enhancing performance.

Remote proxies

Remote proxies are used in distributed object communication. The remote proxy causes execution on the remote object by invoking a local object method.

Smart proxies

Smart proxies are used to implement log calls and reference counting to the object.

23) Explain the Chain of Responsibility Pattern?

In the chain of responsibility pattern, Sender sends a request to a chain of objects, and any object in the chain can handle the request.

A Chain of Responsibility Pattern avoids coupling the sender of a request to its receiver. For example, an ATM service uses the Chain of Responsibility design pattern in monetary transactions.

Moreover, we can explain that usually, each receiver contains the reference of another receiver. If one object can fail to handle the request, then it sends the same to the next receiver and so on.

24) Explain the advantage of Chain of Responsibilities Pattern and when it is used?

- It minimizes the coupling.
- It provides flexibility while assigning the responsibilities to objects.
- It permits a set of classes to act as one. The events produced in one class can be sent to other handler classes with the help of composition.

Usage of Chain of Responsibility Pattern

It is used in the following cases:

- When more than one objects are ready to handle a request, and the handler is unknown.
- In case the collection or a group of objects that can handle the request must be specified dynamically.

25) How is Bridge pattern is different from the Adapter pattern?

The motive of the Adapter pattern is to make interfaces of one or more classes to look similar.

The Bridge pattern is designed to isolate a class's interface from its implementation so we can vary or substitute the implementation without changing the client code.

26) What's the difference between the Dependency Injection and Service Locator patterns?

The service locator is used to create class dependencies. The Class is still responsible for creating its dependencies no matter whether if it is using service locator or not.

Service locators are also used to hide dependencies. We can't say by looking at an object whether it connects with a database or not when it obtains connections from a locator.

With Dependency injection, the class which contains its dependencies neither knows nor cares where they came from.

One significant difference is that Dependency injection is much easier to unit test because we can pass in it mock implementations of its dependent objects. We could combine the two objects and apply the service locator.

27) What are the MVC patterns?

This pattern is one of the most-used patterns from J2EE Design pattern category. It is quite similar to the concept of Model-View-Controller. The abbreviation MVC is taken from the Model-view-controller concept.

Models are objects, used as blueprints for all of the objects that will be used in the application.

Views contain the presentational aspect of the data and information located in the models.

Controllers control both model and view as they serve as a connection between the two objects. The controller plays the role of an interface between View and Model and also intercepts all the incoming requests.

28) Explain the Intercepting Filter Design Pattern and also mention its benefits?

The intercepting filter design pattern is used to intercept and manipulate a request and response before and after the request processing. Filters perform the authentication/ authorization/ logging or tracking of request and then forward the requests to corresponding handlers. Let's have a look at some basic entities of Intercepting design pattern.

Filter

It performs a certain task before or after the execution of request by request handler.

Filter Chain

It contains multiple filters and helps to execute them in defined order on target.

Target

The target object is the request handler

Filter Manager

It manages the filters and Filter Chain.

Client

The client object is one who sends a request to the Target object.

Benefits of Intercepting Filter Design Pattern

- Filter pattern provides central control with loosely coupled handlers.
- It expands reusability.
- The new filter can be added at any time without affecting the client's code.
- Filters can be selected dynamically during program execution.

29) Explain Data Access Object (DAO) pattern?

Data Access Object Pattern is used to isolate low-level data accessing API or actions from high-level business services. Following are the components in the DAO Pattern.

Data Access Object Interface

DAO interface describes the standard actions to be performed on a model object(s).

Data Access Object concrete class

This class implements a DAO interface. This class is accountable to get data from a data source which can be Xml/database or any other storage mechanism.

Model Object or Value Object

This object is a plain old java object containing get/set methods to store data retrieved using DAO class.

30) Mention what is the difference between VO and JDO?

The difference between VO and JDO is that the JDO is a persistent technology that competes against entity beans. It allows to create POJO (plain old java objects) and persevere them to the database.

While VO (value objects) represents an abstract design pattern, that is used in conjunction with entity beans, JDBC and JDO.

Interview Tips	**Job/HR Interview Questions**
JavaScript Interview Questions	**jQuery Interview Questions**
Java Basics Interview Questions	**Java OOPs Interview Questions**
Servlet Interview Questions	**JSP Interview Questions**
Spring Interview Questions	**Hibernate Interview Questions**
PL/SQL Interview Questions	**SQL Interview Questions**
Oracle Interview Questions	**Android Interview Questions**
SQL Server Interview Questions	**MySQL Interview Questions**