Java Design Patterns Interview Questions and Answers

Author: Ramesh Fadatare
Design Patterns Interview
In this article, we will discuss the Java design patterns interview questions asked for experienced Java developers.

Let's discuss the most useful and frequently asked interview questions of Java/Java EE design patterns.

1. Can you name a few design patterns used in the standard JDK library?

Here is a list of design patterns used in **Java's core libraries.**

Abstract factory

Provide an interface for creating families of related or dependent objects without specifying their concrete classes.

- javax.xml.parsers.DocumentBuilderFactory#newInstance()
- javax.xml.transform.TransformerFactory#newInstance()
- javax.xml.xpath.XPathFactory#newInstance()

Builder

Separate the construction of a complex object from its representation so that the same construction process can create different representations.

- java.lang.StringBuilder#append() (unsynchronized)
- java.lang.StringBuffer#append() (synchronized)
- java.nio.ByteBuffer#put() (also on CharBuffer, ShortBuffer, IntBuffer, LongBuffer, FloatBuffer and DoubleBuffer)
- javax.swing.GroupLayout.Group#addComponent()
- All implementations of java.lang.Appendable

Factory method

Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.

- java.util.Calendar#getInstance()
- java.util.ResourceBundle#getBundle()

- java.text.NumberFormat#getInstance()
- java.nio.charset.Charset#forName()
- java.net.URLStreamHandlerFactory#createURLStreamHandler(String) (Returns singleton object per protocol)
- java.util.EnumSet#of()
- javax.xml.bind.JAXBContext#createMarshaller() and other similar methods

Prototype

Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype.

- java.lang.Object#clone() (the class has to implement java.lang.Cloneable)

Singleton

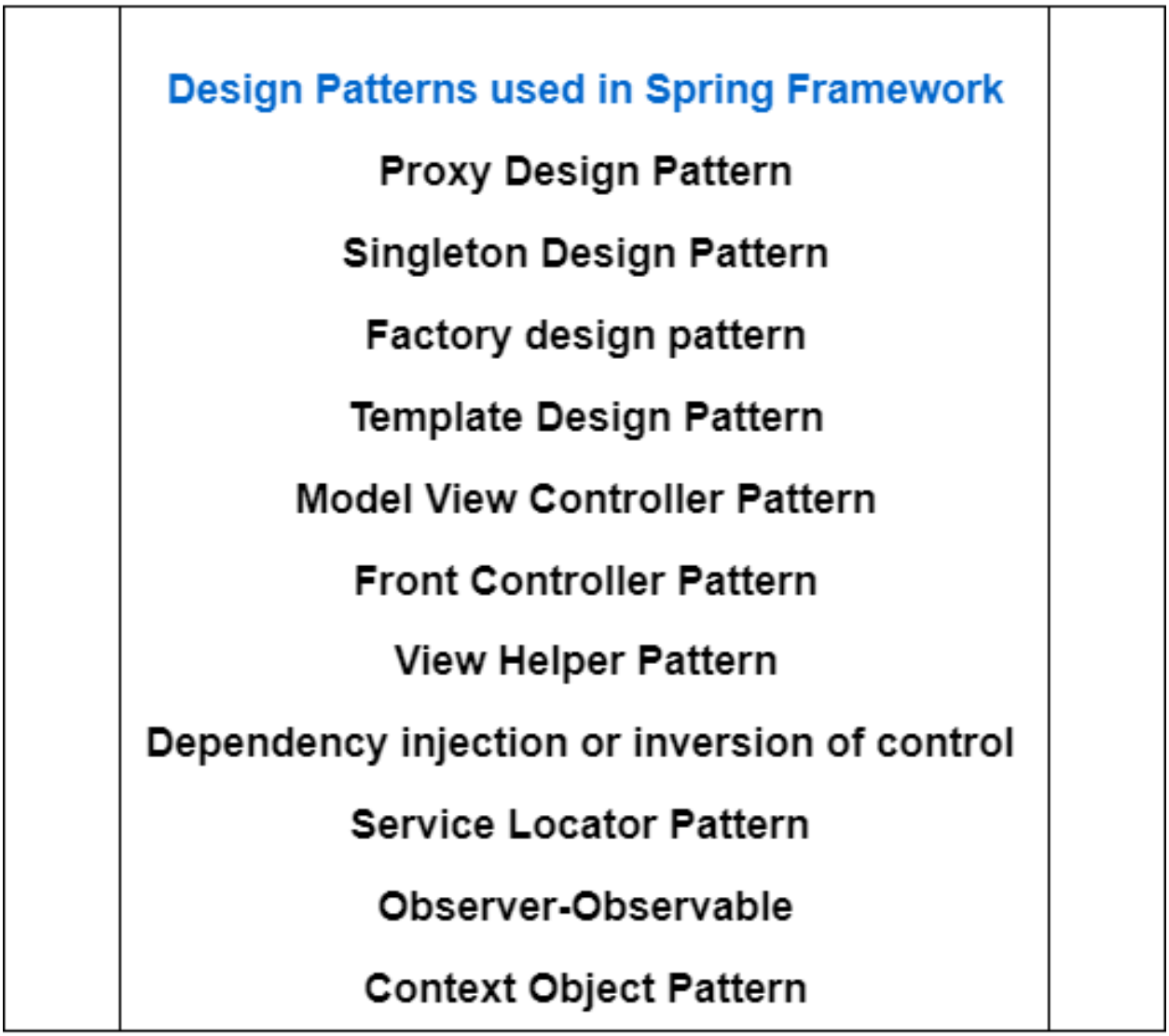
Ensure a class only has one instance, and provide a global point of access to it.

- java.lang.Runtime#getRuntime()
- java.awt.Desktop#getDesktop()
- java.lang.System#getSecurityManager()

Read the complete list of design patterns used in standard JDK libraries at Examples of GoF Design Patterns in Java's core libraries

2. What design patterns are used in the Spring framework?

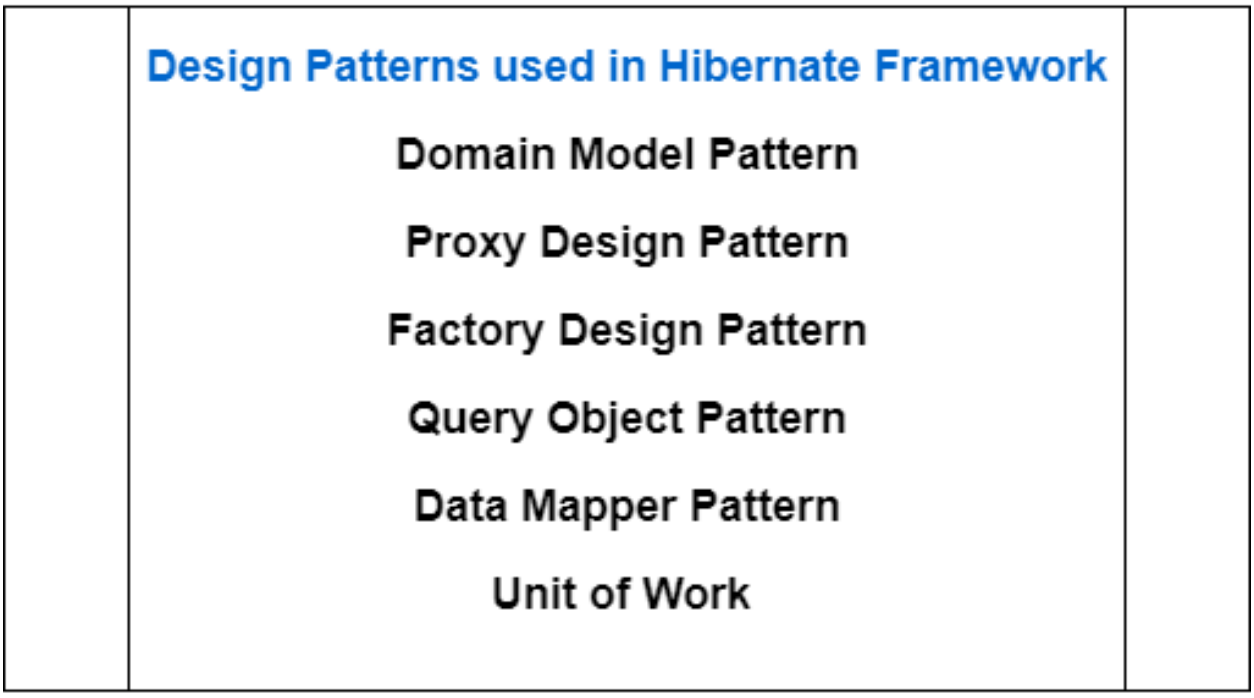
Here is a list of a few known design patterns used in Spring Framework.



Read more about each of the above design patterns in-detail at Design Patterns used in Spring Framework

3. What are design patterns used in Hibernate Framework?

Here is a list of a few known design patterns used in **Hibernate Framework**.



Read more about each of the above design patterns in-detail at Design Patterns used in Hibernate Framework

4. What are the design patterns used in your current projects?

You can explain the design patterns you have been using in yours with real-time scenarios.

5. What is a Singleton design pattern in Java? write code for a thread-safe singleton in Java

Here is a Solution.

6. What are the advantages of the Factory Pattern and when to use the Factory Pattern?

Advantages of the Factory Design Pattern

- Factory Method Pattern allows the sub-classes to choose the type of objects to create.
- It promotes the loose-coupling by eliminating the need to bind application-specific classes into the code. That means the code interacts solely with the resultant interface or abstract class so that it will work with any classes that implement that interface or that extend that abstract class.

Use the Factory Method pattern when

- a class can't anticipate the class of objects it must create
- a class wants its subclasses to specify the objects it creates
- classes delegate responsibility to one of several helper subclasses, and you want to localize the knowledge of which helper subclass is the delegate

Read more about Factory design pattern at Factory Design Pattern in Java with Examples

7. Name a few Object-Oriented Design Principles

1. Encapsulate What Varies
2. Code to an Interface Rather Than to an Implementation
3. Delegation Principle
4. The Open-Closed Principle (OCP)
5. Dry- Don't Repeat Yourself
6. Single Responsibility Principle (SRP)
7. Liskov's Substitution Principle (LSP)
8. Interface Segregation Principle (ISP)
9. Dependency Injection or Inversion Principle

Read more at Object-Oriented Design Principles in Java

8. What is the use of Design Patterns?

- Design patterns can speed up the development process by providing tested, proven development paradigms. Effective software design requires considering issues that may not become visible until later in the implementation. Reusing design patterns helps to prevent subtle issues that can cause major problems and improves code readability for coders and architects familiar with the patterns.

- Often, people only understand how to apply certain software design techniques to certain problems. These techniques are difficult to apply to a broader range of problems. Design patterns provide general solutions, documented in a format that doesn't require specifics tied to a particular problem.

- In addition, patterns allow developers to communicate using well-known, well-understood names for software interactions. Common design patterns can be improved over time, making them more robust than ad-hoc designs.

9. Name a few GOF design patterns?

The GoF Design Patterns are broken into three categories:

1. Creational Patterns for the creation of objects;
2. Structural Patterns to provide relationships between objects
3. Behavioral Patterns to help define how objects interact.

1. Creational Patterns

In software engineering, creational design patterns are design patterns that deal with object creation mechanisms, trying to create objects in a manner suitable to the situation.

- **Singleton Design Pattern**

- **Factory Design Pattern**

- **Abstract Factory Design Pattern**

- **Builder Design Pattern**

- **Prototype Design Pattern**

- **Object Pool Design Pattern**

2. Structural Patterns

In Software Engineering, Structural Design Patterns are Design Patterns that ease the design by identifying a simple way to realize relationships between entities.

- **Adapter Design Pattern**
- **Bridge Design Pattern**
- **Composite Design Pattern**
- **Decorator Design Pattern**
- **Facade Design Pattern**
- **Flyweight Design Pattern**
- **Proxy Design Pattern**

3. Behavioral Patterns

In software engineering, behavioral design patterns are design patterns that identify common communication patterns between objects and realize these patterns. By doing so, these patterns increase flexibility in carrying out this communication.

- **Chain of responsibility**
- **Command Design Pattern**
- **Iterator Design Pattern**
- **Mediator Design Pattern**
- **Memento Design Pattern**

- Observer Design Pattern
- State Design Pattern
- Strategy Design Pattern
- Template Method Design Pattern
- Delegation Pattern

10. Name a few Java EE desgin patterns

List of Core J2EE Design Patterns referred from book Core J2EE Patterns: Best Practices and Design Strategies (2nd Edition).

Presentation Tier

- Intercepting Filter Design Pattern in Java
- Front Controller Design Pattern in Java
- Application Controller Design Pattern in Java
- View Helper Design Pattern in Java
- Composite View Design Pattern in Java
- Context Object Design Pattern in Java

Business Tier

- Data Transfer Object Design Pattern in Java
- Service Locator Design Pattern in Java
- Business Delegate Design Pattern in Java
- Converter Design Pattern in Java
- Transfer Object Assembler Pattern in Java
- Value List Handler Pattern in Java

Integration Tier

- Data Access Object Pattern in Java
- Service Activator Pattern in Java

11. What is Singleton's design pattern and when to use it?

Definition

Ensure a class only has one instance, and provide a global point of access to it.

Use the Singleton pattern when

- there must be exactly one instance of a class, and it must be accessible to clients from a well-known access point.
- when the sole instance should be extensible by subclassing, and clients should be able to use an extended instance without modifying their code.

12. What is the Builder design pattern and when to use it?

Definition

Separate the construction of a complex object from its representation so that the same construction process can create different representations.

Use the Builder pattern when

- the algorithm for creating a complex object should be independent of the parts that make up the object and how they're assembled.
- the construction process must allow different representations for the object that's constructed.