# Java Design Pattern Interview Questions

*Home (http://www.codespaghetti.com) → Java Design Pattern Interview Questions*

# ns

.ons, Programs and Examples.

esign Patterns

ttern Question

rn Interview Questions

Interview Questions

Interview Questions

nterview Questions

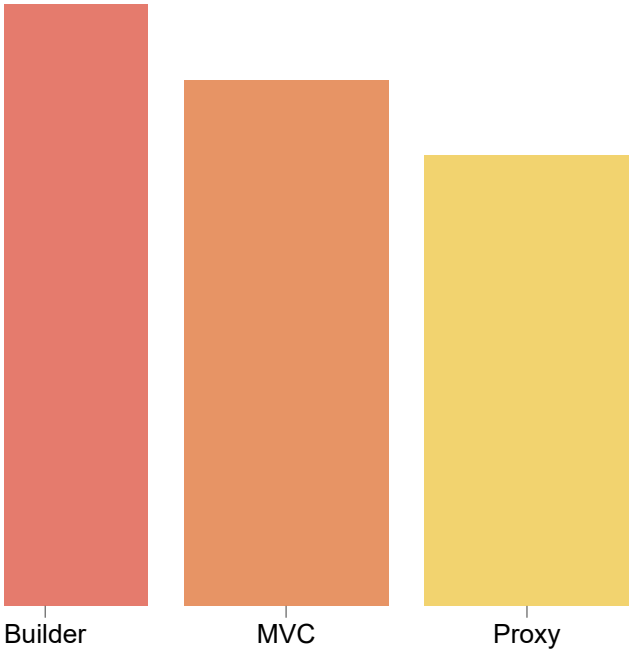
rn Interview Questions

Interview Resources

gn Patterns

ttern Interview Questions PDF

# Pattern Interview Questions




# s of Design Patterns?

**solution** for solving the specific problem.

es in reusability and extensibility of the already developed applicatio

*ed concepts* like decomposition, inheritance and polymorphism.

elp to define the system architecture.

# ges of Design Patterns?

rect code reuse.

imple.

erload.

a software development process is a human-intensive activity.

# tern Interview Questions

## Design Pattern do

instance, and provides a global point of access to it.

## n?

e of a class, and it must be accessible to clients from a well-known ac

extensible by subclassing, and clients should be able to use an extende

code

se

# gleton

er()

## ı

inciple (SRP) by controlling their own creation and lifecycle.

instance which prevents an object and resources used by this object fro

is difficult to test. Makes it almost impossible to subclass a Singletc

# ingleton design pattern?

lest design pattern.It is used when an application needs one, and only

s only one instance, and provide a global point of access to this insta

ime initialization" of the object

if following criteria are satisfied:

cannot be reasonably assigned

ovided for

---

tonInstance = new Singleton();

so that this class cannot be instantiated

ance(){

---

ton class

```
 args) {


nstructor Singleton() is not visible
gleton();

ole
getInstance();
```

in the "single instance" class.

nction in the class.

on first use) in the accessor function.

ected or private .

function to manipulate the Singleton.

# ny ways can you create singleton pattern

two ways.

# etons getInstance is not synchronized , h
# ultithreaded environment?

, we can't guarantee when the method is unsynchronized it would give

# rite thread-safe Singleton in Java?

ead-safe singleton in Java

checked locking.

e initialized during class loading.

ate thread-safe singleton this is most simple way.

# etons getInstance is not synchronized , h
# ultithreaded environment?

, we can't guarantee when the method is unsynchronized it would give

# rite thread-safe Singleton in Java?

ead-safe singleton in Java

checked locking.

e initialized during class loading.

ate thread-safe singleton this is most simple way.

# eate a clone of a singleton object?

# vent cloning of a singleton object?

hin the body of the clone() method.

# ger initialization in singleton?

e of Singleton Class is created at the time of class loading.

as a drawback that instance is created even though it might not be usin

alized instance = new EagerInitialized();

client applications to use constructor

getInstance(){

ɔt of resources, this is the approach to use.

eton classes are created for resources such as Database connections etc
. client calls the getInstance method.

# ıtic block initialization?

ntation is similar to eager initialization, except that instance of c
ɔvides option for exception handling.

---

:ance;

for exception handling

.ock();

:ion("Exception occurred in creating singleton instance");

ıstance(){

---

.c block initialization creates the instance even before it’s being u

:.

# zy Initialization in singleton?

ient Singleton pattern creates the instance in the global access method.

;leton class with this approach.

---

instance;

;etInstance(){

:ialized();

---

incase of single threaded environment but when it comes to multithreade

:iple threads are inside the if loop at the same time. It will destroy t

/ill get the different instances of singleton class.

# read Safe Singleton in Java?

ıfe singleton class is to make the global access method synchronized.

e this method at a time. General implementation of this approach is l

---

eton instance;

.

eadSafeSingleton getInstance(){

nfeSingleton();

---

nd provides thread-safety but it reduces the performance because c

thod.

e first few threads who might create the separate instances (Reac

time, double checked locking principle is used. In this approach.

de the if condition with an additional check to ensure that only one i

ed locking implementation.

```
getInstanceUsingDoubleLocking(){

gleton.class) {

adSafeSingleton();
```

# um Singleton in Java?

lection, Joshua Bloch suggests the use of Enum to implement Singleton
m value is instantiated only once in a Java program.

y accessible, so is the singleton. The drawback is that the enum
does not allow lazy initialization.

```
){
```

# plement singleton class with Serialization

need to implement Serializable interface in Singleton class.

file system and retrieve it at later point of time. Here is a small si

nterface also.

---

plements Serializable{

lVersionUID = -1;

.

lelper{
izedSingleton instance = new SerializedSingleton();

on getInstance(){
ance;

---

singleton class is that whenever we deserialize it, it will create

with a simple program.

```
on;

st {

 ] args) throws FileNotFoundException, IOException, ClassNotFoundExcepti
eOne = SerializedSingleton.getInstance();
ctOutputStream(new FileOutputStream(

);


 object
InputStream(new FileInputStream(

eTwo = (SerializedSingleton) in.readObject();


eOne hashCode="+instanceOne.hashCode());
eTwo hashCode="+instanceTwo.hashCode());
```

n, to overcome this scenario all we need to do it provide the impleme

Code of both the instances are same in test program.

# va Implementation Code With Junits

tp://www.codespaghetti.com/wp-
leton-1.zip)

# rn Interview Questions

# attern Do ?

; an object, but let subclasses decide which class to instantiate. Facto

:ion to subclasses.

s of objects it must create

ɔecify the objects it creates

ɔ one of several helper subclasses, and you want to localize the knowle

gate

s

racle.com/javase/8/docs/api/java/util/Calendar.html#getInstance--)

docs.oracle.com/javase/8/docs/api/java/util/ResourceBundle.html#getBund

cs.oracle.com/javase/8/docs/api/java/text/NumberFormat.html#getInstance

docs.oracle.com/javase/8/docs/api/java/nio/charset/Charset.html#forName

docs/api/java/net/URLStreamHandlerFactory.html#createURLStreamHandler-

racle.com/javase/8/docs/api/java/util/EnumSet.html#of-E-)

/docs/api/javax/xml/bind/JAXBContext.html#createMarshaller--)

d within Template Methods.

inheritance. Prototype: creation through delegation.

is that it can return the same instance multiple times, or can return
that exact type.

ful. There is a difference between requesting an object and creating one
ject, and fails to encapsulate object creation. A Factory Method enforc
object to be requested without inextricable coupling to the act of cre

# you prefer to use a Factory Pattern?

the following cases:

of objects it must create

we need to create an object of any one of sub-classes depending on t

## between Factory and Abstract Factory

of objects delegated to a separate factory class whereas Abstract Facto

y which creates other factories.

## between Factory and Builder Design

Factory pattern wherein the Builder class builds a complex object in mu

## between Factory and Strategy Design

rn whereas Strategy is behavioral design pattern. Factory revolves arou

s Strategy or Policy revolves around the decision at runtime.

## in benefit of using factory pattern?

oproach to code for interface rather than implementation.

ntiation of actual implementation classes from client code.

re robust, less coupled and easy to extend. For example, we can easily

lient program is unaware of this.

ion between implementation and client classes through inheritance.

## od  java implementation with Junits

espaghetti.com/wp-content/uploads/2017/03/factory-method.zip)

## rn Interview Questions

# ttern do?

struction of a complex object from its representation so that th
erent representations.

## ?

lex object should be independent of the parts that make up the object a

ow different representations for the object that's constructed

ocs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html)
oracle.com/javase/8/docs/api/java/nio/ByteBuffer.html#put-byte-) as wel
er, IntBuffer and so on.
cs.oracle.com/javase/8/docs/api/java/lang/StringBuffer.html#append-bool
Appendable
docs/api/java/lang/Appendable.html)
thub.com/apache/camel/tree/0e195428ee04531be27a0b659005e3aa8d159d23/cam
el/builder)

complex object step by step. Abstract Factory emphasizes a family of p
). Builder returns the product as a final step, but as far as the Abstr
gets returned immediately.

ctory Method (less complicated, more customizable, subclasses prolifera
rototype, or Builder (more flexible, more complex) as the designer disc

# advantage of Builder Design Pattern?

rn are as follows:
een the construction and representation of an object.
onstruction process.
l representation of objects.

# cific problems builder pattern solves?

some of the problems with Factory and Abstract Factory design patterns

attributes. Builder pattern solves the issue with large number of o
y providing a way to build the object step-by-step and provide a meth
t.

# e need to consider when implementing

possible representations (or outputs) is the problem at hand.
mmon input in a Reader class.

eating all possible output representations. Capture the steps of this p

r each target representation.

t and a Builder object, and registers the latter with the former.

nstruct".

turn the result.

## n java implementation with Junits

tti.com/wp-content/uploads/2017/03/builder.zip)

## Interview Questions

# tern do?

or placeholder for another object to control access to it.

there is a need for a more versatile or sophisticated reference to an

ral common situations in which the Proxy pattern is applicable

resentative for an object in a different address space.

ojects on demand.

to the original object. Protection proxies are useful when objects shou

rface to its subject. Proxy provides the same interface. Decorator prov

nt purposes but similar structures. Both describe how to provide a l

nd the implementations keep a reference to the object to which they

# ifferent proxies?

:he proxy pattern is useful.

: to the real subject. Based on some condition the proxy filters the ca
) the real subject.

an object is expensive to instantiate. In the implementation the proxy

on is needed and when it can be reused. Virtual proxies are used to o

che expensive calls to the real subject. There are multiple caching str

'ite-through, cache-aside and time-based. The caching proxies are us

stributed object communication. Invoking a local object method on the
: object.

:ment reference counting and log calls to the object.

# between Proxy and Adapter?

ıt than the real subject whereas Proxy object has the same input as t

be placed as it is in place of the real subject.

# ı java implementation with Junits

i.com/wp-content/uploads/2017/03/proxy.zip)

# ttern Interview Questions

# ' pattern do?

al responsibilities to an object dynamically. Decorators provide a f
ıding functionality

## ern?

ıual objects dynamically and transparently, that is, without affecting

withdrawn
impractical. Sometimes a large number of independent extensions are pos
subclasses to support every combination. Or a class definition may be
lassing

oracle.com/javase/8/docs/api/java/io/InputStream.html), java.io.OutputS
docs/api/java/io/OutputStream.html), java.io.Reader
docs/api/java/io/Reader.html) and java.io.Writer
docs/api/java/io/Writer.html)
dXXX()
docs/api/java/util/Collections.html#synchronizedCollection-

eXXX()
docs/api/java/util/Collections.html#unmodifiableCollection-

)

docs/api/java/util/Collections.html#checkedCollection-java.util.Collect

rface to its subject. Proxy provides the same interface. Decorator prov

face, Decorator enhances an object's responsibilities. Decorator is thu
onsequence, Decorator supports recursive composition, which isn't possi

lar structure diagrams, reflecting the fact that both rely on recursive
nded number of objects.
generate Composite with only one component. However, a Decorator adds
isn't intended for object aggregation.
add responsibilities to objects without subclassing. Composite's focus
ation. These intents are distinct but complementary. Consequently, Comp
oncert.
onsibility to let components access global properties through their par
ide these properties on parts of the composition.
t purposes but similar structures. Both describe how to provide a level
d the implementations keep a reference to the object to which they forw

n of an object. Strategy lets you change the guts.

# view Resources:

eering-and-computer-science/6-189-multicore-programming-primer-january-iap-2007/lectur

ogramming-i/ (https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-1

2007/lecture-notes-and-video/l6-design-patterns-for-parallel-programming-i/)

eering-and-computer-science/6-170-laboratory-in-software-engineering-fall-2005/lecture-

s/electrical-engineering-and-computer-science/6-170-laboratory-in-software-engineering-fal

erns (https://www.coursera.org/learn/design-patterns)

(https://sourcemaking.com/design_patterns)

---

important topic of object oriented software development.

ng more and more complex, and it is an essential quality of a good s
pattern should be utilized to solve the problems in the most efficient

e (http://www.codespaghetti.com/array-interview-questions/) and alg
lgorithms-questions/) are essential components of a technical intervi
yourself in these areas, then you will keep r
iew-rejection) over and over again.

vast and complex. You need to make your self familiarize with th
to increase your chances of success in int
iew-success).

# iew Questions PDF

**New Bonus PDF**

**wnload a free PDF version of this guide. It contains all the links, tips and resources explained here... Plus its print friendly.**

able Object-Oriented Software (http://www.amazon.com/Design-Patterns-El
33612)
(http://java-design-patterns.com/)
DesignPatterns/article.html
/DesignPatterns/article.html)
n-Patterns-Object-Oriented-Software (https://www.quora.com/topic/Design
)
tware_design_pattern (https://en.wikipedia.org/wiki/Software_design_pat
atterns/decorator (https://sourcemaking.com/design_patterns/decorator)

Can I ask you a small favour?

(/#facebook) (/#twitter) (/#linkedin) (/#google_plus)

(/#email)

## About Us

My journey from getting rejected by 150+ companies to getting job offers from Google, Microsoft, and Amazon.

→ Read More (http://www.codespaghetti.com/about-codespaghetti/)

→ Privacy Policy (http://www.codespaghetti.com/privacy-policy/)

→ Terms of Service (http://www.codespaghetti.com/terms-of-service/)

→ Contact us ( http://www.codespaghetti.com/contact/)

## Interview

→ Interview sex (http://www.codespaghetti.com/interview-sex/)

→ Interview hack (http://www.codespaghetti.com/interview-hack/)

→ Phone interview (http://www.codespaghetti.com/phone-interview/)

→ Interview memory (http://www.codespaghetti.com/interview-memory/)

→ Interview rejection (http://www.codespaghetti.com/interview-rejection/)

→ Technical interview (http://www.codespaghetti.com/interview-success/)

## Questions

→ Arrays (http://www.codespaghetti.com/array-interview-questions/)

→ ArrayList (http://www.codespaghetti.com/arraylist-interview-questions/)

→ HashMap (http://www.codespaghetti.com/java-hashmap-interview-questions/)

→ LinkedList (http://www.codespaghetti.com/linked-list-interview/)

→ Algorithms (http://www.codespaghetti.com/java-algorithms-questions/)

→ DesignPatterns (http://www.codespaghetti.com/java-design-pattern-interview-questions/)

Job search

→ Google

→ CV tips (http://www.codespaghetti.com/cv-tips/)

→ Job search (http://www.codespaghetti.com/interview-invitation/)

→ Graduates

→ Internships

→ Cover letter (http://www.codespaghetti.com/cover-letter/)

Subscribe



(https://www.facebook.com/JobTactics-552335738298069)

(https://twitter.com/CodeSpaghetti1)

(https://plus.google.com/113605596732441570563)

(https://www.linkedin.com/in/mr-brooks-635b63153/)

(https://www.pinterest.com/codespaghettics/)

(https://www.youtube.com/channel/UCRBQ2kdU73umISJc8-zs9bg)