

Java-Success.com

Industrial strength Java/JEE Career Companion for those who want to go places

[Home](#)
[Java FAQs](#)
[600+ Java Q&As](#)
[Career](#)
[Tutorials](#)
[Member](#)
[Why?](#)
[Can u Debug?](#)
[Java 8 ready?](#)
[Top X](#)
[Productivity Tools](#)
[Judging Experience?](#)

[Home](#) › [member-paid](#) › ♦ Java immutable objects interview questions & answers

♦ Java immutable objects interview questions & answers

Posted on [August 16, 2014](#) by [Arulkumaran Kumaraswamipillai](#) — No

[Comments](#) ↓



Best Practice: “Classes should be immutable unless there’s a very good reason to make them mutable....If a class cannot be made immutable, limit its mutability as much as possible.”
— by Joshua Bloch

Q1. What is an immutable object?

Q2. Immutable objects are objects whose state (the object’s data) cannot change after construction. Examples of immutable objects from the JDK include String and wrapper classes like Integer, Double, Character, etc.

[9 tips to earn more](#) | [What can u do to go places?](#) | **945+** members. [LinkedIn Group](#). [Reviews](#)

600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

[open all](#) | [close all](#)

☰ [Ice Breaker Interview](#)

- └─ 01: ♦ 15 Ice breakers
- └─ 02: ♥♦ 8 real life examples
- └─ 03: ♦10+ Know your Java
- └─ 04: Can you think of 10 immutable classes?
- └─ 05: ♥ What job interviews ask about immutability?
- └─ 06: ► Tell me about the benefits of immutability
- └─ 07: ♥ 20+ Pre interview questions

☰ [Core Java Interview Questions](#)

☰ [Java Overview \(4\)](#)

└─ 01: ♦ ♥ 17 Java Interview Questions

Q2.How do you create an immutable type?

A2.

1. Make the class **final** so that it cannot be extended or use static factories and keep the constructors private.

```
1 public final class MyImmutable { ... }
```

2. Make the fields private and final.

```
1 private final int[] myArray;
```

3. Don't provide any methods that can change the state of the immutable object in any way from outside the object – not just `setXXX` methods, but any methods which can change the state.

4. The “**this**” reference is not allowed to escape during construction from the immutable class, and the immutable class should have exclusive access to fields that contain references to other mutable objects like arrays, collections and mutable classes like `Date` by:

- a) Declaring the mutable references as private.
- b) Not returning or exposing the mutable references to the caller. This can be done by defensively copying the objects by deeply cloning them.

Example: satisfying the above conditions

```
1 public class User {
2     private final String firstName; //final, and
3     private final String lastName; //final, and
4
5     // constructor is private
6     private User(String firstName, String lastName) {
7         this.firstName = firstName;
8         this.lastName = lastName;
9     }
10
11     //factory method
12     public static User getInstance(String firstName, String lastName) {
13         return new User(firstName, lastName);
14     }
15 }
```

02: ♥♦ Java Con

03: ♦ 9 Core Jav

04: ♦ Top 10 mos

☐ Data types (6)

01: Java data ty

02: ♥♦ 10 Java S

03: ♦ ♥ Java aut

04: Understandir

05: Java primitiv

Working with Da

☐ constructors-methc

Java initializers,

☐ Reserved Key Wor

♥♦ 6 Java Modifi

Java identifiers

☐ Classes (3)

♦ Java abstract c

♦ Java class loa

♦ Java classes a

☐ Objects (8)

► Beginner Jav

♥♦ HashMap & H

♦ 5 Java Object

♦ Java enum inte

♦ Java immutabl

♥♦ Object equals

Java serialization

Mocks, stubs, dc

☐ OOP (10)

♥ Design princip

♦ 30+ FAQ Java

♦ Why favor cor

08: ♦ Write code

Explain abstracti

How to create a

Top 5 OOPs tips

Top 6 tips to go

Understanding C

What are good r

☐ GC (2)

♦ Java Garbage

```

15
16     //only getters, no setters
17
18     public String getFirstName() {
19         return firstName;
20     }
21
22     public String getLastName() {
23         return lastName;
24     }
25
26 }

```

Q3. Is the following class immutable?

```

1  import java.util.Arrays;
2
3  public final class MyImmutable {
4
5      private final Integer[] myArray;
6
7      public MyImmutable(Integer[] anArray) {
8          this.myArray = anArray;
9      }
10
11     public Integer[] getMyArray() {
12         return myArray;
13     }
14
15     //....equals(), hashCode(), etc
16
17     @Override
18     public String toString() {
19         return Arrays.toString(myArray);
20     }
21 }

```

A3. No. The above class is **not immutable** as it fails #4 condition where the “myArray” reference can escape, and mutated from outside as demonstrated below.

```

1  public class MyImmutableTest {
2
3      public static void main(String[] args) {
4          Integer[] array1 = {1,2,3};
5          MyImmutable mi = new MyImmutable(array1);
6          System.out.println("before modifying: ");
7
8          mi.getMyArray()[2] = 4; //change 3 to 4
9          System.out.println("after modifying: " +
10      }
11 }

```

Output:

03: Java GC tun

Generics (5)

♥ Java Generics

♥ Overloaded m

♦ 12 Java Gener

♦ 7 rules to reme

3 scenarios to ge

FP (8)

01: ♦ 19 Java 8 I

02: ♦ Java 8 Stre

03: ♦ Functional

04: ♥♦ Top 6 tips

05: ♥ 7 Java FP

Fibonacci numb

Java 8 String str

Java 8: What is c

IO (7)

♥ Reading a text

♦ 15 Java old I/C

06: ♥ Java 8 way

Processing large

Processing large

Read a text file f

Reloading config

Multithreading (12)

01: ♥♦ 15 Beginn

02: ♥♦ 10+ Java

03: ♦ More Java

04: ♦ 6 popular J

05: ♦ How a thre

06: ♦ 10+ Atomic

07: 5 Basic multi

08: ♦ ThreadLoc

09: Java FutureT

10: ♦ ExecutorSe

Java ExecutorSe

Producer and Co

Algorithms (5)

♦ Splitting input t

♦ Tree traversal :

♥ ♦ Java coding

```
1 before modifying: [1, 2, 3]
2 after modifying: [1, 2, 4]
```

FIX: “myArray” reference is deeply copied, and can’t escape

```
1 import java.util.Arrays;
2
3 public final class MyImmutable {
4
5     private final Integer[] myArray;
6
7     public MyImmutable(Integer[] anArray) {
8         this.myArray = anArray.clone(); //cloned
9     }
10
11     public Integer[] getMyArray() {
12         return myArray.clone(); // cloned array
13     }
14
15     // ....equals(), hashCode(), etc
16
17     @Override
18     public String toString() {
19         return Arrays.toString(myArray);
20     }
21 }
```

If you run the “MyImmutableTest” again,

```
1 before modifying: [1, 2, 3]
2 after modifying: [1, 2, 3]
3
```

Q4. What are the advantages of immutable objects?

A4.

1) Immutable classes can greatly simplify programming by freely allowing you to cache and share the references to the immutable objects without having to defensively copy them or without having to worry about their values becoming stale or corrupted.

2) Immutable classes are inherently thread-safe and you do not have to synchronize access to them to be used in a multi-threaded environment. So there are no chances for negative

Searching algori

Swapping, partiti

Annotations (2)

8 Java Annotati

More Java annot

Collection and Data

Find the first n

Java Collection

Java Iterable \

HashMap & H

Sorting objects

02: Java 8 Stre

04: Understandi

4 Java Collection

If Java did not ha

Java 8: Different

Part-3: Java Tre

Sorting a Map by

When to use whi

Differences Between

Java Iterable \

Multithreading

Why do Proxy,

Core Java Modif

Differences betw

Java Collection i

Event Driven Progr

Event Driven Pr

Event Driven Pr

Exceptions (2)

Java exception

Top 5 Core Java

Java 7 (2)

Java 7 fork and j

Java 7: Top 8 ne

Java 8 (24)

01: 19 Java 8 I

02: Java 8 Stre

03: Functional

04: Top 6 tips

04: Convert Lists

performance consequences as multiple threads can share the same instance.

3) Eliminates the possibility of data becoming inaccessible when used as keys in HashMaps or as elements in Sets.

These types of errors are hard to debug and fix.

4) Eliminates the need for class invariant check once constructed.

5) Allow hashCode() method to use lazy initialization, by caching its return value.

6) Cloning is not required.

7) Simpler to construct, use, and test due to its deterministic state.

Q5. Why is it a best practice to implement the user defined key class as an immutable object?

A5. Immutable objects generally make the best map keys as the keys cannot be modified once they have been added to the Map. In general *String*, *Integer*, or *Long* are used as keys, which are immutable objects. If you define your own key class, make sure that they are immutable. Otherwise, if the keys are accidentally modified after adding to a Map, you will never be able to retrieve the stored value as the key values have been changed. This is a common pitfall many Java developers, especially beginners fall for.

Example:

Immutable key class

```

1
2 import java.util.Date;
3
4 public final class MyKey {
5
6     private final String name;
7     private final Date myDate;
8
9     public MyDiary(Date aDate) {

```

04: Understanding

05: ♥ 7 Java FP

05: ♦ Finding the

06: ♥ Java 8 way

07: ♦ Java 8 API

08: ♦ Write code

10: ♦ ExecutorSe

Fibonacci numbe

Java 8 String str

Java 8 using the

Java 8: 7 useful

Java 8: Different

Java 8: Does "O

Java 8: What is c

Learning to write

Non-trivial Java 8

Top 6 Java 8 fea

Top 8 Java 8 fea

Understanding J

☐ JVM (6)

♦ Java Garbage

01: jvisualvm to s

02: jvisualvm to c

05: Java primitiv

06: ♦ 10+ Atomic

5 JMX and MBes

☐ Reactive Program

07: Reactive Pro

10: ♦ ExecutorSe

3. Multi-Threadir

☐ Swing & AWT (2)

5 Swing & AWT i

Q6 – Q11 Swing

☐ JEE Interview Q&A (3

☐ JEE Overview (2)

♦ 8 Java EE (aka

Java EE interview

☐ Web basics (8)

01: ♦ 12 Web ba

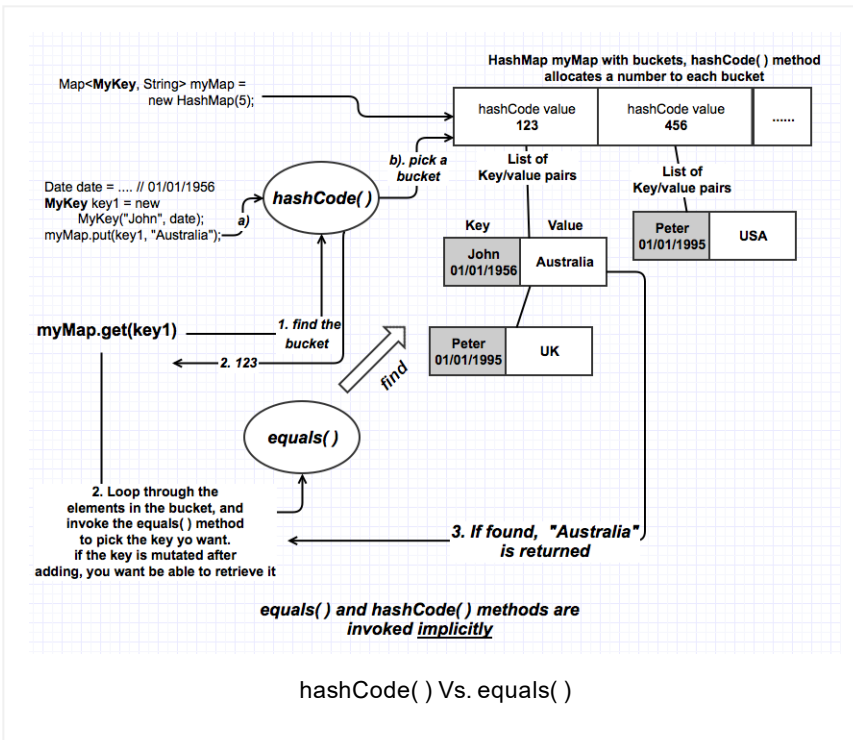
02: HTTP basics

03: Servlet inter

```

10      this.myDate = new Date(aDate.getTime( ))
11    }
12
13    public Date getDate( ) {
14        return new Date(myDate.getTime( )); //de
15    }
16
17    public String getName( ) {
18        return name; //String is immutable
19    }
20
21    //...equals( ), hashCode(), etc
22 }
23

```



As shown, when Maps are used in Java, the **`equals()`** and **`hashCode()`** methods are implicitly invoked. If these methods are incorrectly implemented or the keys are modified once added to the map, then unpredictable behavior will be experienced, and these behaviors are harder to debug. The `hashCode()` and `equals()` methods are implicitly invoked to determine where the key is stored, and to retrieve the value for a particular key respectively. More than one key/value pairs can be stored in the same bucket.

The `hashCode()` method does not give a unique value each time. Its duty is to spread out the numbers so that your key value pairs get spread out in multiple buckets. So, always remember this.

[04: JSP overview](#)
[05: Web patterns](#)
[06: ♦ MVC0, MV](#)
[07: When to use](#)
[08: Web.xml inte](#)

[WebService \(11\)](#)

[01: ♥♦ 40+ Java](#)
[02: ♦ 6 Java RE](#)
[03: ♥ JAX-RS hc](#)
[04: 5 JAXB inter](#)
[05: RESTful We](#)
[06: RESTful Wel](#)
[07: HATEOAS R](#)
[08: REST constr](#)
[09: 11 SOAP We](#)
[10: SOAP Web S](#)
[11: ♥ JAX-WS hc](#)

[JPA \(2\)](#)

[10: Spring, Java](#)
[8 JPA interview c](#)

[JTA \(1\)](#)

[JTA interview Q&](#)

[JDBC \(4\)](#)

[♦ 12 FAQ JDBC](#)
[JDBC Overview](#)
[NamedParamete](#)
[Spring, JavaCon](#)

[JMS \(5\)](#)

[♦ 16 FAQ JMS ir](#)
[Configuring JMS](#)
[JMS versus AMC](#)
[Spring JMS with](#)
[Spring JMS with](#)

[JMX \(3\)](#)

[5 JMX and MBe](#)
[Event Driven Pr](#)
[Yammer metrics](#)

[JNDI and LDAP \(1\)](#)

[JNDI and LDAP](#)

[Pressed for time? Jav](#)

[Job Interview Ice B](#)

The `hashCode()` method is used to store the key/value in a bucket, and both the `hashCode()` and `equals()` methods are called to retrieve the stored key/value. If they are implemented inconsistently or the key is mutated, then the stored object cannot be retrieved as the returned values of these methods will vary in between different invocations. To be more specific, the `hashCode()` method is called to determine the key index (aka the bucket) of the array, and the `equals()` method is called to retrieve the exact key from the list of keys belonging to that particular key index (or bucket) as the same bucket will be holding multiple keys linked to multiple values. Remember the contract between these two methods?

“If 2 objects are equal, they must return the same `hashCode()` value, but the reverse is not true. Which means, if 2 objects return the same `hashCode()` value does not mean that those 2 objects are equal()”.

Q6. How would you defensively copy a `Date` field in your immutable class?

A6.

```

1 public final class MyDiary {
2
3     private final Date myDate;
4
5     public MyDiary(Date aDate) {
6         this.myDate = new Date(aDate.getTime());
7     }
8
9     public Date getDate() {
10        return new Date(myDate.getTime()); //de
11    }
12 }
13
14

```

Q7. How will you prevent the caller from adding or removing elements from a collection of pets?

A7.

```

1
2 import java.util.ArrayList;
3 import java.util.Collections;
4 import java.util.List;

```

01: ♦ 15 Ice brea

02: ♥♦ 8 real life

03: ♦10+ Know y

FAQ Core Java Jot

♥♦ Q1-Q10: Top

♦ Q11-Q23: Top

♦ Q24-Q36: Top

♦ Q37-Q42: Top

♦ Q43-Q54: Top

01: ♥♦ 15 Beginr

02: ♥♦ 10+ Java

FAQ JEE Job Inter

♦ 12 FAQ JDBC

♦ 16 FAQ JMS ir

♦ 8 Java EE (aka

♦ Q01-Q28: Top

♦ Q29-Q53: Top

01: ♦ 12 Web ba

06: ♦ MVC0, MV

JavaScript mista

JavaScript Vs Ja

JNDI and LDAP

JSF interview Q

JSON interview

FAQ Java Web Ser

01: ♥♦ 40+ Java

02: ♦ 6 Java RE

05: RESTFul We

06: RESTful Wel

09: 11 SOAP We

Java Application Ar

001A: ♦ 7+ Java

001B: ♦ Java arc

04: ♦ How to go

Hibernate Job Inter

01: ♥♦ 15+ Hiber

01b: ♦ 15+ Hiber

06: Hibernate Fil

8 JPA interview c

Spring Job Intervie

♦ 11 Spring boot

```

5
6 public class PetCage {
7     private final List<Pet> pets;
8
9     public PetCage(List<Pet> pets) {
10         this.pets = Collections.unmodifiableLi
11     }
12
13     public List<Pet> getPets( ) {
14         return pets;
15     }
16 }
17

```

It ensures that you cannot add or remove pets. However, there is no guarantee that the pets are also immutable. To make this instance fully immutable, the Pet instance itself must be immutable or use the decorator pattern as a wrapper around each of the pets to make them also immutable. For example, The **Integer wrapper class provides immutability to mutable primitive int value** . You could also defensively deep copy the list of pets in the constructor and getPets() method.

Q8. How about data that needs to be mutable, but less frequently? Is there any way to obtain the benefits of immutability with the added benefit of thread-safety for data that changes less frequently?

A8. The **Copy-On-Write** collections like *CopyOnWriteArrayList* and *CopyOnWriteArraySet* classes introduced from JSE 5.0 `util.concurrent` package are good examples of how to harness the power of immutability whilst permitting occasional modifications for infrequently changing data. *CopyOnWriteArrayList* behaves much like the *ArrayList* class, except that when the list is modified, instead of mutating the underlying array, a new array is created and the old array is discarded. *CopyOnWriteArrayList* is designed for cases where:

- reads hugely outnumber writes.
- the array is small (or writes are very infrequent).
- the caller genuinely needs the functionality of a list rather than an array.

When you obtain an iterator, which holds a reference to the underlying array, the array referenced by the iterator is

01: ♥♦ 13 Spring
 01b: ♦ 13 Spring
 04 ♦ 17 Spring b
 05: ♦ 9 Spring B
 Java Key Area Ess
 ♦ Design pattern
 ♥ Top 10 causes
 ♥♦ 01: 30+ Writir
 ♦ 12 Java desigr
 ♦ 18 Agile Develo
 ♦ 5 Ways to debi
 ♦ 9 Java Transac
 ♦ Monitoring/Pro
 02: ♥♦ 13 Tips to
 15 Security key :
 4 FAQ Performa
 4 JEE Design Pa
 5 Java Concurr
 6 Scaling your J
 8 Java memory i
 OOP & FP Essenti
 ♦ 30+ FAQ Java
 01: ♦ 19 Java 8 I
 04: ♥♦ Top 6 tips
 Code Quality Job I
 ♦ Ensuring code
 ♦ 5 Java unit tes
 SQL, XML, UML, JSC
 ERD (1)
 ♦ 10 ERD (Entity
 NoSQL (2)
 ♦ 9 Java Transac
 3. Understanding
 Regex (2)
 ♥♦ Regular Expr
 Regular Express
 SQL (7)
 ♦ 15 Database d
 ♦ 14+ SQL interv
 ♦ 9 SQL scenari
 Auditing databas

effectively immutable and therefore can be traversed without synchronization or risk of concurrent modification. This eliminates the need to either clone the list before traversal or synchronize on the list during traversal. If reads are much more frequent than insertions or removals, which is the case very often, the Copy-on-Write collections and *ConcurrentHashMap* offer better performance and development convenience. The development convenience is provided not needing to deal with synchronization, deep cloning, or “ConcurrentModificationException”. The *ConcurrentModificationException* is generally thrown by an *ArrayList*, *HashSet*, or a *HashMap* implementation when you try to remove an object from a collection while iterating over it.

Q8. Can builder design pattern be used to create immutable objects?

A8. Yes.

Q9. Can you give a builder design pattern example to create immutable objects?

A9. Example: [Builder pattern and immutability in Java.](#)

Popular Posts

♦ [11 Spring boot interview questions & answers](#)

857 views

♦ [Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers](#)

825 views

[18 Java scenarios based interview Questions and Answers](#)

447 views

[001A: ♦ 7+ Java integration styles & patterns interview questions & answers](#)

401 views

♦ [7 Java debugging interview questions & answers](#)

311 views

♦ [10 ERD \(Entity-Relationship Diagrams\) Interview Questions and Answers](#)

302 views

[Deleting records](#)

[SQL Subquery ir](#)

[Transaction man](#)

[UML \(1\)](#)

♦ [12 UML intervi](#)

[JSON \(2\)](#)

[JSON interview \(](#)

[JSON, Jackson,](#)

[XML \(2\)](#)

[XML basics inter](#)

[XML Processing](#)

[XSD \(2\)](#)

[11 FAQ XSD inte](#)

[XSD reuse inter](#)

[YAML \(2\)](#)

[YAML with Java](#)

[YAML with Sprin](#)

[Hadoop & BigData Int](#)

♥ 01: Q1 – Q6 Had

02: Q7 – Q15 Hada

03: Q16 – Q25 Hada

04: Q27 – Q36 Apa

05: Q37 – Q50 Apa

05: Q37-Q41 – Dat

06: Q51 – Q61 HB

07: Q62 – Q70 HDI

[Java Architecture Inte](#)

♥♦ 01: 30+ Writing

001A: ♦ 7+ Java int

001B: ♦ Java archi

01: ♥♦ 40+ Java W

02: ♥♦ 13 Tips to w

03: ♦ What should l

04: ♦ How to go ab

05: ETL architectur

1. Asynchronous pi

2. Asynchronous pi

[Scala Interview Q&As](#)

01: ♥ Q1 – Q6 Scal

02: Q6 – Q12 Scal

03: Q13 – Q18 Sca

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

292 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

286 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

263 views

8 Git Source control system interview questions & answers

215 views

Bio

Latest Posts



Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.



About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.

◀ Java serialization, cloning, and casting interview Q&A

04: Q19 – Q26 Sca

05: Q27 – Q32 Sca

06: Q33 – Q40 Sca

07: Q41 – Q48 Sca

08: Q49 – Q58 Sca

09: Q59 – Q65 Hig

10: Q66 – Q70 Pat

11: Q71 – Q77 – S

12: Q78 – Q80 Rec

Spring, Hibernate, & I

Spring (18)

Spring boot (4)

♦ 11 Spring bc

01: Simple Sp

02: Simple Sp

03: Spring box

Spring IO (1)

Spring IO tuto

Spring JavaConf

10: Spring, Ja

Spring, JavaC

Spring, JavaC

Spring, JavaC

01: ♥♦ 13 Spring

01b: ♦ 13 Spring

02: ► Spring DI

03: ♥♦ Spring DI

04 ♦ 17 Spring b

05: ♦ 9 Spring B

06: ♥ Debugging

07: Debugging S

Spring loading p

Spring Hibernate (13)

01: ♥♦ 15+ Hiber

01b: ♦ 15+ Hiber

02: Understandir

03: Identifying ar

04: Identifying ar

05: Debugging H

06: Hibernate Fil

07: Hibernate mi

♦ 5 Java Object class methods interview questions & answers >

Posted in member-paid, Objects

Tags: Core Java FAQs, Java/JEE FAQs, Novice FAQs

Leave a Reply

Logged in as geethika. Log out?

Comment

Post Comment

08: Hibernate au
09: Hibernate en
10: Spring, Java
11: Hibernate de
12: Hibernate cu

AngularJS (2)

♥ 8 AngularJS in
More Angular JS

Git & SVN (6)

♥ Git & Maven fc
♥ Merging Vs rel
♥ Understanding
6 more Git interv
8 Git Source cor
Setting up Cygw

JMeter (2)

♥ JMeter for test
♦ JMeter perform

JSF (2)

JSF interview Q&
More JSF intervi

Maven (3)

♥ Git & Maven fc
12 Maven intervi
7 More Maven ir

Testing & Profiling/Sa

Automation Testing

♥ Selenium and

Code Coverage (2)

Jacoco for unit te
Maven and Cobr

Code Quality (2)

♥ 30+ Java Code
♦ Ensuring code

jvisualvm profiling (

01: jvisualvm to :

02: jvisualvm to :

03: jvisualvm to :

Performance Testir

♥ JMeter for test
♦ JMeter perform

Unit Testing Q&A (2)

BDD Testing (4)

Java BDD (Be

jBehave and E

jBehave and j

jBehave with t

Data Access Uni

♥ Unit Testing

Part #3: JPA H

Unit Test Hibe

Unit Test Hibe

JUnit Mockito Sp

JUnit Mockito

Spring Con

Unit Testing

Part 1: Unit te

Part 2: Mockit

Part 3: Mockit

Part 4: Mockit

Part 5: Mockit

Testing Spring T.

Integration Un

Unit testing Sp

♦ 5 Java unit tes

JUnit with Hamc

Spring Boot in ui

Other Interview Q&A 1

Finance Domain In

12+ FX or Forex

15 Banking & fin

FIX Interview Q&A

20+ FIX basics in

Finding your way

Groovy Interview Q

Groovy Coding C

Cash balance

Sum grades C

♥ Q1 – Q5 Groov

♦ 20 Groovy clos

♦ 9 Groovy meta

Groovy method c

- Q6 – Q10 Groov
- JavaScript Interview
- JavaScript Top I
- ♥ Q1 – Q10 J
- ♦ Q11 – Q20
- ♦ Q21 – Q30
- ♦ Q31 – Q37
- JavaScript mis
- JavaScript Vs Ja
- JavaScript Vs
- Unix Interview Q&A
- ♥ 14 Unix intervi
- ♥ Hidden Unix, C
- sed and awk to v
- Shell script inter
- Unix history com
- Unix remoting in
- Unix Sed comm
- Free Java Interview
- ▶ Java Integration
- ▶ Java Beginner I
- 02: ▶ Spring DIP, I
- 06: ▶ Tell me abou

As a Java Architect

[Java architecture & design concepts interview Q&As with diagrams | What should be a typical Java EE architecture?](#)

Senior Java developers must have a good handle on

[open all](#) | [close all](#)

- [Best Practice \(6\)](#)
- [Coding \(26\)](#)
- [Concurrency \(6\)](#)
- [Design Concepts \(7\)](#)
- [Design Patterns \(11\)](#)
- [Exception Handling \(3\)](#)
- [Java Debugging \(21\)](#)
- [Judging Experience \(1\)](#)
- [Low Latency \(7\)](#)
- [Memory Management \(1\)](#)
- [Performance \(13\)](#)
- [QoS \(8\)](#)
- [Scalability \(4\)](#)
- [SDLC \(6\)](#)
- [Security \(13\)](#)
- [Transaction Management \(1\)](#)

80+ step by step Java Tutorials

[open all](#) | [close all](#)

- [Setting up Tutorial \(6\)](#)
- [Tutorial - Diagnosis \(2\)](#)
- [Akka Tutorial \(9\)](#)
- [Core Java Tutorials \(2\)](#)
- [Hadoop & Spark Tutorials \(1\)](#)
- [JEE Tutorials \(19\)](#)
- [Scala Tutorials \(1\)](#)
- [Spring & Hibernate Tutorials \(1\)](#)
- [Tools Tutorials \(19\)](#)
- [Other Tutorials \(45\)](#)

Preparing for Java written & coding tests

open all | close all

- ✚ ♦ Complete the given
- ✚ Can you write code? |
- ✚ Converting from A to I
- ✚ Designing your classe
- ✚ Java Data Structures
- ✚ Passing the unit tests
- ✚ What is wrong with th
- ✚ Writing Code Home A
- ✚ Written Test Core Jav
- ✚ Written Test JEE (1)

How good are your...to go places?

open all | close all

- ✚ Career Making Know-
- ✚ Job Hunting & Resum

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ [Java generics in no time](#) ☀ [Top 6 tips to transforming your thinking from OOP to FP](#) ☀ [How does a HashMap internally work? What is a hashing function?](#)
 ☀ [10+ Java String class interview Q&As](#) ☀ [Java auto un/boxing benefits & caveats](#) ☀ [Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect](#)

Non-Technical Know Hows

☀ [6 Aspects that can motivate you to fast-track your career & go places](#) ☀ [Are you reinventing yourself as a Java developer?](#) ☀ [8 tips to safeguard your Java career against offshoring](#) ☀ [My top 5 career mistakes](#)

Prepare to succeed

☀ [Turn readers of your Java CV go from “Blah blah” to “Wow”?](#) ☀ [How to prepare for Java job interviews?](#) ☀ [16 Technical Key Areas](#) ☀ [How to choose from multiple Java job offers?](#)

Select Category ▼

© Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.