

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 identifiers](#)

Java identifiers

Posted on [October 1, 2014](#) by [Arulkumaran Kumaraswamipillai](#) — [No Comments](#)

↓



Tweet



Q1. Which of these are legal Identifiers in Java?

- a) \$Ident1
- b) _Ident1
- c) -Ident1
- d) 2Ident1
- e) private
- f) private1
- g) null
- h) Ident-1
- i) Ident\$1
- j) \u00A3Ident1
- k) \u00A5Ident1

A1. The legal identifiers in Java are are a,b, f, i, j, and k.

[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 breake
- 02: ♥♦ 8 real life ex
- 03: ♦10+ Know you
- 04: Can you think c
- 05: ♥ What job inte
- 06: ► Tell me abou
- 07: ♥ 20+ Pre inter

[Core Java Interview C](#)

[Java Overview \(4\)](#)

01: ♦ ♥ 17 Java c

Identifiers are names you give to your variables, constants, classes, interfaces and methods. A valid identifier, complying with the following rules:

- The first character of an identifier must be a letter, an underscore(_), or a currency sign(e.g. \$).
- The rest of the characters in the identifier can be a letter, underscore, currency sign, or digit. Note that spaces are NOT allowed in identifiers.
- Identifiers are case-sensitive. This means that age and Age are considered as different identifiers.
- Identifiers cannot match any of Java's reserved words like for, int, etc or literals like null, true, and false.

Q2. Can you write a sample code in Java that determines if a given string is a reserved keyword or not?

A2.

```

1 import java.util.Arrays;
2 import java.util.HashSet;
3 import java.util.List;
4 import java.util.Set;
5
6 public final class ValidIdentifiers {
7
8     private enum Validity {
9         Valid, InvalidIdentifierStart, InvalidId
10    };
11
12    private static final String[ ] RESERVED_KEYW
13        "continue", "for", "new", "switch",
14        "if", "package", "synchronized", "bo
15        "private", "this", "break", "double"
16        "protected", "throw", "byte", "else"
17        "throws", "case", "enum", "instanceo
18        "transient", "catch", "extends", "in
19        "char", "final", "interface", "stati
20        "finally", "long", "strictfp", "vola
21        "float", "native", "super", "while"
22
23    private static final String[ ] RESERVED_LITE
24
25    private static Set<String> KEYWORDS = new Ha
26        (int)(RESERVED_KEYWORDS.length/0.75)
27
28    private static Set<String> LITERALS = new Ha
29        (int)(RESERVED_LITERALS.length/0.75)
30
31    static {
32        List<String> list = Arrays.asList(RESERV
33        KEYWORDS = new HashSet<String>(list);
34
35        List<String> listLit = Arrays.asList(RES

```

02: ♥♦ Java Con

03: ♦ 9 Core Jav

04: ♦ Top 10 mos

☰ Data types (6)

01: Java data typ

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

```

36     LITERALS = new HashSet<String>(listLit);
37 }
38
39 public static final Validity valid(String in
40     if (input.length() == 0
41         || !Character.isJavaIdentifierSt
42         return Validity.InvalidIdentifierSta
43     }
44
45     for (int i = 1; i < input.length(); i++
46         if (!Character.isJavaIdentifierPart(
47             return Validity.InvalidIdentie
48         }
49     }
50
51     if (KEYWORDS.contains(input)) {
52         return Validity.ReservedKeyWord;
53     }
54
55     if (LITERALS.contains(input)) {
56         return Validity.ReservedLiteral;
57     }
58     return Validity.Valid;
59 }
60 }

```

Note: true, false, and null might seem like keywords, but they are actually literals; you cannot use them as identifiers in your programs.

Note: **const** and **goto** are reserved, but not currently used. **enum** was added in Java 5. **strictfp** allows you to have more predictable control over floating-point arithmetic.

Q3. Can you talk us through the code highlighting some of the key language and design features?

A3.

- Use of enums and generics indicates that this code must be using JDK version 1.5 or greater.
- **Private** access modifiers are used where required to encapsulate the internal details.
- The class is marked final so that it cannot be extended.
- It follows the “code to interface” design principle. For example, the Set, List, etc shown below are interfaces.

```

1 private static Set<String> KEYWORDS = new H
2 //...

```

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 Beginr

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

```
3 List<String> list = Arrays.asList(REERVED_
4
```

- Making use of the **Java API methods** where possible. For example, the methods shown below from the Character and Arrays classes, simplify your code. So don't memorize the Java API, but keep it handy and constantly refer to it.

```
1 Character.isJavaIdentifierStart(input.char
2 Character.isJavaIdentifierPart(input.charA
3 List<String> list = Arrays.asList(REERVED_
```

- If a size of a collection is known in advance, it is a best practice to set its initial size appropriately to prevent any resizing. Implementing the code as shown below would not quite work.

```
1 private static Set<String> KEYWORDS = new H
```

The internal threshold for HashSets and HashMaps are calculated as (int) (capacity * loadFactor). The default loadFactor is 0.75. This means the HashSet will resize after 75% of its capacity has been reached. The resizing and rehashing of the set can be prevented as follows,

```
1 private static Set<String> KEYWORDS = new H
```

- Checking for null and empty string as a precondition in the beginning of the valid(String input) method.

```
1 if (input == null || input.length( ) == 0)
```

The above code snippet is a slight deviation of the **fail fast principle**, which states that check and report any possible failures as soon as possible. Testing your code for failure points will lead to better, safer, and more reliable code.

Searching algori

Swapping, partiti

Annotations (2)

8 Java Annotati

More Java annot

Collection and Data

Find the first ne

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

Q4. Do you have any recommendations to further improve the code?

A4. Yes.

1) The **Apache commons library** class *StringUtils* can be introduced here. The method *isEmpty (String input)* can be used to check for both null and empty string. This library does have other useful methods that can be used elsewhere in the application code to enforce the fail fast principle.

2) The `RESERVED_KEYWORDS` and `RESERVED_LITERALS` constants may be loaded from a configuration file. This will ensure that if new keywords or literals are added to Java in the future, it will require only a configuration change.

Q5. What is an escape character? Can you list some character escape codes in Java? What are the differences between decimal, octal, and hexadecimal literals?

A5. The **escape character**, the back slash `\`, is the character that signals the following character is not what it normally means. Java provides escape sequences for several non-graphical characters. All characters can be specified as a hexadecimal Unicode character (`\u0041`) with some as an octal character (`\ddd` where the first d is limited to 0-3, and the others 0-7).

```
1 char c1 = 'A';           //Prints 'A',
2 char c2 = '\n';         //Prints new
3 char c3 = '\u0041';     //Prints 'A',
4 char c4 = 65;           //Prints 'A'
5
```

Q6. How does Java treat Unicode escapes within string literals?

A6.

```
1 System.out.println("\u00A3-Pound, \u00A5-Yen");
```

Q7. Why is it a best practice to adhere to naming conventions when naming the variables, classes, interfaces,

[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](#)

[02: jvisualvm to](#)

[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](#)

etc?

A7. Naming conventions make programs more understandable by making them easier to read. They can also give information about the function of the identifier. For example, whether it's a constant, package, or class.

- For classes and interfaces, use **UpperCamelCase**.
- For class members and local variables use **lowerCamelCase**
- For packages, use reverse URI, e.g.
org.project.subsystem
- For constants, use ALL_CAPS.
-

If you really want to begin a variable name with a digit, prefix the name you'd like to have (e.g. 9pins) with an underscore, e.g. _9pins. Otherwise use something like ninePins.

Some try to differentiate between member variables, local variables, and arguments with different patterns. For example,

- field (aka instance or member variable): **_name** or **this.name**. "this" is a reserved keyword in Java referring to the current object.
- argument : **aName**
- parameter : **pName**
- local variable : **_name**

There is no real objective reason to prefer one style over the other as long as it is consistent across the team or organization. **Checkstyle** is an open-source tool that can help enforce coding standards and best practices.

Popular Posts

♦ 11 Spring boot interview questions & answers

857 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

825 views

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

11: ♥ JAX-WS hc

JPA (2)

10: Spring, Java

8 JPA interview

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 AM

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

18 Java scenarios based interview Questions and Answers

447 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

400 views

♦ 7 Java debugging interview questions & answers

311 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

301 views

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](#)

01: ♦ 15 Ice breaker

02: ♥♦ 8 real life

03: ♦10+ Know y

FAQ Core Java Job

♥♦ Q1-Q10: Top

♦ Q11-Q23: Top

♦ Q24-Q36: Top

♦ Q37-Q42: Top

♦ Q43-Q54: Top

01: ♥♦ 15 Beginn

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 Interview

♦ 11 Spring boot



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](https://www.amazon.com) in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.

◀ Flyweight pattern and improve memory usage & performance

01: Java data types interview Q&A ▶

Posted in member-paid, Reserved Key Words

Leave a Reply

Logged in as geethika. [Log out?](#)

Comment

Post Comment

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 Devel
♦ 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

	Deleting records
	SQL Subquery in
	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 Hadc
	03: Q16 – Q25 Hac
	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 archit
	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

- 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 JavaConl
 - 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
 - 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

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 u

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 Ja
- ♦ 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 In](#)
- [Low Latency \(7\)](#)
- [Memory Management](#)
- [Performance \(13\)](#)
- [QoS \(8\)](#)
- [Scalability \(4\)](#)
- [SDLC \(6\)](#)
- [Security \(13\)](#)
- [Transaction Managen](#)

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 Tuto](#)
- [JEE Tutorials \(19\)](#)
- [Scala Tutorials \(1\)](#)
- [Spring & Hibernate Ti](#)
- [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.