

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](#) › [Interview](#) › [Core Java Interview Q&A](#) › [Data types](#) › 03: ♦ ♥ Java autoboxing & unboxing – benefits & caveats interview Q&A

03: ♦ ♥ Java autoboxing & unboxing – benefits & caveats interview Q&A

Posted on [June 29, 2015](#) by [Arulkumaran Kumaraswamipillai](#)

12

Like

Share

3

G+1

Q1. What do you understand by the terms “autoboxing” and “autounboxing” in Java?

A1. Java automatically converts a primitive type like “int” into corresponding wrapper object class Integer. This is known as the **autoboxing**. When it converts a wrapper object class Integer back to its primitive type “int”, it is known as “**autounboxing**”.








Example 1:

[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 Java](#)
-  [05: ♥ What job interviews](#)
-  [06: ► Tell me about Java](#)
-  [07: ♥ 20+ Pre interview](#)

 [Core Java Interview Questions](#)

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

 [01: ♦ ♥ 17 Java Interview](#)

```

1 package com.autoboxing;
2
3 public class AutoBoxUnbox {
4
5     public static void main(String[] args) {
6
7         int i = 5;
8         Integer objI = i; //autoboxing takes place
9
10        if(objI != null)
11        {
12            int result = objI + 3; //auto unboxing
13            System.out.println(result);
14        }
15    }
16 }

```

This can be applied to one of 8 primitives in Java to convert from primitive to wrapper via autoboxing and from wrapper to primitive via autounboxing. Autoboxing and unboxing can happen anywhere where an object is expected and primitive type is available

Example 2:

```

1 package com.autoboxing;
2
3 import java.util.ArrayList;
4 import java.util.HashMap;
5 import java.util.List;
6 import java.util.Map;
7
8 public class AutoBoxUnbox {
9
10    public static void main(String[] args) {
11
12        List<Character> characters = new ArrayList();
13        characters.add('C'); //autoboxed to Character
14
15        Map<Long, Double> myMap = new HashMap<>();
16        myMap.put(5L, 12.50); //autoboxed to Long and Double
17
18        char myChar = characters.get(0); //unboxing
19        System.out.println(myChar);
20
21        double myDouble = myMap.get(5L); //unboxing
22        System.out.println(myDouble);
23    }
24
25 }

```

Q2. What are the benefits of autoboxing?

A2. Less code to write, and the code looks cleaner.

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 load

♦ Java classes a

☐ Objects (8)

► Beginner Jav

♥♦ HashMap & H

♦ 5 Java Object i

♦ 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 a

Understanding C

What are good r

☐ GC (2)

♦ Java Garbage

For example, you don't have to do as shown below:

```
1 list.add(Integer.valueOf(6));
```

More readable with autoboxing

```
1 list.add(6);
```

Q3. What are some of the pitfalls of autoboxing?

A3. It is very convenient to have autoboxing, but it can cause issues and many beginners fall into its caveats.

1. Unnecessary Object creation due to Autoboxing

```
1 package com.autoboxing;
2
3 public class AutoBoxUnbox {
4
5     public static void main(String[] args) throws Exception {
6
7         Integer sum = 0;
8         for (int i = 1000; i < 500000; i++) {
9             sum += i;
10            Thread.sleep(100);
11        }
12    }
13 }
```

Q. How do you know unnecessary objects are being created?

A. jmap to the rescue.

Step 1: Run the above code.

Step 2: Open a DOS or Unix command prompt and run the following commands. "jps" to find the process id, and then "jmap" to print the object graph

```
1 C:\>jps
2 8148
3 8420 Jps
4 3832 JConsole
5 8896 AutoBoxUnbox
6 10300 JConsole
7 10948 JConsole
```

03: Java GC tuning

Generics (5)

♥ Java Generics

♥ Overloaded methods

♦ 12 Java Generics

♦ 7 rules to remember

3 scenarios to go

FP (8)

01: ♦ 19 Java 8 FP

02: ♦ Java 8 Stream

03: ♦ Functional

04: ♥♦ Top 6 tips

05: ♥ 7 Java FP

Fibonacci number

Java 8 String stream

Java 8: What is a

IO (7)

♥ Reading a text

♦ 15 Java old I/O

06: ♥ Java 8 way

Processing large

Processing large

Read a text file f

Reloading config

Multithreading (12)

01: ♥♦ 15 Beginner

02: ♥♦ 10+ Java

03: ♦ More Java

04: ♦ 6 popular J

05: ♦ How a thread

06: ♦ 10+ Atomic

07: 5 Basic multi

08: ♦ ThreadLocal

09: Java FutureTask

10: ♦ ExecutorService

Java ExecutorService

Producer and Consumer

Algorithms (5)

♦ Splitting input text

♦ Tree traversal

♥♦ Java coding

```

8
9 C:\>jmap -histo:live 8896 > mem.txt
10
11

```

Step 3: Inspect the mem.txt file

```

1
2 num      #instances      #bytes  class name
3 -----
4
5 7:        1318           21088  java.lang.Int
6

```

after some time

```

1
2 num      #instances      #bytes  class name
3 -----
4
5 7:        1704           27264  java.lang.Int
6

```

You can see the growing instances and bytes.

Now try the something after fixing the code as shown below.

```

1 package com.autoboxing;
2
3 public class AutoBoxUnbox {
4
5     public static void main(String[] args) throw
6
7         int sum = 0; //FIX. change to primitive
8         for (int i = 1000; i < 500000; i++) {
9             sum += i;
10            Thread.sleep(100);
11        }
12    }
13 }

```

```

1
2 num      #instances      #bytes  class name
3 -----
4
5 8:        256            4096   java.lang.Int
6

```

after some time

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: Understandir

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

1				
2	num	#instances	#bytes	class name
3	-----			
4	8:	256	4096	java.lang.Integer
5				
6				

The improved code does not create unnecessary Integer objects. You may also like the detailed “[javap, jps, jmap, and jvisualvm tutorial – analyzing the heap histogram](#)”

2. GC overhead

Unnecessarily creating too many objects and then discarding them will increase the Garbage Collection overhead. This may cause performance impact due to more frequent garbage collection.

3. java.lang.NullPointerException

Especially when mixing object and primitive in equality and relational operator.

```

1 package com.autoboxing;
2
3 public class AutoBoxUnbox {
4
5     public static void main(String[] args) throw
6         Integer i = null;
7
8     if(i > 6) { // tries to do i.intValue();
9         System.out.println("I am in here");
10    }
11 }
12 }
```

Conditional operators can cause **NullPointerException**.

```

1 package com.autoboxing;
2
3 public class AutoBoxUnbox {
4
5     public static void main(String[] args) throw
6         boolean b = false;
7         double d1 = 0d;
8         Double d2 = null;
9         Double d = b ? d1 : d2; //NullPointerException
10    }
11 }
12 }
```

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 Programn

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

Since d1 is primitive, d2 is implicitly tried to auto unbox. To fix it, you need to change d1 to wrapper object type **"Double"**. This way auto unboxing won't take place.

4. Overloading

Q. What will be the output of the following code?

```

1 package com.autoboxing;
2
3 public class AutoBoxUnbox {
4
5     public static void main(String[] args) throw
6         Integer value = 0;
7         new AutoBoxUnbox().eval(value);
8     }
9
10    void eval(long val) {
11        System.out.println(1);
12    }
13
14    void eval(Long value) {
15        System.out.println(2);
16    }
17 }
```

A. The result is 1, because there is no direct conversion from Integer to Long, so the "conversion" from Integer to long is used.

Q4. How will you go about debugging auto boxing and unboxing error?

A4.

1) Being aware of the potential auto boxing and unboxing caveats discussed above.

2) Configuring your IDE to pick up auto boxing and unboxing error. For example, in **eclipse**

```
1 Java --> Compiler --> Errors/Warnings --> "Potent
```

Popular Posts

♦ 11 Spring boot interview questions & answers

[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 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](#)

856 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

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.



01: ♦ 15 Ice breake

02: ♥♦ 8 real life

03: ♦10+ Know y

FAQ Core Java J

♥♦ 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



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 Generics in no time “? extends” & “? super” explained with a diagram

javap, jps, jmap, and jvisualvm tutorial – analyzing the heap

histogram ▶

Posted in Data types

Tags: Free Content

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

	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 HBa
	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

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
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 in](#)

[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 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.