

# 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](#) › 02: ♥♦ 10 Java String class interview questions & answers

## 02: ♥♦ 10 Java String class interview questions & answers

Posted on [August 10, 2014](#) by [Arulkumaran Kumaraswamipillai](#) — 4 Comments



15

Like

4

16

Share

G+1

Share

**Q1.** What will be the output of the following code snippet?

```
1 String s = " Hello ";
2 s += " World ";
3 s.trim();
4 System.out.println(s);
```

**A1.** The output will be

```
1 " Hello World "
```

[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

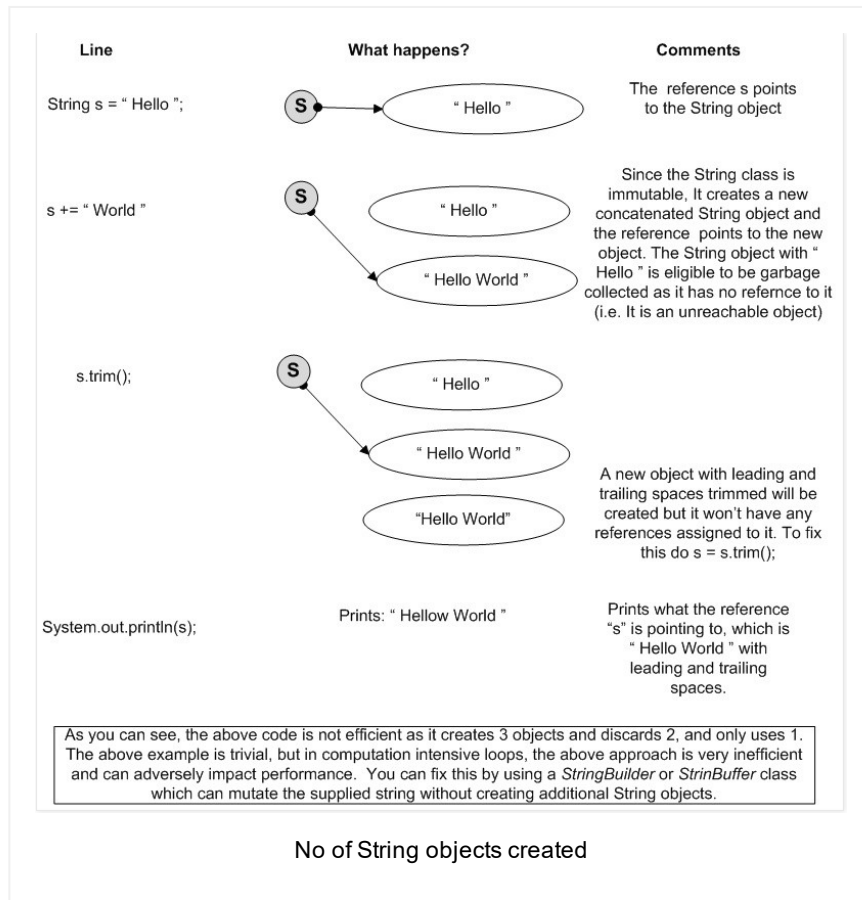
with the **leading and trailing spaces**. Some would expect a trimmed "Hello World". **So, what concepts does this question try to test?**

1. String objects are immutable and there is a trick in `s.trim()` line.
2. Concept of object references and unreachable objects that are eligible for garbage collection. 3 String objects are created, and 2 of them become unreachable as there are no references to them, and gets garbage collected.

### What follow on questions can you expect?

1. You might get a follow on question on how many string objects are created in the above example and when will it become an unreachable object to be garbage collected.
2. You might also be asked a follow on question as to if the above code snippet is efficient.

The best way to explain this is via a self-explanatory diagram as shown below. Click on it to enlarge.



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 i

♦ Java enum inte

♦ Java immutabl

♥♦ Object equals

Java serialization

Mocks, stubs, dc

#### OOP (10)

♥ Design princip

♦ 30+ FAQ Java

♦ Why favor com

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

If you want the above code to output "Hello World" with leading and trailing spaces trimmed then assign the `s.trim()` to the variable "s". This will make the reference "s" to now point to the newly created trimmed String object.

The above code can be rewritten as shown below

```
1 StringBuilder sb = new StringBuilder(" Hello ");
2 sb.append(" World ");
3 System.out.println(sb.toString().trim());
4
```

**Q2.** What is the main difference between *String*, *StringBuffer*, and *StringBuilder*?

**A2.**

- **String** is **immutable** in Java, and this immutability gives the benefits like security and performance discussed above.
- **StringBuffer** is **mutable**, hence you can add strings to it, and when required, convert to an immutable String with the `toString()` method.
- **StringBuilder** is very similar to a *StringBuffer*, but *StringBuffer* has one disadvantage in terms of performance as all of its public methods are synchronized for thread-safety. *StringBuilder* in Java is a copy of *StringBuffer* but without synchronization to be used in local variables which are inherently thread-safe. So, if thread-safety is required, use *StringBuffer*, otherwise use *StringBuilder*.

**Q3.** Can you write a method that reverses a given String?

**A3.** A popular Java interview coding question.

**Example 1:** It is always a best practice to reuse the API methods as shown below with the `StringBuilder(input).reverse()` method as it is fast, efficient (uses bit wise operations) and knows how to handle Unicode surrogate pairs, which most other solutions ignore. The code shown below handles null and empty strings, and a *StringBuilder* is used as opposed to a thread-safe

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

StringBuffer, as the StringBuilder is locally defined, and local variables are implicitly thread-safe.

```

1
2 public static String reverse(String input) {
3
4     if(input == null || input.length() == 0){
5         return input;
6     }
7
8     return new StringBuilder(input).reverse().toString();
9
10 }
11

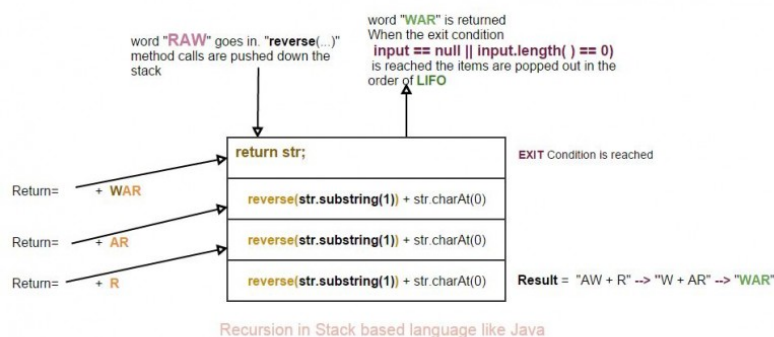
```

**Example 2:** Some interviewers might probe you to write other lesser elegant code using either recursion or iterative swapping. Some developers find it very difficult to handle recursion, especially to work out the termination condition. All recursive methods need to have a condition to terminate the recursion. **Recursive** solution.

```

1 public String reverse(String str) {
2     // exit or termination condition
3     if ((null == str) || (str.length() <= 1))
4         return str;
5 }
6
7 // put the first character (i.e. charAt(0))
8 // and recurse with 2nd character (i.e. substring(1))
9 return reverse(str.substring(1)) + str.charAt(0);
10 }
11

```



Java Recursion – String example

### Step 1: reverse("RAW")

Searching algorithm

Swapping, partitioning

Annotations (2)

8 Java Annotations

More Java annotations

Collection and Data Structures

Find the first non-repeating character in a string

Java Collections Framework

Java Iterable and Iterator

HashMap & HashSet

Sorting objects

02: Java 8 Stream

04: Understanding

4 Java Collections

If Java did not have

Java 8: Different

Part-3: Java Tree

Sorting a Map by

When to use which

Differences Between

Java Iterable and

Multithreading

Why do Proxy,

Core Java Modifiers

Differences between

Java Collections and

Event Driven Programming

Event Driven Programming

Event Driven Programming

Exceptions (2)

Java exception

Top 5 Core Java

Java 7 (2)

Java 7 fork and join

Java 7: Top 8 new

Java 8 (24)

01: 19 Java 8 Interview

02: Java 8 Stream

03: Functional

04: Top 6 tips

04: Convert Lists

**Step 2: reverse(AW) + "R"** [Note: charAt[0] = "R", and str.substring(1) = "AW" ]

**Step 3: reverse(W) + "A" + "R"** [Note: charAt[0] = "A", and str.substring(1) = "W" ]

**Step 4: return "W" + "A" + "R"** [Exit condition is reached when "str.length( ) <=1" ]

**outputs: "WAR"**

**Example 3: Iterative solution.**

```

1 public String reverse(String str) {
2     // validate
3     if ((null == str) || (str.length( ) <= 1))
4         return str;
5 }
6
7 char[ ] chars = str.toCharArray( );
8 int rhsIdx = chars.length - 1;
9
10 //iteratively swap until exit condition lhsI
11 for (int lhsIdx = 0; lhsIdx < rhsIdx; lhs
12     char temp = chars[lhsIdx];
13     chars[lhsIdx] = chars[rhsIdx];
14     chars[rhsIdx--] = temp;
15 }
16
17 return new String(chars);
18 }
19

```

**Q4.** Can you remember a design pattern discussed in this post?

**A4. Flyweight design pattern.** The flyweight design pattern is a structural pattern used to improve **memory usage** (i.e. due to fewer objects and object reuse) and **performance** (i.e. due to shorter and less frequent garbage collections).

**Q5.** Can you give some examples of the usage of the flyweight design pattern in Java?

**A5.**

**Example 1:** As discussed above, String objects are managed as flyweight. Java puts all fixed String literals into a literal

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 MBea

☞ 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

pool. For redundant literals, Java keeps only one copy in the pool.

```

1 String author = "Little brown fox";
2 String authorCopy = "Little brown fox";
3
4 //only 1 String object is created. Both author and authorCopy
5 if(author == authorCopy) {
6     System.out.println("referencing the same object");
7 }
8

```

**Example 2:** The Wrapper classes like *Integer*, *Float*, *Decimal*, *Boolean*, and many other classes like *BigDecimal* having the *valueOf* static factory method to apply the flyweight design pattern to conserve memory by reusing the objects.

```

1 public class FlyWeightWrapper {
2
3     public static void main(String[] args) {
4         Integer value1 = Integer.valueOf(5);
5         Integer value2 = Integer.valueOf(5);
6
7         //only one object is created
8         if (value1 == value2) {
9             System.out.println("referencing the same object");
10        }
11    }
12 }
13
14 }
15

```

If you use `new Integer(5)`, a new object will be created every time.

Both the above examples will print “**referencing the same object**”.

**Q6.** What is a static factory method, and when will you use it?

**A6.** The factory method pattern is a way to encapsulate object creation. It has the benefits like

1. Factory can choose what to return from many subclasses or implementations of an interface. This allows the caller to specify the behavior desired via parameters, without having

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



to know or understand a potentially complex class hierarchy. The lesser a caller knows about a callee's internal details, the more **loosely coupled** a callee is from the caller.

2. The factory can apply the fly weight design pattern to **cache objects** and return cached objects instead of creating a new object every time. In other words, objects can be pooled and reused. This is the reason why you should favor using *Integer.valueOf(6)* as opposed to *new Integer(6)*.

3. The factory methods have **more meaningful names** than the constructors. For example, *getInstance()*, *valueOf()*, *getConnection()*, *deepCopy()*, etc.

```

1 public static List<Car> deepCopy(List<Car> listC
2     List<Car> copiedList = new ArrayList<Car>(1
3     for (Car car : listCars) {
4         Car carCopied = new Car( );
5         carCopied.setColor((car.getColor( ));
6         copiedList.add(carCopied);
7     }
8     return copiedList;
9 }
10

```

**Q7.** How will you split the following string of text into individual vehicle types?

"Car,Jeep, Wagon Scooter Truck, Van"

**A7.** Regular expressions to the rescue.

```

1 public class String3 {
2     public static void main(String[ ] args) {
3
4         String pattern = "[,\\s]+"; //regex pat
5
6         String vehicles = "Car,Jeep, Wagon Sco
7         String[ ] result = vehicles.split(patter
8         for (String vehicle : result) {
9             System.out.println("Vehicle = \"" +
10         }
11     }
12 }
13

```

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

**Q8.** What are the different ways to concatenate strings? and which approach is most efficient?

**A8.**

**Plus (“+”) operator:**

```
1 String s1 = "John" + "Davies";
2
```

Using a **StringBuilder** or **StringBuffer** class.

```
1 StringBuilder sb = new StringBuilder("John");
2 sb.append("Davies");
3
```

Using the **concat(...)** method.

```
1 "John".concat("Davies");
2
```

The efficiency depends on what you are concatenating and how you are concatenating it.

Concatenating constants: Plus operator is more efficient than the other two as the JVM optimizes constants.

```
1 String s1 = "John" + "Davies";
2
```

Concatenating String variables: Any one of the three methods should do the job.

```
1 String s1 = s2 + s3 + s4;
2 String s1 = "name=";
3 s1 += name;
4
```

Concatenating in a for/while loop: **StringBuilder** or **StringBuffer** is the most efficient. Avoid using plus operator as it is the worst offender.

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



```

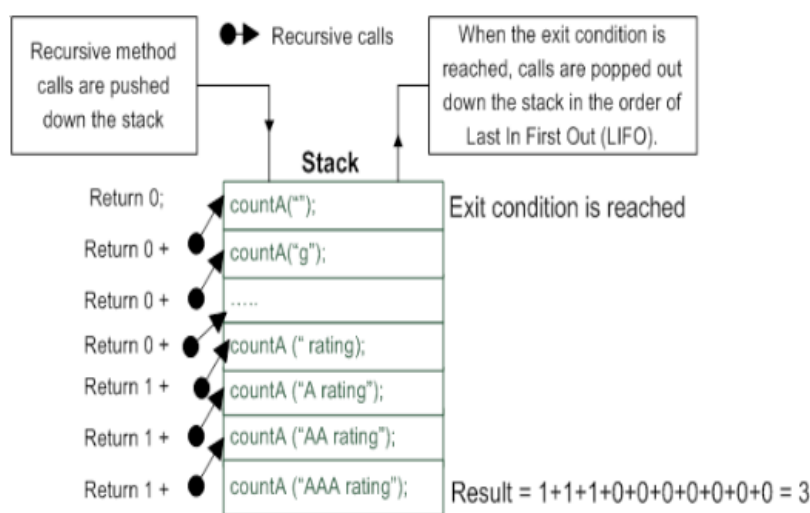
1 StringBuilder sb = new StringBuilder(250);
2 for( int i=0; i<SIZE; i++ ) {
3     sb.append("Item:" + i);
4 }
5

```

Prefer StringBuilder to StringBuffer unless multiple threads can have access to it.

**Q9.** Java being a stack based language, allows you to make recursive method calls. Can you write a recursion based solution to count the number of A's in string "AAA rating"?

**A9.** A function is recursive if it calls itself. Given enough stack space, recursive method calls are perfectly valid in Java though it is tough to debug. Recursive functions are useful in removing iterations from many sorts of algorithms.



Recursion in stack based language like Java

```

1 public class RecursiveCall {
2
3     public int countA(String input) {
4
5         // exit condition - recursive calls must
6         if (input == null || input.length() == 0) {
7             return 0;
8         }
9
10        int count = 0;
11
12        //check first character of the input
13        if (input.substring(0, 1).equals("A")) {
14            count = 1;
15        }
16    }
17 }

```

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

```

16
17         //recursive call to evaluate rest of the
18         //(i.e. 2nd character onwards)
19         return count + countA(input.substring(1)
20     }
21
22     public static void main(String[] args) {
23         System.out.println(new RecursiveCall( ).
24     }
25 }
26

```

Recursion might not be the efficient way to code, but recursive functions are shorter, simpler, and easier to read and understand. Recursive functions are very handy in working with tree structures and avoiding unsightly nested for loops.

## Bonus Java String Q&A

**Q10.** How do you stream a string class in Java 8? ★ ⓘ

**A10.** `chars()` method.

```

1 public static void main(String[] args) {
2     "cactus".chars().forEach(c -> System.out.prin
3 }

```

**Q.** Does parallel processing as shown below preserve the order?

```

1 public static void main(String[] args) {
2     "cactus".chars().parallel().forEach(c -> Syst
3 }

```

**A.** No.

## You may also like

[17 Java overview interview Q&A](#) | [Web Services interview Q&A](#) | [Java EE Overview interview Q&A](#) | [Multithreading scenarios in Java applications interview Q&A](#)

## Popular Posts

[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](#)

## ♦ 11 Spring boot interview questions & answers

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](http://Amazon.com) in 2005, and sold 35,000+ copies. Books are



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 &amp; 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&amp;

More JSF intervi

### Maven (3)

♥ Git &amp; 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

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

♥♦ 6 Java Modifiers every interviewer seems to like >

Posted in Data types

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

## 4 comments on “02: ♥♦ 10 Java String class interview questions & answers”



markad krushna says:

September 16, 2016 at 3:04 pm

it will be helpfull quitions for a stundent like mi.....thanks

Reply



Raghavan alias Saravanan Muthu says:

January 19, 2016 at 4:57 pm

Hi Arul,

A nice post. I have a few suggestions.

Q6.3 Iterative solution uses a char variable created everytime inside a for loop. Though it is not a big

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 C

Groovy Coding C

Cash balance

Sum grades C

♥ Q1 – Q5 Groov

♦ 20 Groovy clos

♦ 9 Groovy meta

Groovy method c

overhead, it can still be avoided if it was declared outside the loop. After all, we are talking about memory consumption and hence it would be appropriate.

Q12. Recursion can be used with indexOf() with an exit condition instead of going over every character, which may be a bit more efficient in this situation.

Cheers,  
Raghavan alias Saravanan Muthu

[Reply](#)



**Arulkumaran Kumaraswamipillai** says:

January 20, 2016 at 12:12 pm

Well spotted Raghavan. Handy tips for death by 1000 cuts type of performance issues. I am not going to change the code so that others can understand the concepts you have mentioned.

[Reply](#)



**satish** says:

October 1, 2015 at 2:25 am

excellent explanation. for more interview questions on String refer this link <http://techno-terminal.blogspot.in/2015/09/important-string-interview-questions.html>

[Reply](#)

## Leave a Reply

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

**Comment**

[Q6 – Q10 Groov](#)

[JavaScript Interview](#)

[JavaScript Top I](#)

[♥ Q1 – Q10 J](#)

[♦ Q11 – Q20](#)

[♦ Q21 – Q30](#)

[♦ Q31 – Q37](#)

[JavaScript mix](#)

[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**

Post Comment

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