

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](#) › 01: Java data types interview Q&A

01: Java data types interview Q&A

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



Tweet



Q1. How would you go about choosing the right data types for your application?

A1. Java is what is known as a strongly typed language. This means Java only accepts specific values within specific variables or parameters. Some languages, such as JavaScript, PHP, and Perl are weakly typed languages.

1. Know the data limits to prevent any data overflow

[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

byte → **short** → **char** → **int** → **long** → **float** → **double**
 (1 byte) (2 bytes) (2 bytes) (4 bytes) (8 bytes) (4 bytes) (8 bytes)

Java primitive data types

```
1 int bad = 2100000000; // Close to int max valu
2 long good = 2100000000L;
3
4 long badAgain = 9000000000000000000L; // Close
5 BigInteger goodAgain = BigInteger.valueOf(9000000
6
```

2. Prefer immutable wrapper objects to primitives.

Each primitive data type has a corresponding wrapper class like *Integer*, *Long*, *Character*, *Float*, *Double*, etc. There are 8 primitive variables and as many wrapper objects. In Java 5, additional wrapper classes like *AtomicInteger*, *AtomicLong*, *AtomicBoolean* and *AtomicReference* were introduced to provide atomic operations for addition, increment, and assignment. These additional classes are mutable and cannot be used as a replacement for regular immutable wrapper classes.

Q2. What are wrapper classes, and why do you need them?

A2. The **wrapper classes** are a good example of the **decorator design pattern**. They decorate the primitive values, and enhance their behavior by providing immutability, atomicity, null checking, etc.

1) Wrapper objects can be **initialized to null**. This can't be done with primitives. Many programmers initialize numbers to 0 or -1 to signify default values, but depending on the scenario, this may be incorrect or misleading.

2) Wrapper objects are very useful for optional data. Databases almost always have a significant fraction of their fields as optional (that is, as possibly NULL). In addition, the

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 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

forms submitted in Web applications can contain optional parameters. Both NULL database fields and missing request parameters naturally map to null object references. With primitives, there is no such natural mapping.

3) Wrapper objects will also set the scene for a *NullPointerException* when something is being used incorrectly, which is much more programmer-friendly as it fails fast than some arbitrary exception buried down the line. Preferably, check for null early on in the method and report it immediately where applicable to adhere to the fail fast principle.

4) The wrapper objects are immutable, hence **inherently thread-safe**. Other threads can only read the values set by the thread that initialized this object.

5) When you create wrapper objects, use the `valueOf()` static factory method for efficiency.

```
1 Integer i2 = new Integer(5);           //first approach
2 Integer i1 = Integer.valueOf(5);       //2nd approach
3
```

The second approach is in fact an implementation of the flyweight design pattern.

Q. When to prefer primitives?

A. Primitives are faster to create and use than wrapper objects. Wrapper objects need to be auto-unboxed before use. Thus there is an extra step for the JVM to perform. For example, in order to perform arithmetic on an *Integer*, it must first be converted to an *int* before the arithmetic can be performed. In many business applications this rarely matters unless you were writing something very number-crunching or profiling indicates that the auto-boxing is a performance or memory issue in a particular part of your code as it is executed very frequently.

Anti-pattern: Watch out for premature-optimization anti-pattern where you are tempted to code for a perceived

03: Java GC tuning

Generics (5)

♥ Java Generics

♥ Overloaded methods

♦ 12 Java Generics

♦ 7 rules to remember

3 scenarios to get

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 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

performance gain and sacrificing good design and maintainability.

Q3. When working with floating-point data types, what are some of the key considerations?

A3.

1. Never compare float or double with “==” or != operator

```
1 //endless loop -- don't compare float or double f
2 for (float f = 5f; f != 10.0; f += 0.1) {
3     System.out.println(f);
4 }
```

2. Use long, int, or *BigDecimal* for storing money, and performing monetary calculations.

Floating point data types like float, double, Float, or Double can result in inaccurate results. use either the *BigDecimal* or int/long representing the value in its lowest units like cents.

```
1 private void calculateTotalAccurately1(float uni
2     BigDecimal total = BigDecimal.ZERO;
3     //use the right constructor
4     BigDecimal uc = new BigDecimal(String.va
5     BigDecimal ic= new BigDecimal(String.val
6     total = uc.multiply(ic);
7
8     total = total.setScale(2, RoundingMode.H
9     System.out.println("Total3 --> " + total
10 }
```

Q4. What is your understanding of widening versus narrowing conversions of primitive data types?

A4. Left to right (e.g. byte to short) is a widening conversion and considered safe because there is no chance for data loss. For example, byte has a range between -128 and 127 and short has a wider range between -32768 and 32767. So

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 Betwe

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

when you go from left to right, the data types are implicitly cast by the compiler since it is safe to do so.

byte → **short** → **char** → **int** → **long** → **float** → **double**
 (1 byte) (2 bytes) (2 bytes) (4 bytes) (8 bytes) (4 bytes) (8 bytes)

Java primitive data types

Right to left (e.g. short to byte) is a narrowing conversion and considered unsafe because there is a chance for data loss.

So when you go from right to left, the compiler expects you to explicitly cast the data to clearly state that it is safe to do so. If you do not cast explicitly, you will get a compile-time error.

For example,

```

1 byte b = 0;           // valid values are -128 to 127
2 short s = 0;          // valid values are -32768 to 32767
3 int i = 0;            // valid values are -2147483648 to 2147483647
4 long l = 0L;          // valid values are -9223372036854775808 to 9223372036854775807
5
6 float f = 0.0F;       // valid values are 1.4E-45 to 3.4E+38
7 double d = 0.0;        // valid values are 4.9E-324 to 1.8E+308
8
9 b = 30;               // okay (30 is of type int,
10 b = 128;              // Not okay (128 is of type int)
11
12 s = b;               // okay: short is wider than byte
13 i = s;               // okay: int is wider than short
14 l = i;               // okay: long is wider than int
15
16 /**
17  * compile-time errors
18  */
19 c = s;               // Not okay: type char is unassignable
20 b = i;               // Not okay: type int is wider than byte
21 i = l;               // Not okay: type long is wider than int
22 l = f;               // Not okay: type float is wider than long
23 f = d;               // Not okay: type double is wider than float
24
25 /**
26  * fix above compile-time errors with explicit casts
27  */
28
29 c = (char)s;
30
31 b = (byte) i;
32 i = (int) l;
33 l = (long) f;
34 f = (float) d;
35

```

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 MBe

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

Note: byte and short are signed data types and they cannot be implicitly cast to unsigned char data type even though it is a widening conversion.

Q5. What are the dangers of explicit casting?

A5. Not knowing the MIN and MAX values can result in unexpected results due to loss of data during narrowing.

Trap #1 Be careful when casting explicitly.

```
1 int iWithinByteRange = 125;
2 int iOutsideByteRangeMax = 129;
3
4 byte bGood = (byte) iWithinByteRange;
5 System.out.println("b0kay=" + bGood);
6
7 byte bBad = (byte) iOutsideByteRangeMax;
8 System.out.println("Trap #1 - bBad=" + bBad);
9
```

Trap #2: Simple binary operations apply 'binary numeric promotions'

The **binary numeric promotion** rule automatically casts each operand to the size of the larger operand type. If neither operand is larger, then both are cast to the same type. In byte `b = b + 10`, the value 10 is of type `int` and is the larger operand type compared to `b`, which is of type `byte`, hence will be evaluated as follows:

```
1 byte b = (int)b + 10;
```

Above code throws a compile time error because `b+10` evaluates to an `int`. You need an explicit cast to convert an `int` to `byte` as it is a **narrowing conversion**.

[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](#)

Trap #3: Compound Operators, such as +=, -=, etc contain an explicit cast

In byte `b+=10` compound binary operation, you may be thinking that `b+=10` will be expanded as `b = b + 10`. But it is really not correct. This is because the compound operations will have an **explicit cast** in the converted byte code.

```
1 byte b = (byte) (b + 10);
```

performing a compound assignment operation (e.g. `+=`, `-=`, `*=`, etc). You may be thinking that `b+=10` will be expanded as `b = b + 10`. But it is really not correct. This is because the compound operations will have an explicit cast even though it is not shown in the code.

Q6. Do you think the following code will throw a compile-time exception? If yes, how will you fix it?

```
1 float myVal = (float)3.0/2.0;
```

A6. Yes. It is cast first and then divided as casting operator has precedence over division operator as per the precedence table. So the above code is equivalent to

```
1 float myVal = 3.0f/2.0;    // float divided by d
2                             //as per the "binary
3
```

To fix it, you need to get the division operator to evaluate prior to casting. You can achieve this by introducing a parenthesis around the division as parenthesis has higher precedence (in fact highest) than casting as per the **precedence** table.

```
1 float myVal = (float)(3.0/2.0);    // dou
```

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 Interview

♦ 11 Spring boot

2
3
4

// doub
// cast

Q7. What is the output of the following code snippet?

```
1 public class PrePostOperators {
2
3     public static void main(String[] args) {
4         int x = 5;
5         int y = ++x;
6         int z = y++;
7
8         System.out.println("x=" + x + ", y=" + y);
9     }
10 }
```

A7. x=6, y=7, z=6

line 1: x=5, y=0, z=0

line 2: x=6, y=6, z=0;

line 3: x=6, y=7, z=6

You need to understand the pre and post increment operators to get this right. ++x is a pre-increment and x++ is a post increment. ++x means x is incremented before being used and x++ means x is incremented after being used. So line2 increments x by one and then assign it to y. Whereas in line 3, z is assigned the old y value (i.e. prior to incrementing) of 6 and then y value is incremented to 7.

As you may rightly ask that as per the precedence table, both pre-increment (i.e. ++expr) and post-increment (i.e. expr++) operators do have precedence over the assignment operator (i.e. "="). The line 3 `int z = y++;` is roughly evaluated as follows:

```
1 int oldY = 6;           // current value
2 y = y+ 1;               // increment y by 1
3 z = oldY;               // z is set to 6
4
```

Q8. Can you list some practical applications where the bitwise operations can be applied?

A8.

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

Example 1: To pack and unpack values.

For example, to represent

- age of a person in the range of 0 to 127. Use 7 bits.
- gender of a person 0 or 1 (0 – female and 1 – male). Use 1 bit.
- height of a person in the range of 0 to 255. Use 8 bits.

To pack this info: $((\text{age} \ll 1) | \text{gender}) \ll 8 | \text{height}$. For example, age = 25, gender = 1, and height = 255cm. Shift the age by 1 bit, and combine it with gender, and then **shift** the age and gender by 8 bits and **combine** it with the height.

Packing

	Age							Gender	Height							
Bits	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
age (25 years) using 7 bits	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1
Age << 1 (Shift age by 1 bit)	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0
Gender (1 – male) using 1 bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Combine age with gender:	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
$(\text{age} \ll 1) \text{gender}$	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0
$((\text{age} \ll 1) \text{gender}) \ll 8$, shift age and gender by 8 bits.	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0
Height (255 cm) using 8 bits.	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
Combine height with age and gender:	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1
$\text{val} = ((\text{age} \ll 1) \text{gender}) \ll 8 \text{height}$																
	Age range							Gender	Height range							
	16 + 8 + 1 = 25							= 1	128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255.							

Bitwise operations

Deleting records

SQL Subquery in

Transaction man

UML (1)

12 UML interview

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 Hado

03: Q16 – Q25 Hado

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 I

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

```

1 public class Binary {
2
3     public static void main(String[] args) {
4
5         //packing
6         int val = (((25 << 1) | 1) << 8) | 255)
7         System.out.println("packed=" + val);
8         System.out.println("packed binary="
9                             + Integer.toBinaryString(val));
10
11        //unpacking
12        System.out.println("height=" + (val & 0x000000ff));
13        System.out.println("gender=" + ((val >>> 8) & 1));
14        System.out.println("age=" + ((val >>> 9) & 0x000000ff));
15
16    }
17 }

```

Output:

```

packed=13311
packed binary=11001111111111
height=255
gender=1
age=25

```

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

Unpacking (or extracting) height: Extract the low order 8 bits.

packed value:	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1
Masking: 0xFF	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
Extract height = value & 0xFF:	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
									2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹ 2 ⁰
height:									128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255						

Unpacking (or extracting) gender: Extract bit 9.

packed value:	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1
8 lower order bits (i.e. shaded area) are shifted out.	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1
value >>> 8															
Masking: 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Extract gender = (value >>> 8) & 1:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
															2 ⁰
gender:															1

Unpacking (or extracting) age: Extract bits 10 – 16 (i.e. higher order bits).

packed value:	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1
Extract age = value >>> 9:	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
												2 ⁶	2 ⁵	2 ⁴	2 ³ 2 ² 2 ¹ 2 ⁰
age:												2 ⁴ + 2 ³ + 2 ⁰ = 16 + 8 + 1 = 25			

Example 2: To compactly represent a number of attributes like being bold, italics, etc of a character in a text editor.

This is a more practical example.

shadow	blink	subscript	superscript	strikethrough	underline	italics	bold
0	1	0	1	0	0	0	1

```

1 import java.util.Arrays;
2
3 public class Binary6 {
4     public static void main(String[] args) {
5         byte[] vals = { 0, 1, 0, 1, 0, 0, 0, 1 };
6
7         byte value = pack(vals);
8         System.out.println("packedValue=" + value);

```

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

```

9      System.out.println("unpackedValues="
10                          + Arrays.toString(unpack(value)))
11  }
12
13  public static byte pack(byte[] vals) {
14      byte result = 0;
15      for (byte bit : vals) {
16          result = (byte) ((result << 1) | (bit & 1));
17      }
18      return result;
19  }
20
21  public static byte[] unpack(byte val) {
22      byte[] result = new byte[8];
23      for (int i = 0; i < 8; i++) {
24          result[i] = (byte) ((val >> (7 - i)) & 1);
25      }
26      return result;
27  }
28  }

```

Example 3: If you can think of anything as slots or switches that need to be flagged on or off,

you can think of bitwise operators. For example, if you want to mark some events on a calendar.

6 Saturday	5 Friday	4 Thursday	3 Wednesday	2 Tuesday	1 Monday	0 Sunday

Example 4: To multiply or divide by 2^n

```

1  public class ShiftOperator {
2
3      //multiply 10 by 2 power n where n = 6
4      private static final int MULTIPLY = 10 << 6;
5
6      //Divide 640 by 2 power n where n = 6.
7      private static final int DIVIDE = 640 >> 6;
8
9      public static void main(String[] args) {
10         System.out.println(MULTIPLY);
11         System.out.println(DIVIDE);
12     }
13 }

```

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 C

Groovy Coding C

Cash balance

Sum grades C

♥ Q1 – Q5 Groov

♦ 20 Groovy clos

♦ 9 Groovy meta

Groovy method c

Popular Posts

♦ 11 Spring boot interview questions & answers

852 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



Q6 – Q10 Groov

JavaScript Interview

JavaScript Top I

♥ Q1 – Q10 J

♦ Q11 – Q20

♦ Q21 – Q30

♦ Q31 – Q37

JavaScript mi

JavaScript Vs J

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

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 identifiers](#)[♥ 14 Unix interview questions & answers for Java developers](#)**Posted in** Data types, member-paid

Leave a Reply

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

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 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.