Register | Login | Logout | Contact Us

Java-Success.com

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

search here … Go

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 › Classes › ♦ Java class loading interview Q&A to ascertain your depth of Java knowledge

♦ Java class loading interview Q&A to ascertain your depth of Java knowledge

Posted on August 12, 2014 by Arulkumaran Kumaraswamipillai — 2 Comments ↓

0 Like Share Tweet 0 G+1 Share

Q1. What do you know about Java class loading? Explain Java class loaders?

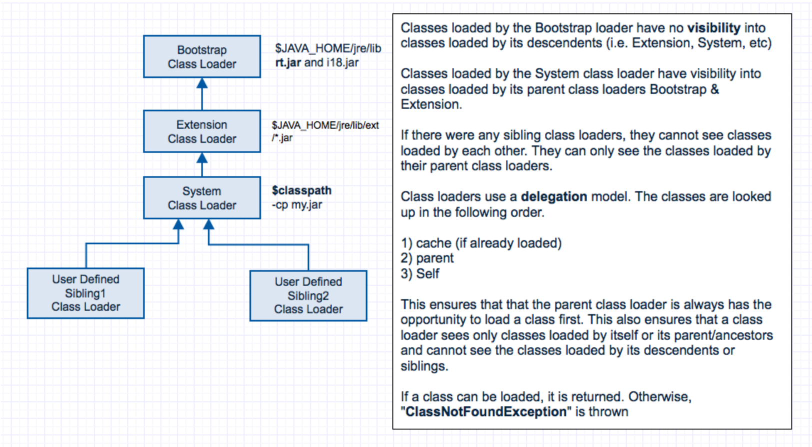
A1. Class loaders are hierarchical. Classes are introduced into the JVM as they are referenced by name in a class that is already running in the JVM. So, how is the very first class loaded? The very first class is specially loaded with the help of static main( ) method declared in your class. All the subsequently loaded classes are loaded by the classes,

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

which are already loaded and running. A class loader creates a namespace. All JVMs include at least one class loader that is embedded within the JVM called the primordial (or bootstrap) class loader. The JVM has hooks in it to allow user defined class loaders to be used in place of primordial class loader. Let us look at the class loaders created by the JVM.



Classes loaded by the Bootstrap loader have no **visibility** into classes loaded by its descendents (i.e. Extension, System, etc)

Classes loaded by the System class loader have visibility into classes loaded by its parent class loaders Bootstrap & Extension.

If there were any sibling class loaders, they cannot see classes loaded by each other. They can only see the classes loaded by their parent class loaders.

Class loaders use a **delegation** model. The classes are looked up in the following order.

1) cache (if already loaded)
2) parent
3) Self

This ensures that that the parent class loader is always has the opportunity to load a class first. This also ensures that a class loader sees only classes loaded by itself or its parent/ancestors and cannot see the classes loaded by its descendents or siblings.

If a class can be loaded, it is returned. Otherwise, "**ClassNotFoundException**" is thrown

Java Class Loader Basics

Class loaders are hierarchical and use a delegation model when loading a class. Class loaders request their parent to load the class first before attempting to load it themselves. When a class loader loads a class, the child class loaders in the hierarchy will never reload the class again. Hence uniqueness is maintained. Classes loaded by a child class loader have visibility into classes loaded by its parents up the hierarchy but the reverse is not true as explained in the above diagram.

Q2. Explain static vs. dynamic class loading?
A2. Classes are statically loaded with Java's "new" operator.

```

class MyClass {
    public static void main(String args[]) {
       Car c = new Car( );
    }
}

```

Dynamic loading is a technique for programmatically invoking the functions of a class loader at run time. Let us look at how to load classes dynamically.

```

//static method which returns a Class
Class.forName (String className);

```

The above static method returns the class object associated with the class name. The string className can be supplied dynamically at run time. Unlike the static loading, the dynamic loading will decide whether to load the class Car or the class Jeep at runtime based on a properties file and/or other runtime conditions. Once the class is dynamically loaded the following method returns an instance of the loaded class. It's just like creating a class object with no arguments.

```

// A non-static method, which creates an instance
// class (i.e. creates an object).
class.newInstance ( );

```

Static class loading throws "**NoClassDefFoundError**" if the class is not found and the dynamic class loading throws "**ClassNotFoundException**" if the class is not found.

Q3. What tips would you give to someone who is experiencing a class loading or "Class Not Found" exception?
A3. "ClassNotFoundException" could be quite tricky to troubleshoot. When you get a ClassNotFoundException, it means the JVM has traversed the entire classpath and not found the class you've attempted to reference.

1) Stand alone Java applications use -cp or -classpath to define all the folders and jar files to look for. In windows separated by ";" and in Unix separated by ":".

```

java -classpath "C:/myproject/classes;C:/myprojec
</code>>/pre>

```

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 numbe
Java 8 String str
Java 8: What is
IO (7)
♥ Reading a text
♦ 15 Java old I/C
06: ♥ Java 8 way
Processing large
Processing large
Read a text file f
Reloading config
Multithreading (12)
01: ♥♦ 15 Beginn
02: ♥♦ 10+ Java
03: ♦ More Java
04: ♦ 6 popular J
05: ♦ How a thre
06: ♦ 10+ Atomic
07: 5 Basic multi
08: ♦ ThreadLoc
09: Java FutureT
10: ♦ ExecutorSe
Java ExecutorSe
Producer and Co
Algorithms (5)
♦ Splitting input t
♦ Tree traversal
♥ ♦ Java coding

```
2) Determine the jar file that should contain the

<pre class="prettyprint"><code class="language-ja
$ find . -name "*.jar" -print -exec jar -tf '{}'

```

You can also search for the class at www.jarfinder.com

3) Check the version of the jar in the manifest file MANIFEST.MF, access rights (e.g. read-only) of the jar file, presence of multiple versions of the same jar file and any jar corruption by trying to unjar it with “jar -xvf …”. If the class is dynamically loaded with Class.forName(“com.myapp.Util”), check if you have spelled the class name correctly.

4) Check if the application is running under the right JDK? Check the JAVA_HOME environment property

```

$  echo $JAVA_HOME

```

5) **-verbose:class** option in your JVM. With the -verbose option all the classes that are loaded are listed, along with the JAR file or directory from which they were loaded. The “class” output shows additional information, such as when superclasses are being loaded, and when static initializers are being run.

6) Creating a **Java dump and analyzing the Java dump** for class loading issues. The Java dumps are created under following circumstances.

— When a fatal native JVM error.
— When the JVM runs out of heaps memory space.
— When a signal is sent to the JVM (e.g. Control-Break is pressed on Windows, Control-\ on Linux, or kill -3 on Unix)

There are tools like **jstack**, **jmap**, **hprof**, and Eclipse Memory Analyzer (MAT) to analyze the Java dumps.

7) Some of the libraries provide **API to list the version number**. For example, The Eclipse link MOXy library

Searching algori
Swapping, partiti
⊟ Annotations (2)
8 Java Annotatic
More Java anno
⊟ Collection and Data
♦ Find the first n
♦ Java Collectior
♥ Java Iterable V
♥♦ HashMap & H
♦ Sorting objects
02: ♦ Java 8 Stre
04: Understandir
4 Java Collectior
If Java did not ha
Java 8: Different
Part-3: Java Tree
Sorting a Map by
When to use whi
⊟ Differences Betwee
♥ Java Iterable V
♦ Multithreading
♦ Why do Proxy,
Core Java Modif
Differences betw
Java Collection i
⊟ Event Driven Progr
Event Driven Pro
Event Driven Pro
⊟ Exceptions (2)
♦ Java exceptior
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

provides a method as shown below.

```

org.eclipse.persistence.Version.getVersion();

```

8) The org.jboss.test.util.Debug class has a method *displayClassInfo(Class clazz, StringBuffer results)* to display the loaded class details. This is done programmatically. What this class essentially does is

```

    URL loc = MyClass.class.getProtectionDomain

```

9) The http://www.findjar.com is an onlime search engine that can list possible jar files in which a particular class file like java.sql.Connection can be found.

Popular Posts

- ♦ 11 Spring boot interview questions & answers

  857 views

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

  825 views

- 18 Java scenarios based interview Questions and Answers

  447 views

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

  400 views

- ♦ 7 Java debugging interview questions & answers

  311 views

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

  302 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

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.

‹ Top 6 tips to go about writing loosely coupled Java applications

♦ Java classes and interfaces are the building blocks ›

04: JSP overviev
05: Web patterns
06: ♦ MVC0, MV
07: When to use
08: Web.xml inte
WebService (11)
01: ♥♦ 40+ Java
02: ♦ 6 Java RES
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 AMQ
Spring JMS with
Spring JMS with
JMX (3)
5 JMX and MBea
Event Driven Pro
Yammer metrics
JNDI and LDAP (1)
JNDI and LDAP
Pressed for time? Jav
Job Interview Ice B

Posted in Classes, member-paid

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

2 comments on "♦ Java class loading interview Q&A to ascertain your depth of Java knowledge"

richdyang says:

May 5, 2015 at 4:09 pm

Classes loaded by a child class loader have visibility into classes loaded by its parents up the hierarchy.

This statement is obviously wrong! The fact is the reverse.

Reply

Arulkumaran says:

May 6, 2015 at 11:10 pm

Why do you think it is the revers? For example, the java.lang.Object class loaded by the bootsrtap class loader does not want know about your own class say MyTest.java loaded by the System class loader. But you your class MyTest.java must know the java.lang.Object as every Java object is inherited from java.lang.Object.

Reply

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 Be
- Java Key Area Ess
  - ♦ Design pattern
  - ♥ Top 10 causes
  - ♥♦ 01: 30+ Writin
  - ♦ 12 Java design
  - ♦ 18 Agile Develo
  - ♦ 5 Ways to debu
  - ♦ 9 Java Transac
  - ♦ Monitoring/Pro
  - 02: ♥♦ 13 Tips to
  - 15 Security key a
  - 4 FAQ Performa
  - 4 JEE Design Pa
  - 5 Java Concurre
  - 6 Scaling your Ja
  - 8 Java memory i
- OOP & FP Essentia
  - ♦ 30+ FAQ Java
  - 01: ♦ 19 Java 8 I
  - 04: ♥♦ Top 6 tips
- Code Quality Job In
  - ♦ 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 scenario
    - Auditing databas

- Deleting records
- SQL Subquery ir
- Transaction mar

- 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 interv
- 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 archit
  - 01: ♥♦ 40+ Java W
  - 02: ♥♦ 13 Tips to w
  - 03: ♦ What should I
  - 04: ♦ How to go ab
  - 05: ETL architectur
  - 1. Asynchronous pı
  - 2. Asynchronous pı
- Scala Interview Q&As
  - 01: ♥ Q1 – Q6 Sca
  - 02: Q6 – Q12 Scala
  - 03: Q13 – Q18 Sca

    - 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 perforn
  - 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 Cobe
  - 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 perforn

- 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 f
  - Finance Domain In
    - 12+ FX or Forex
    - 15 Banking & fin
  - FIX Interview Q&A
    - 20+ FIX basics ii
    - Finding your way
  - Groovy Interview Q
    - Groovy Coding (
      - Cash balance
      - Sum grades G
    - ♥ Q1 – Q5 Groov
    - ♦ 20 Groovy clos
    - ♦ 9 Groovy meta
    - Groovy method (

Q6 – Q10 Groov
JavaScript Intervie
JavaScript Top I
♥ Q1 – Q10 J
♦ Q11 – Q20
♦ Q21 – Q30
♦ Q31 – Q37
JavaScript mi
JavaScript Vs J
JavaScript Vs
Unix Interview Q&
♥ 14 Unix intervi
♥ Hidden Unix,
sed and awk to
Shell script inter
Unix history com
Unix remoting in
Unix Sed comm
▶ Free Java Intervie
▶ 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 I

⊞ Low Latency (7)

⊞ Memory Managemen

⊞ 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 Tu

⊞ 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.

↑ Responsive Theme powered by WordPress