Java-Success.com

Industrial strength Java/JEE Career Companion to open more doors

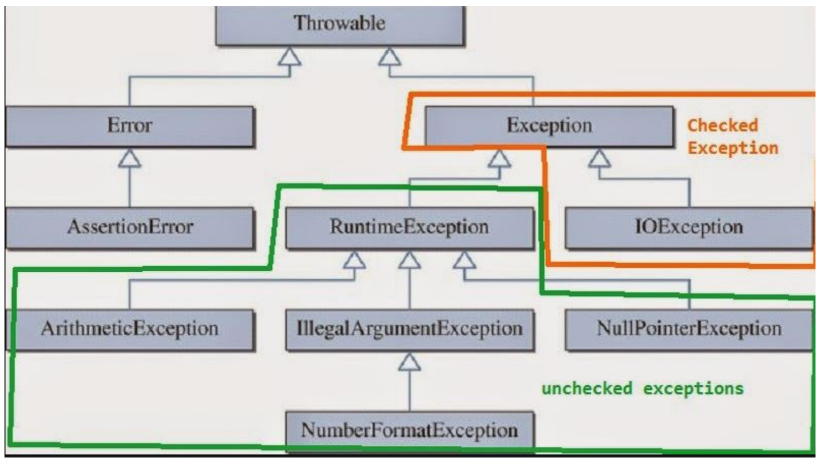
♦ Java exception handling interview questions and answers

Posted on September 3, 2014 by Arulkumaran Kumaraswamipillai — No Comments ↓

Q1. What is the difference between a runtime exception and a checked exception?

A1. You must either catch or throw a checked exception. The unchecked exception (aka Runtime exception) does not have to be explicitly handled.

600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs



Checked Vs Unchecked

Q. So, when to use **checked exception**, and when to use **unchecked exception**?
A. There is no clear cut answer. Document a consistent exception handling strategy. In general favor unchecked (i.e. Runtime) exceptions, which you don't have to handle with catch and throw clauses. Use checked exceptions in a rare scenarios where you can recover from the exception like deadlock or service retries. In this scenario, you will catch a checked exception like *java.io.IOException* and wait a few seconds as configured with the retry interval, and retry the service configured by the retry counts.

Q2. What is the problem or benefit of catching or throwing type "java.lang.Exception"?
A2. Exceptions are **polymorphic** in nature., which means you need to catch a more specific ones before the generic ones. For example, ***IOException*** must be caught before ***Exception*** as the *IOException* extends *Exception*. If you catch the *Exception* before *IOException*, then *IOException* catch block will never be reached as *Exception* catches everything.

So, it is **wrong** to catch Exception before IOException.

```
try{
   //....
} catch(Exception ex){
   log.error("Error:" + ex)
} catch(IOException ex){
```

16 Technical Key Areas

```
    log.error("Connectivity issue:" + ex); //neve
}
```

Fix this by catching the more specific IOException first

```
try{
   //....
} catch(IOException ex){
    log.error("Connectivity issue:" + ex);
} catch(Exception ex){
    log.error("Error:" + ex)
}
```

In **Java 7 on wards**, you can catch multiple exceptions like

```
try{
   //....
}
catch (ParseException | IOException exception) {
    // handle I/O problems.
} catch (Exception ex) {
    //handle all other exceptions
}
```

Q3. Does Java's exception handling use any design pattern?
A3. Yes. Java's exception handling mechanism uses the "Chain of Responsibility" design pattern.

1. Sender of an exception will not know which object in the chain will serve its request.
2. Each processor in the chain may decide to serve the exception by catching and logging or
3. wrapping it with an application specific exception and then rethrowing it to the caller or
4. don't handle it and leave it to the caller

Q4. What are the different ways to look at the trace of your program execution?
A4. Java is a stack based language, and the program execution is pushed and popped out of a stack. When a method is entered into, it is pushed into a stack, and when that method invokes many other methods, they are pushed into the stack in the order in which they are executed. As each method completes its execution, they are popped out of the stack in the LIFO order. Say methodA( ) invoked

methodB( ), and methodB( ) invoked methodC ( ), when execution of methodC( ) is completed, it is popped out first, and then followed by methodB( ) and then methodA( ). When an exception is thrown at any point, a stack trace is printed for you to be able to find where the issue is.

A Java developer can access a stack trace at any time. One way to do this is to call

```
Thread.currentThread().getStackTrace() ; //handy
```

You could get a stack trace of all the threads using the Java utilities such as jstack, JConsole or by sending a kill -quit signal (on a Posix operating system) or on WIN32 platform to get a thread dump. The thread dumps are very useful in identifying concurrency issues like dead locks, contention issues, thread starvation, etc.

Q5. How would you go about analyzing stack traces correctly?
A5.

1. One of the most important concepts of correctly understanding a stack trace is to recognize that it lists the execution path in reverse chronological order from most recent operation to earliest operation. That is, it is LIFO.

2. The stack trace below is simple and it tells you that the root cause is a NullPointerException on ClassC line 16. So you look at the top most class.

```
Exception in thread "main" java.lang.NullPointerE
        at com.myapp.ClassC.methodC(ClassC.java:1
        at com.myapp.ClassB.methodB(ClassB.java:2
        at com.myapp.ClassA.main(ClassA.java:14)

```

3. The stack trace can get more complex with multiple “caused by” clauses, and in this case you usually look at the bottom most “caused by”. For example,

```
Exception in thread "main" java.lang.IllegalStat
        at com.myapp.ClassC.methodC(ClassC.java:
        at com.myapp.ClassB.methodB(ClassB.java:
        at com.myapp.ClassA.main(ClassA.java:14)
Caused by: com.myapp.MyAppValidationException
        at com.myapp.ClassB.methodB(ClassB.java:
        at com.myapp.ClassC.methodC(ClassC.java:
        ... 1 more
Caused by: java.lang.NullPointerException
        at com.myapp.ClassC.methodC(ClassC.java:
        ... 1 more

```

The root cause is the last “caused by”, which is a NullPointerException on ClassC line 16.

4. When you use plethora of third-party libraries like Spring, Hibernate, etc, your stack trace’s “caused by” can really grow and you need to look at the bottom most “caused by” that has the package relevant to you application like com.myapp.ClassC and skip library specific ones like org.hibernate.exception.*.

Popular Posts

♦ 11 Spring boot interview questions & answers

825 views

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

765 views

18 Java scenarios based interview Questions and Answers

399 views

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

388 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

295 views

♦ 7 Java debugging interview questions & answers

293 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

285 views

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

279 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

239 views

001B: ♦ Java architecture & design concepts interview questions & answers

201 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.**945+** paid members. join my LinkedIn Group. **Reviews**

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.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ “In your Java experience” interview questions & answers

♦ 12 FAQ JDBC interview questions and answers ›

Posted in Exception Handling, Exceptions, member-paid

Tags: Core Java FAQs, Java/JEE FAQs

Leave a Reply

Logged in as geethika. Log out?

Comment

Post Comment

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