8 Java memory management interview Q&A

Posted on September 12, 2014 by Arulkumaran Kumaraswamipillai — No Comments ↓

Q1. Are memory leaks possible in Java, which has memory management via automatic Garbage Collection?

A1. Memory and resource leaks are possible in any robust application. In managed languages such as Java and C#, the developers do not have to worry too much about memory management as the garbage collector (GC) will do this job for you. The garbage collectors can be tuned with different algorithms and hints depending on the behavior of the application. This does not mean that the garbage collected languages are immuned from memory leaks. Memory leaks can occur in managed languages when objects of longer life cycle (e.g. static or global variables, singleton classes, etc) hold on to objects of a shorter life cycle. This prevents the objects with short life cycle being garbage collected. The

developers must remember to remove the references to the short-lived objects from the long-lived objects.

Q2. What causes memory leaks in your Java applications?
A2.

— Long living objects having reference to short living objects, causing the memory to slowly grow. For example, singleton classes referring to short lived objects. This prevents short-lived objects being garbage collected.

— Improper use of thread-local variables. The thread-local variables will not be removed by the garbage collector as long as the thread itself is alive. So, when threads are pooled and kept alive forever, the object might never be removed by the garbage collector.

— Using mutable static fields to hold data caches, and not explicitly clearing them. The mutable static fields and collections need to be explicitly cleared.

— Objects with circular or bidirectional references that are accessible from a thread.

— JNI memory leaks.

Q3. How will you go about creating a memory leak in Java?
A3. In Java, memory leaks are possible under a number of scenarios. Here is a typical example where hashCode( ) and equals( ) methods are not implemented for the Key class that is used to store key/value pairs in a HashMap. This will end up creating a large number of duplicate objects. All memory leaks in Java end up with java.lang.OutOfMemoryError, and it is a matter of time. The following code agressively creates the OutOfMemoryError via an endless loop for demonstration purpose.

If you are not familiar with the significance of equals( ) and hashCode ( ) methods in Java learn how to define proper key class in Java.

```
1  import java.util.HashMap;
2  import java.util.Map;
3
4  public class MemoryLeak {
5
6   public static void main(String[] args) {
7      Map<Key, String> map = new HashMap<Key, Stri
8
9      int counter = 0;
10     while (true) {
11     // creates duplicate objects due to bad Key
12       map.put(new Key("dummyKey"), "value");
13       counter++;
14       if (counter % 1000 == 0) {
15         System.out.println("map size: " + map.si
16         System.out.println("Free memory after co
17            + " is " + getFreeMemory() + "MB");
18
19        sleep(1000);
20       }
21     }
22  }
23
24  // inner class key without hashcode() or equals
25  static class Key {
26    private String key;
27
28    public Key(String key) {
29      this.key = key;
30    }
31
32  }
33
34  //delay for a given period in milli seconds
35  public static void sleep(long sleepFor) {
36     try {
37       Thread.sleep(sleepFor);
38     } catch (InterruptedException e) {
39       e.printStackTrace();
40     }
41  }
42
43  //get available memory in MB
44  public static long getFreeMemory() {
45     return Runtime.getRuntime().freeMemory() / (
46  }
47 }
48
```

Output:

```
1  map size: 1000
2  Free memory after count 1000 is 4MB
3  map size: 2000
4  Free memory after count 2000 is 4MB
5  map size: 1396000
6  Free memory after count 1396000 is 2MB
7  map size: 1397000
8  Free memory after count 1397000 is 2MB
9  map size: 1398000
10 Free memory after count 1398000 is 2MB
```


- Git & SVN (6)
- JMeter (2)
- JSF (2)
- Maven (3)
- Testing & Profiling/Sa
- Other Interview Q&A f
- Free Java Interviev

16 Technical Key Areas

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

```
map size: 1399000
Free memory after count 1399000 is 1MB
map size: 1400000
Free memory after count 1400000 is 1MB
map size: 1401000
Free memory after count 1401000 is 1MB
.....
.....
map size: 1452000
Free memory after count 1452000 is 0MB
map size: 1453000
Free memory after count 1453000 is 0MB
Exception in thread "main" java.lang.OutOfMemory
 at java.util.HashMap.addEntry(HashMap.java:753)
 at java.util.HashMap.put(HashMap.java:385)
 at MemoryLeak.main(MemoryLeak.java:10)

```

As you could see, the size of the map keeps growing with the same objects and the available memory keeps coming down from 4MB to 0MB. At the end, the program dies with an **OutOfMemoryError**.

Q4. How will you fix the above memory leak?

A4. By providing proper implentation for the key class as shown below with the equals() and hashCode() methods.

```
static class Key {
   private String key;

   public Key(String key) {
     this.key = key;
   }

   @Override
   public boolean equals(Object obj) {
      if (obj instanceof Key)
        return key.equals(((Key) obj).key);
      else
         return false;
   }

   @Override
   public int hashCode() {
     return key.hashCode();
   }
}

```

Q6. In real applications, how do you know that you have a memory leak?

A6. If you profile your application, you can notice a graph like a saw tooth. Here is how you can determine this with the help of jconsole for the above bad key class example. All you have

JEE Tutorials (19)
Scala Tutorials (1)
Spring & HIbernate Tu
Tools Tutorials (19)
Other Tutorials (45)

100+ Java pre-interview coding tests

open all | close all
Can you write code? (
♦ Complete the given
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 .....?

open all | close all
Career Making Know-
Job Hunting & Resum

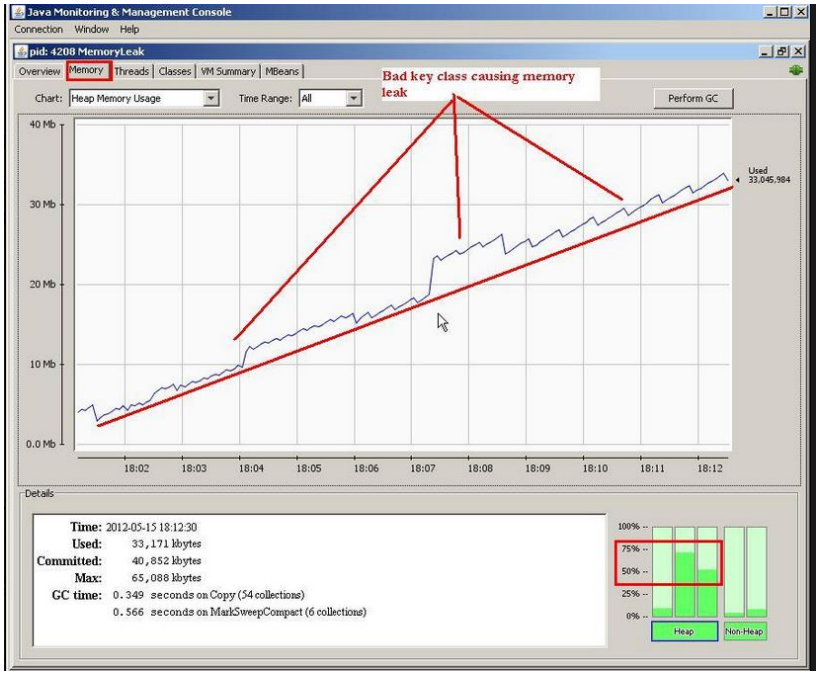
to to do is while your MemoryLeak class is running, get the Java process id by typing

```
C:\>jps
5808 Jps
4568 MemoryLeak
3860 Main

```

Now, open up the jconsole as shown below on a command line

```
C:\>jconsole 4568
```



Memory Leak

Q7. How will you go about profiling your Java application for memory usage?

A7. There are number of tools both commercial and open-source. There are command line tools that get shipped with Java like **hprof**, **jconsole**, **jhat**, and **jmap**. **Visual VM** is a graphical tool shipped with Java 6 on wards.

There are commercial tools that can be used in production environment like **YourKit** for Java, **JProfiler** for Java, etc and

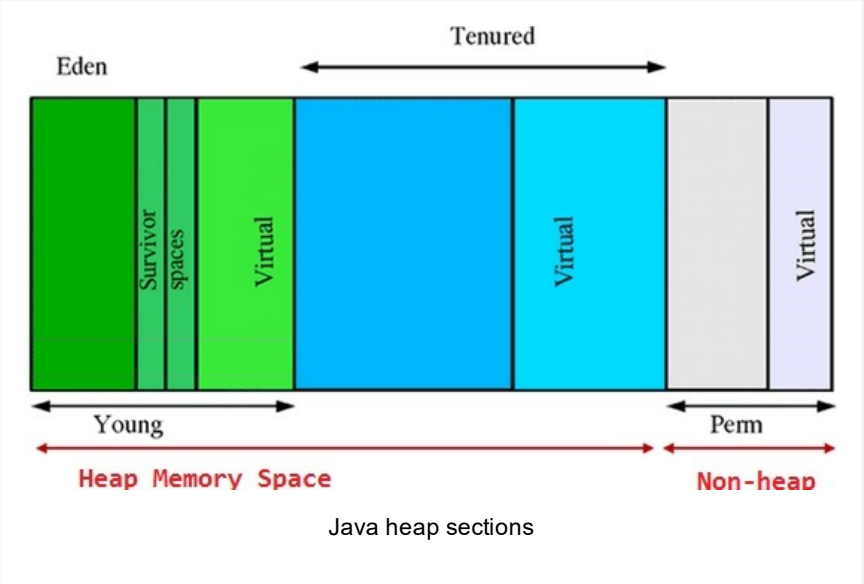
for larger distributed and clustered systems with large number of nodes there are profilers like CA Wiley **Introscope** for Java, **ClearStone** for Java, and **HP Performance managers**.

Q8. What causes the java.lang.OutOfMemoryError: PermGen space memory leak?

A8. Every JVM nowadays uses a separate region of memory, called the **Permanent Generation** (or PermGen for short), to hold internal representations of java classes entailing the fields and more.

— Methods of a class (including the bytecodes)
— Names of the classes (in the form of an object that points to a string also in the permanent generation)
— Constant pool information (data read from the class file, see chapter 4 of the JVM specification for all the details).
— Object arrays and type arrays associated with a class (e.g., an object array containing references to methods).
— Internal objects created by the JVM (java/lang/Object or java/lang/exception for instance)
— Information used for optimization by the compilers (JITs)

The diagram below shows the PermGen space with "**Perm**"



Java heap sections

The PermGen space memory leak can be caused by Leaking Threads, Leaking Drivers, and not allocating enough PermGen Space via JVM config.

One possible scenario for a classloader leak is through **long running threads**. This happens when your application or or a 3rd party library used by your application starts some long running thread. An example of this could be a timer thread whose job is to execute some code periodically.

Another typical case of a leak can be caused by database drivers.

Refer to "Java Garbage Collection " for GC related interview Q&A.

Popular Posts

♦ 11 Spring boot interview questions & answers

828 views

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

768 views

18 Java scenarios based interview Questions and Answers

400 views

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

389 views

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

296 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

286 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

240 views

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

202 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**

‹ ♦ 9 Java Transaction Management Interview Q&A

Security holes & how to fix them interview Q&A ›

Posted in Java Key Area Essentials, member-paid, Memory Management

Tags: Architect 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

 ↑