JavaMadeSoEasy.com (JMSE)

Search This Blog

Search

OutOfMemoryError in java in lots of detail

Contents of page >

1) Different types of java.lang.OutOfMemoryError >

Exception in thread: java.lang.OutOfMemoryError: **Java heap space**
Exception in thread: java.lang.OutOfMemoryError: **GC Overhead limit exceeded**
Exception in thread: java.lang.OutOfMemoryError: **Requested array size exceeds VM limit**
Exception in thread: java.lang.OutOfMemoryError: **Metaspace**
Exception in thread: java.lang.OutOfMemoryError: **request size bytes for reason. Out of swap space?**
Exception in thread: java.lang.OutOfMemoryError: **Compressed class space**
Exception in thread: java.lang.OutOfMemoryError: **reason stack_trace_with_native_method**
Out of memory:**Kill process or sacrifice child**
Exception in thread : java.lang.OutOfMemoryError: **permgen**

java.lang.OutOfMemoryError is the commonly a **indication** of a **memory leak** in java.
Now, let's discuss all of them in detail.

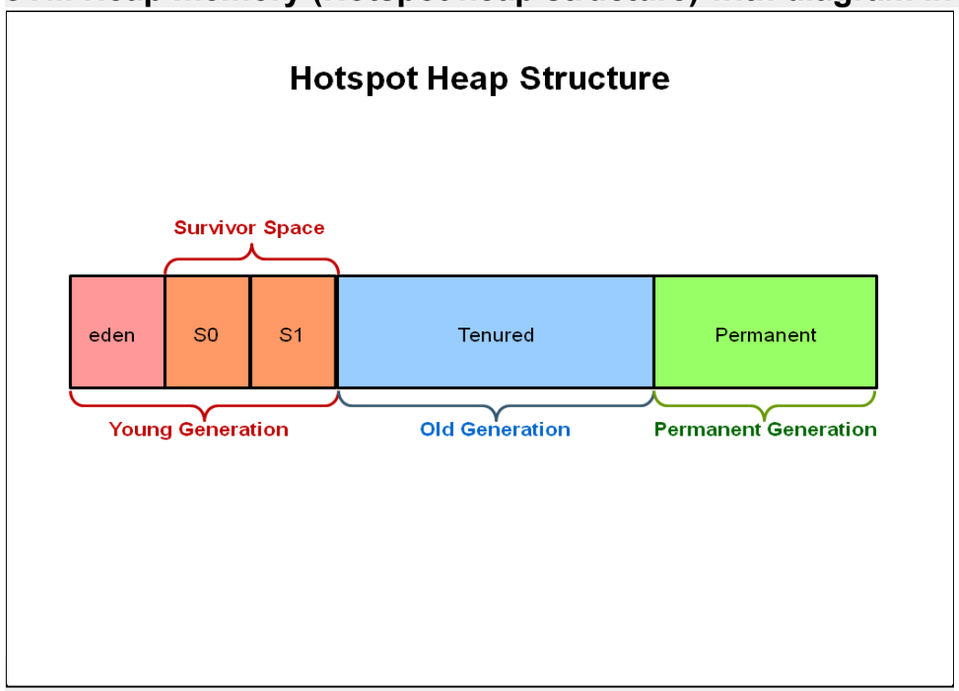
- OutOfMemoryError
- Overriding equals and hashCode
- PostgreSQL
- Producer Consumer problem/pattern
- Pyramid generation
- RegEx
- Serialization
- Thread Concurrency
- Threads
- TUTORIALS
- Windows

Join our Groups>

- FACEBOOK
- LINKED IN

Before learning in detail about OutOfMemoryError in java, we must know about JVM Heap memory (Hotspot heap structure). Let's learn about heap in just 1 minute.

JVM Heap memory (Hotspot heap structure) with diagram in java >



JVM Heap memory (Hotspot heap structure) in java consists of following elements>

1. **Young** Generation
    - 1a) **Eden**,

- 1b) **S0 (Survivor** space 0**)**
- 1c) **S1 (Survivor** space 1**)**

2. **Old** Generation (Tenured)
3. **Permanent** Generation.

Learn this in more detail here : JVM Heap memory (**Hotspot heap structure**), What are Young, Old (tenured) and Permanent Generation, Minor, Major and Full garbage collection in JVM

2) Scenarios where OutOfMemoryError may be thrown >

- Usually whenever there is **insufficient** space to **allocate an object** in the Java **heap** OutOfMemoryError is thrown.

- What happens when there is insufficient space to **allocate an object** in the Java **heap >** The garbage collector is **unable** make some space available to **accommodate a new object**, and even the java **heap cannot be expanded** to create new object. (See how to adjust heap size using -Xms and -Xmx jvm parameters)

- OutOfMemoryError may also be thrown when there is **insufficient native memory** to **support** the **loading of** a **Java class**.

- OutOfMemoryError may also be thrown when an **excessive amount of time** is being by jvm in performing **garbage collection** and very little memory is being freed.

All Labels

Algorithm(34) Apache Tomcat Server(4) Arrays Java(30) Autoboxing(5) Basic java programs for beginners(15) Binary Trees(6) Collection Framework(68) Collection programs(105) Collections implementation(16) Comparable and Comparator program(22) Core Java(1032) core java Basics(38) Core java Conversions(21) Core Java Differences(11) Core Java Tutorials(12) CRUD MongoDB java(5) CRUD operations MongoDB(3) Custom implementation of Data Structures(11) Data Structure(26) Database(1) Database Tutorials(3) Date tutorials(11) DeSerialization(19) eclipse(27) Exceptions(71) File IO/File handling(71) Garbage collection in java(43) Generics(5) Hashing(8) Hibernate(1) Interview questions(14) iText Pdf tutorial(45) Java 4(1) java 5(3) Java 7(32) Java 8(80) Java 9(12)

- OutOfMemoryError exception can also be thrown by **native library code** when a **native allocation cannot be satisfied** (Example - if swap space is low).

3) OutOfMemoryError: Java heap space

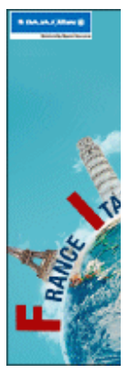
3.1) Exception in thread thread_name java.lang.**OutOfMemoryError: Java heap space** in java

OutOfMemoryError : Java heap space - is thrown whenever there is **insufficient** space to **allocate an object** in the **Java heap.**

Is **Exception in thread threadName -** java.lang.**OutOfMemoryError - Java heap space**
Indicates memory leak?
No, this **OutOfMemoryError** does not necessarily means that it is memory leak.

3.2) How to **solve** Exception in thread thread_name java.lang.OutOfMemoryError - Java heap space in java?

You may need to increase the heap size using -Xms and -Xmx jvm parameters as a solution to this issue.



2.1) What is -Xms JVM parameter in java?

-Xms : Xms is **minimum heap size** which is allocated at initialization of JVM in java.

Examples of using **-Xms** VM (JVM) option in java >

Example1 of using **-Xms** VM (JVM) option in java >
java -Xms512m MyJavaProgram
It will set the minimum heap size of JVM to 512 megabytes.

2.2) What is -Xmx JVM parameter in java?

-Xmx : Xmx is the **maximum heap size** that JVM can use.

Examples of using **-Xmx** VM option in java >

Example1 of using **-Xmx** VM (JVM) option in java >
java -Xmx512m MyJavaProgram
It will set the maximum heap size of JVM to 512 megabytes.

Read more in detail - What are **-Xms** and **-Xmx** JVM parameters in java, and differences between them with examples

3.3) **OutOfMemoryError** may also be thrown in java when >

Popular Posts of JavaMadeSoEasy

- HashMap Custom implementation in java - How HashMap works internally with diagrams and full program
- Collection Quiz in Java - MCQ - Multiple choice questions
- CORE JAVA - Top 120 most interesting and important interview questions and answers in core java
- THREADS - Top 80 interview questions and answers in java for freshers and experienced(detailed explanation with programs) Set-1 > Q1- Q60
- Core Java Tutorial in detail with diagram and programs - BEST EXPLANATION EVER
- COLLECTION - Top 100 interview questions and answers in java for fresher and experienced in detail - Set-1 > Q1- Q50

OutOfMemoryError may also be thrown when an **excessive amount of time** is being by jvm in performing **garbage collection** and very little memory is being freed.

A long lived application might be unintentionally **holding references to objects** and this **prevents the objects from being garbage collected**. Holding of objects for a long time is also a **kind of memory leak** in java.

Also one of the most important source of **OutOfMemoryError (Java heap space)** could be the **excessive use** of **finalizers in** the **application**.

What happens with the **excessive use** of **finalizers in** the **application**?
If a **class has** a **finalize method**, then **space for object** is **not reclaimed at garbage collection** time. Instead the **objects** are **queued** for **finalization after garbage collection**, finalization occurs at some **later time**.

3.4) How excessive use of finalizers could **cause** OutOfMemoryError in java?

finalizers are **executed by** a **daemon threads**. As we discussed above that finalization occurs at some later time. **Holding** finalizer daemon **threads for long time** could **fill the Java heap** and **cause** OutOfMemoryError.

3.5) Example/Program in java to generate OutOfMemoryError: Java heap space >

Before executing program pass this vm parameter in eclipse.

- Find sum of both diagonals in matrix - program in java
- LinkedHashMap Custom implementation in java - How LinkedHashMap works internally with diagrams and full program
- Thread/multi threading Quiz in Java - MCQ - Multiple choice questions
- EXCEPTIONS - Top 60 interview questions and answers in java for fresher and experienced - detailed explanation with diagrams Set-1 > Q1- Q25

-Xmx5m *(Here maximum heap memory set to just* 5 megabytes, so that we could easily produce OutOfMemoryError: Java heap space*)*

```
package outofmemory;

import java.util.ArrayList;
import java.util.List;

/**
 *
 * Write a program which could throw
 * java.lang.OutOfMemoryError : Java Heap Space
 *
 */
public class OutOfMemoryErrorJavaHeapSpace {
   static List<String> list=new ArrayList();
   public static void main(String[] args) {
        while(true){
            list.add(new String("a"));   //Line 16
        }
   }
}
```

OUTPUT of program >

```
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
   at java.util.Arrays.copyOf(Unknown Source)
   at java.util.Arrays.copyOf(Unknown Source)
   at java.util.ArrayList.grow(Unknown Source)
   at java.util.ArrayList.ensureExplicitCapacity(Unknown Source)
   at java.util.ArrayList.ensureCapacityInternal(Unknown Source)
```

at java.util.ArrayList.add(Unknown Source)
at outofmemory.OutOfMemoryErrorJavaHeapSpace.main(**OutOfMemoryErrorJavaHeapSpace.java:16**)

The solution is simply to increase the Xmx parameter to ***-Xmx512m*** *as we discussed above.*

For more details please read : **Exception in thread java.lang.OutOfMemoryError: Java heap space**

Must read: How to set, change, increase or decrease heap size in tomcat server and eclipse to avoid OutOfMemoryError ?

>How to set or change permgen size in tomcat server, eclipse?

4) OutOfMemoryError: GC Overhead limit exceeded

4.1) Exception in thread threadName - java.lang.**OutOfMemoryError: GC Overhead limit exceeded** in java

OutOfMemoryError: GC Overhead limit exceeded - **indicates that** the **garbage collector is running all the time** and **Java program is making** very **slow progress**.

After a GC (**garbage collection)**, if the **garbage collector** is spending more than **98% of its time in doing garbage collection** and if **less than 2% of the java heap**

memory space is reclaimed, then **OutOfMemoryError** - GC Overhead limit exceeded - is thrown in java.

This OutOfMemoryError is generally thrown because all the **live objects are not getting garbage collected properly and java heap space is not available for new objects.**

4.2) How to **avoid** OutOfMemoryError - GC Overhead limit exceeded in java?

You must **increase the heap size** to avoid OutOfMemoryError - GC Overhead limit exceeded in java as a solution to this issue..

4.3) How to **turn off** OutOfMemoryError - GC Overhead limit exceeded in java?

You can **turn it off** by using **VM** (JVM) argument **-XX:-UseGCOverheadLimit**

4.4) Example/Program in java to **generate** OutOfMemoryError: GC Overhead limit exceeded >

Before executing program pass this vm parameter in eclipse.
-Xmx4m -XX:+UseParallelGC *(*so that we could easily produce OutOfMemoryError: GC Overhead limit exceeded*)*

```
package outofmemory;

import java.util.ArrayList;
import java.util.List;

/**
 *
 * Write a program which could throw
 * java.lang.OutOfMemoryError : GC Overhead limit exceeded
 *
 */
public class OutOfMemoryErrorGCoverheadLimitExceeded {
   static List<String> list=new ArrayList();
   public static void main(String[] args) {
        while(true){
            list.add(new String("a"));  //Line 16
        }
   }
}
```

OUTPUT of program >

Exception in thread "main" java.lang.**OutOfMemoryError: GC overhead limit exceeded**
at
outofmemory.OutOfMemoryErrorGCoverheadLimitExceeded.main(OutOfMemoryErrorGCoverheadLimitExceeded.java:**16**)

The solution is simply to increase the Xmx parameter to ***-Xmx512m*** *as we discussed above.*

For more details please read : java.lang.OutOfMemoryError: GC Overhead limit exceeded - solved

5) OutOfMemoryError : unable to create new native Thread - Xss JVM option

5.1) What is java.lang.OutOfMemoryError : unable to create new native Thread ?

When JVM don't have enough memory/space to create new thread it throws OutOfMemoryError : unable to create new native Thread.

Also read : Read in detail about : OutOfMemoryError in java , **JVM (java virtual machine)** in detail in java and How **Garbage Collection (GC) works internally** in detail in java - BEST EXPLANATION EVER

5.2) Solving OutOfMemoryError : unable to create new native Thread ?

You can **resolve** "java.lang.OutOfMemoryError : unable to create new native Thread" by setting the **appropriate size using -Xss** vm option.

Solution 1 to "java.lang.OutOfMemoryError : unable to create new native Thread" >

Try to increase the the -Xss value so that new threads gets enough stack space.

Solution 2 to "java.lang.OutOfMemoryError : unable to create new native Thread" >
Alternatively you could also increase the heap size available using -Xms and -Xmx options
and then try to **increase and set appropriate -Xss value**.

5.3) How to use Using the **VM** (virtual machine) option **ss**?

We can use the VM option **ss** to adjust the maximum stack size.
VM option is passed using -X followed by your VM option. So, passing ss with -X forms **-Xss**.

Examples of using -Xss

Pass memory value you want to allocate to thread stack with -Xss.

Example1 of using -Xss >
java -Xss512m MyJavaProgram
It will set the default stack size of JVM to 512 megabytes.

Example2 of using -Xss >
java -Xss1g MyJavaProgram
It will set the default stack size of JVM to 1 gigabyte.

5.4) What happens if value of -Xss set is **too high**?

Setting excessive value of -Xss parameter could cause StackOverFlowError in java.

5.5) Program which throws java.lang.OutOfMemoryError : unable to create new native Thread

```
/**
 *
 * Write a program which could throw
 * java.lang.OutOfMemoryError : unable to create new native Thread
 *
 */
public class OutOfMemoryErrorUnableToCreateNewNativeThreadExample {
   public static void main(String[] args) {
        //start/spawn infinite Threads
        while(true){
             final int i=0;
         new Thread(new Runnable(){
            public void run() {
              System.out.println(Math.random() +" ");
              try {
                  Thread.sleep(10000000);
              } catch(InterruptedException e) { }
            }
         }).start(); //When JVM don't have enough space to create new thread it
                        throws OutOfMemoryError : unable to create new native Thread

        }
   }
}
```

In the above program we are **spawning infinite Threads.**
In **while loop** we are **starting threads** and putting them to **sleep for a long time**, **so that** the **threads don't die. (**Learn thread states in java**)**

After certain time JVM won't have enough space to create new thread and throw OutOfMemoryError : unable to create new native Thread.

Note : As above program will throw **OutOfMemoryError : unable to create new native Thread.** **So, executing above program might make your system to hang as it will run out of memory.**

Output of above >

```
Exception in thread "main" java.lang.OutOfMemoryError: unable to create new native thread
at java.lang.Thread.start0(Native Method)
at java.lang.Thread.start(Unknown Source)
at infiniteThreads.main(infiniteThreads.java:19)
```

5.6) -Xss option is also known as >

Also you must know that -Xss option is same as **-XX:ThreadStackSize**

5.7) Default Values of -Xss for different platforms >

For windows 32 bit its 64 KB.
For linux 32 bit its 128 KB.
For windows 64 bit its 128 KB.
For linux 64 bit its 256 KB.
For Solaris Sparc it's 512KB.

5.8) Every thread has its own stack >

You must know that each and every thread has its own stack, which makes the methods thread-safe as well.

For more details please read : Solve java.lang.**OutOfMemoryError : unable to create new native Thread - Xss** JVM option

6) OutOfMemoryError: Requested array size exceeds VM limit

6.1) Exception in thread threadName - java.lang.**OutOfMemoryError: Requested array size exceeds VM limit** in java

OutOfMemoryError: Requested array size exceeds VM limit - **indicates** that the java **application tried to allocate an array larger than the heap size**.

6.2) Example of OutOfMemoryError - Requested array size exceeds VM limit in java >

If **heap size is 512 MB,** and
java **application tries to allocate an array of size 1024 MB.**

In this case, OutOfMemoryError - Requested array size exceeds VM - will be thrown **because** java application tried to allocate an array larger than the heap size.

6.3) How to **avoid/solve** OutOfMemoryError - Requested array size exceeds VM limit in java?

You must **increase the heap size** to avoid OutOfMemoryError - Requested array size exceeds VM limit.

Also, you may check the size of array which you are creating, because generally size of array shouldn't be not that large, if size of array is more than size of java heap it may be a faulty array.

The solution is simply to increase the Xmx parameter to ***-Xmx512m***

6.4) Example/program in java to generate OutOfMemoryError: **Requested array size exceeds VM limit >**

Before executing program pass this vm parameter in eclipse.
-Xmx5m *(Here maximum heap memory set to just* 5 megabytes, so that we could easily produce OutOfMemoryError: Requested array size exceeds VM limit*)*

```
package outofmemory;

/**
 *
 * Write a program which could throw
 * java.lang.OutOfMemoryError : Requested array size exceeds VM limit
 *
 */
public class OutOfMemoryErrorRequestedArraySizeExceedsVMlimit {
   public static void main(String[] args) {
        Integer[] array = new Integer[10000 * 10000];  //Line 11
   }
}
```

OUTPUT of program >

```
Exception in thread "main" java.lang.OutOfMemoryError: Requested array size exceeds VM limit
    at
outofmemory.OutOfMemoryErrorRequestedArraySizeExceedsVMlimit.main(OutOfMemoryErrorRequestedArraySizeExceedsVMlimit.java:11)
```

The solution is simply to increase the Xmx parameter to ***-Xmx512m*** *as we discussed above.*

For more details please read : **How to solve OutOfMemoryError: Requested array size exceeds VM limit**

7) OutOfMemoryError: Metaspace

7.1) Exception in thread threadName - java.lang.**OutOfMemoryError: Metaspace** in java

java.lang.OutOfMemoryError:Metaspace is thrown when **there is no space to allocate metaspace** for **java class metadata.**

7.2) What is Java class **metadata**?

Java class metadata is the **JVM's** (Java virtual machine) **internal presentation** of **Java class**.

7.3) What is **metaspace**?

Java class metadata is **allocated in native memory** called **metaspace**.

7.4) When OutOfMemoryError: Metaspace is thrown in java?

OutOfMemoryError: Metaspace is thrown **when -**

- **Too many class are loaded or**
- **Classes loaded very huge in size.**

In this case there is no space to allocate metaspace for java class metadata.

7.5) How Metaspace to be allocated is decided in java?

Metaspace **available for java class metadata** is limited by **VM** (JVM) argument **-XX:MaxMetaspaceSize**,

7.6) Example of using **MaxMetaSpaceSize** in java >

-XX:MaxMetaspaceSize=64m

Whenever this specified metaspace becomes full and there is no further space to allocate metaspace for java class metadata OutOfMemoryError: Metaspace is thrown.

7.7) How to avoid OutOfMemoryError - Metaspace in java?

You may increase the value of metaspace by passing the above VM argument (-XX:MaxMetaspaceSize).

7.8) **Tradeoff** between java heap and MetaSpace in java >

MetaSpace is allocated from the **same address spaces as the Java heap**.
Reducing the size of the **Java heap** will make **more space available for MetaSpace**.
This is only a correct **trade-off** if there is an excess of free space in the Java heap.

7.9) Example/program in java to generate OutOfMemoryError: **Metaspace** >

Before executing program pass this vm parameter in eclipse.
-XX:MaxMetaspaceSize=5m *(So* that we could easily produce OutOfMemoryError: Metaspace*)*

We will use javassist.ClassPool to create new classes.

```
import javassist.ClassPool;
/**
 *
 * Write a program which could throw
 * java.lang.OutOfMemoryError : Metaspace
 *
 */
public class OutOfMemoryErrorMetaspace {

   //ClassPool objects hold all the CtClasses.
   static ClassPool classPool = ClassPool.getDefault();

   public static void main(String[] args) throws Exception {
```

```
        for (int i = 0; i < 100000; i++) {
            //makeClass method - Creates a new class (or interface) from the given class file.
            Class clas = classPool.makeClass(
                    i + " outofmemory.OutOfMemoryErrorMetaspace ").toClass();
            //Print name of class loaded
            System.out.println(clas.getName());
        }
    }
}
```

Download Jar used in the program javassist.jar

OUTPUT of program >

```
0 outofmemory.OutOfMemoryErrorMetaspace
1 outofmemory.OutOfMemoryErrorMetaspace
2 outofmemory.OutOfMemoryErrorMetaspace
3 outofmemory.OutOfMemoryErrorMetaspace
4 outofmemory.OutOfMemoryErrorMetaspace
5 outofmemory.OutOfMemoryErrorMetaspace
6 outofmemory.OutOfMemoryErrorMetaspace
7 outofmemory.OutOfMemoryErrorMetaspace
8 outofmemory.OutOfMemoryErrorMetaspace
9 outofmemory.OutOfMemoryErrorMetaspace
10 outofmemory.OutOfMemoryErrorMetaspace
11 outofmemory.OutOfMemoryErrorMetaspace
12 outofmemory.OutOfMemoryErrorMetaspace
13 outofmemory.OutOfMemoryErrorMetaspace
Exception in thread "main" java.lang.OutOfMemoryError: Metaspace
```

So, from output we can clearly see that only 13 classes were loaded and then OutOfMemoryError: Metaspace was thrown.

The solution is simply to increase the Xmx parameter to ***-XX:MaxMetaspaceSize=512m*** *as we discussed above.*

For more details please read : OutOfMemoryError: Metaspace Solved

8) OutOfMemoryError: "request size bytes for reason. Out of swap space"

OutOfMemoryError: "request size bytes for reason. Out of swap space" indicates that **allocation from the native heap failed**.

8.1) What does "request size bytes for reason. Out of swap space" indicates in java?

It indicates the **size** (in bytes) **of the request that failed** and the **reason for the memory request.**
Usually the **reason is** the **name of the source** module **reporting** the **allocation failure**, although sometimes it is the actual reason.

8.2) What happens in OutOfMemoryError: "request size bytes for reason. Out of swap space"?

When OutOfMemoryError: "request size bytes for reason. Out of swap space" happens the

- JVM generates a **error log file** which contains information about the
  - threads,
  - processess,
  - system state at the time of the crash.

8.3) How to analyze OutOfMemoryError: “request size bytes for reason. Out of swap space” in java?

You can **analyze above information,** heap memory and memory map **to find out reason for OutOfMemoryError:** “request size bytes for reason. Out of swap space”.

You can also use different **OS** (operating system) **tools to analyze** OutOfMemoryError: “request size bytes for reason. Out of swap space”.

For more details please read : **Solving OutOfMemoryError: “request size bytes for reason. Out of swap space” in java**

Must read Related : How to monitor and analyze the garbage collection in 10 ways in java and Detecting and fixing memory leak in java

9) OutOfMemoryError: Compressed class space

9.1) Cause of OutOfMemoryError: Compressed class space in java >

If you working on 64-bit platforms a pointer to class metadata can be represented by a 32-bit offset (by using vm option UseCompressedClassPointers - This vm option is enabled by default).

If vm option is kept enabled then amount of space available for class metadata is fixed (i.e. specified by vm option)

If amount of space available for class metadata is exceeds CompressedClassSpaceSize, then java.lang.OutOfMemoryError **Compressed class space** is thrown.

9.2) Example of using **UseCompressedClassPointers vm option** in java >

-XX: CompressedClassSpaceSize=2g

It will set size of 2 gigabyte.

Now, if space available for class metadata exceeds 2 gigabyte, then java.lang.OutOfMemoryError **Compressed class space** is thrown.

9.3) You must also know that there different type of class metadata -

- **klass metadata** (only it is **stored** in CompressedClassSpaceSize) and
- **other metadata** (it is **not stored** in CompressedClassSpaceSize, it is stored in Metaspace).

For more details please read : **OutOfMemoryError: Compressed class space in java**

10) Exception in thread threadName : java.lang.OutOfMemoryError: reason stack_trace_with_native_method

10.1) What happens in **OutOfMemoryError: reason stack_trace_with_native_method**

Whenever this OutOfMemoryError is thrown >

- **a stack trace is printed**
- In this stack top frame is a native method

Then this OutOfMemoryError **indicates** that a **native method has encountered an allocation failure.**

10.2) When this **OutOfMemoryError: reason stack_trace_with_native_method** is detected**?**

In this OutOfMemoryError: “reason stack_trace_with_native_method” the **allocation failure is detected in** >

- Java Native Interface (JNI) or
- native method

rather than in the JVM code.

10.3) Action you should take in case of **OutOfMemoryError: reason stack_trace_with_native_method?**

Use native utilities of the OperatingSystem to diagnose the issue.

10.4) For solving this **OutOfMemoryError: reason stack_trace_with_native_method** is detected**?**

For solving use tools like >

- pmap and
- pstack

For more tools for various OS, please check Native Operating System Tools.

11) Out of memory:Kill process or sacrifice child

11.1) When does **OutOfMemoryError : kill process or sacrifice** child occurs?

OutOfMemoryError : kill process or sacrifice occurs when one of the process consumes too much virtual memory and makes OS unstable, than OS decides to kills that process.

11.2) Solution to **OutOfMemoryError : kill process or sacrifice** >

Increasing swap space can solve this OutOfMemoryError**.**

12) Exception in thread threadName : java.lang.**OutOfMemoryError: permgen**

12.1) When you are facing **OutOfMemoryError: permgen you need to** change **permgen size** in tomcat server?

Generally when we are facing java.lang.**OutOfMemoryError - Java permgen space**, then we need to change permgen size of tomcat or eclipse or JVM wherever you are facing this error.

12.2) How to set or **change permgen size** in tomcat server?

For setting permgen size in tomcat server you need to make changes values in the Tomcat Catalina start file. Change **CATALINA_OPTS** option in the file.

12.3) Where is exactly **catalina.bat** file **located**?

tomcatServerHome\bin\catalina.bat

12.4) How to set permgen size in tomcat server in **windows, linux and Mac** platform >

12.4.1) How to set permgen size in tomcat server in **windows** platform >

Open or create **setenv.bat** file (Location of setenv.bat file is tomcatHome\bin\setenv.bat)

set **CATALINA_OPTS**=-server **-Xms**512m **-Xmx**1024m **-XX:PermSize**=512m **-XX:MaxPermSize**=1024m

Best practices while **setting permgen size in tomcat server in windows platform >**
You should not modify tomcatServerHome\bin**catalina.bat**. You should create a new file in tomcatServerHome\bin**setenv.bat** to keep your custom environment configurations separate from default tomcat server configurations.

Also learn Adding vm argument in tomcat in eclipse.

12.4.2) How to set permgen size in tomcat server in **Linux** platform >

Open or create **setenv.sh** file (Location of setenv.bat file is tomcatHome\bin\setenv.sh)

export **CATALINA_OPTS**="$**CATALINA_OPTS**=-server **-Xms**512m **-Xmx**1024m **-XX:PermSize**=512m **-XX:MaxPermSize**=1024m

Best practices while **setting permgen size in tomcat server in linux platform >**
You should not modify tomcatServerHome\bin**catalina.sh**. You should create a new file in tomcatServerHome\bin**setenv.sh** to keep your custom environment configurations separate from default tomcat server configurations.

12.4.3) How to set permgen size in tomcat server in **Mac** OS platform >

Open or create setenv.sh file (Location of setenv.bat file is tomcatHome\bin\setenv.sh)

export **CATALINA_OPTS**="$**CATALINA_OPTS**=-server **-Xms**512m **-Xmx**1024m **-XX:PermSize**=512m **-XX:MaxPermSize**=1024m"

Best practices while **setting permgen size in tomcat server in Mac OS platform >**
You should not modify tomcatServerHome\bin**catalina.sh**. You should create a new file in tomcatServerHome\bin**setenv.sh** to keep your custom environment configurations separate from default tomcat server configurations.

12.5) You may use following **VM** (JVM) PARAMETERS to set up **permgen memory** (or permanent generation or permanent space) >

Read: How to write java program to pass VM parameters through CMD

Learn how to pass **vmargs** (VM parameters) to java program in **eclipse**?

-XX:PermSize: It's is initial value of Permanent Space which is allocated at startup of JVM.

Example1 of using **-XX:PermSize** VM (JVM) option in java >
java -XX:PermSize=1g MyJavaProgram
It will set initial value of Permanent Space as 512 gigabyte to JVM

-XX:MaxPermSize: It's maximum value of Permanent Space that JVM can allot up to.

Examples of using -XX:MaxPermSize VM option in java >

java -XX:MaxPermSize=512m MyJavaProgram
It will set maximum value of Permanent Space as 512 megabytes to JVM

For more explanation and example - Read : What are **-XX:PermSize** and **-XX:MaxPermSize** with Differences

What is the maximum perm gen size value you can set on your system, if you aren't sure about the system configurations?
Try out for -XX:MaxPermSize=256m, if it works then try -XX:MaxPermSize=512m, if it works then try -XX:MaxPermSize=1024m and so on. Be cautious before going beyond 8g.

For more details, please read :

> **OutOfMemoryError: Permgen space - How to set or change permgen size in tomcat server, eclipse?**

>How to pass VM argument to tomcat in eclipse

>What are **-XX:PermSize** and **-XX:MaxPermSize** JVM parameters with examples in java | Differences

5) Summary -

So in this tutorial we learned about different OutOfMemoryError in java and how to solve them with example and programs.

Having any doubt? or you liked the tutorial! Please comment in below section.

*Please express your love by liking **JavaMadeSoEasy.com** (**JMSE**) on **facebook**, following on **google+** or Twitter.*

RELATED LINKS>

Read: How to write java program to pass VM parameters through CMD

>**JVM (java virtual machine)** in detail in java

>How **Garbage Collection (GC) works internally** in detail in java - BEST EXPLANATION EVER

Important VM parameters >

>What are **-Xms** and **-Xmx** JVM parameters in java, And differences between them with examples

>**-Xmn** JVM parameters in java with examples - Setting young generation size

>What are **-XX:NewSize** and **-XX:MaxNewSize** JVM parameters in java

>What are **-XX:PermSize** and **-XX:MaxPermSize** JVM parameters with examples in java | Differences

>Solve java.lang.**OutOfMemoryError** : **unable to create new native Thread - Xss** JVM option

More VM parameters >

>How to use **-verbose:gc** VM argument

>**-Xverify option** in java

Apache tomcat server,outOfMemory and Garbage collection in java >

>How to set or change permgen size in tomcat server, eclipse?

>How to set, change, increase or decrease heap size in tomcat server and eclipse to avoid OutOfMemoryError ?

>How to pass VM argument to tomcat in eclipse

Pass VM para through CMD, eclipse to java program and to Apache tomcat >

>How to write java **program** to **pass VM**/JVM **parameters through CMD**

>How to **pass** vmArgs(JVM parameters) to java **program** in **eclipse**

>How to pass VM argument to tomcat in eclipse

Labels: Core Java Exceptions OutOfMemoryError

Must read for you :

Newer Post Home Older Post