FrOSCon
Free and Open Source Software Conference

anderScore
trust in competence

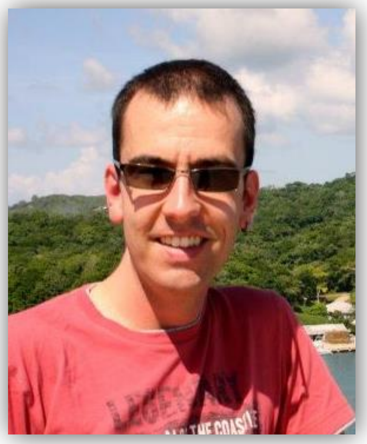
Back to Basics

Wissenswertes aus java.lang.*

Christian Robert
Senior Software Engineer
christian.robert@anderscore.com

anderScore GmbH
Frankenwerft 35
50667 Köln
www.anderScore.com

Christian Robert



- Senior Software Engineer
- Seit über 10 Jahren Entwicklung
  im e-Commerce und Consulting-Umfeld
- Schwerpunkt auf Entwicklung von
  pragmatischen Architekturkonzepten

anderScore GmbH

- Projektorientierte Entwicklung von
  Individualsoftware
- Agiles Vorgehen, kurze Zyklen,
  qualitativ hochwertige Ergebnisse

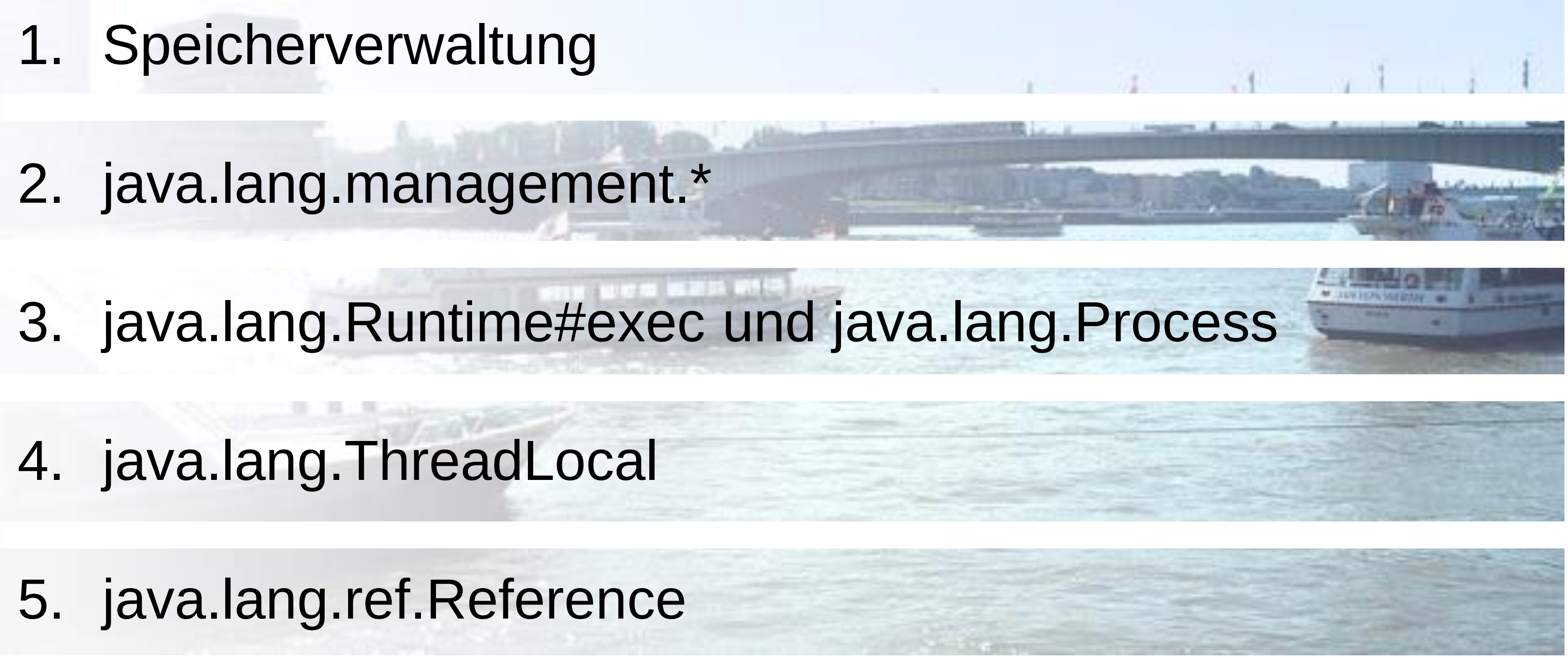
Christian Robert

- Senior Software Engineer
- Seit über 10 Jahren Entwicklung im e-Com
- Schwerp pragmati

anderSc

- Projektor Individua
- Agiles Vo qualitativ hochwertige Ergebnisse

Agenda



1. Speicherverwaltung
2. java.lang.management.*
3. java.lang.Runtime#exec und java.lang.Process
4. java.lang.ThreadLocal
5. java.lang.ref.Reference
6. java.lang.Object#finalize

JAVA VIRTUAL MACHINE

Speicherverwaltung

java.lang.OutOfMemoryError
und manuelle Garbage Collection

java.lang.OutOfMemoryError

```java
public void foo() {
  try {
    this.doExpensiveOperation();
  } catch(OutOfMemoryError e) {
    System.gc();
    this.doExpensiveOperation();
  }
}
```

> *"Calling the gc method **suggests** that the Java Virtual Machine expend effort toward recycling unused objects […] When control returns […], the Java Virtual Machine **has made a best effort** to reclaim space from all discarded objects."*
>
> *-- Javadoc java.lang.Runtime*

java.lang.OutOfMemoryError

```
public void foo() {
  try {
    this.doExpensiveOperation();
  } catch(OutOfMemoryError e) {
    System.gc();
    this.doExpensiveOperation();
  }
}
```

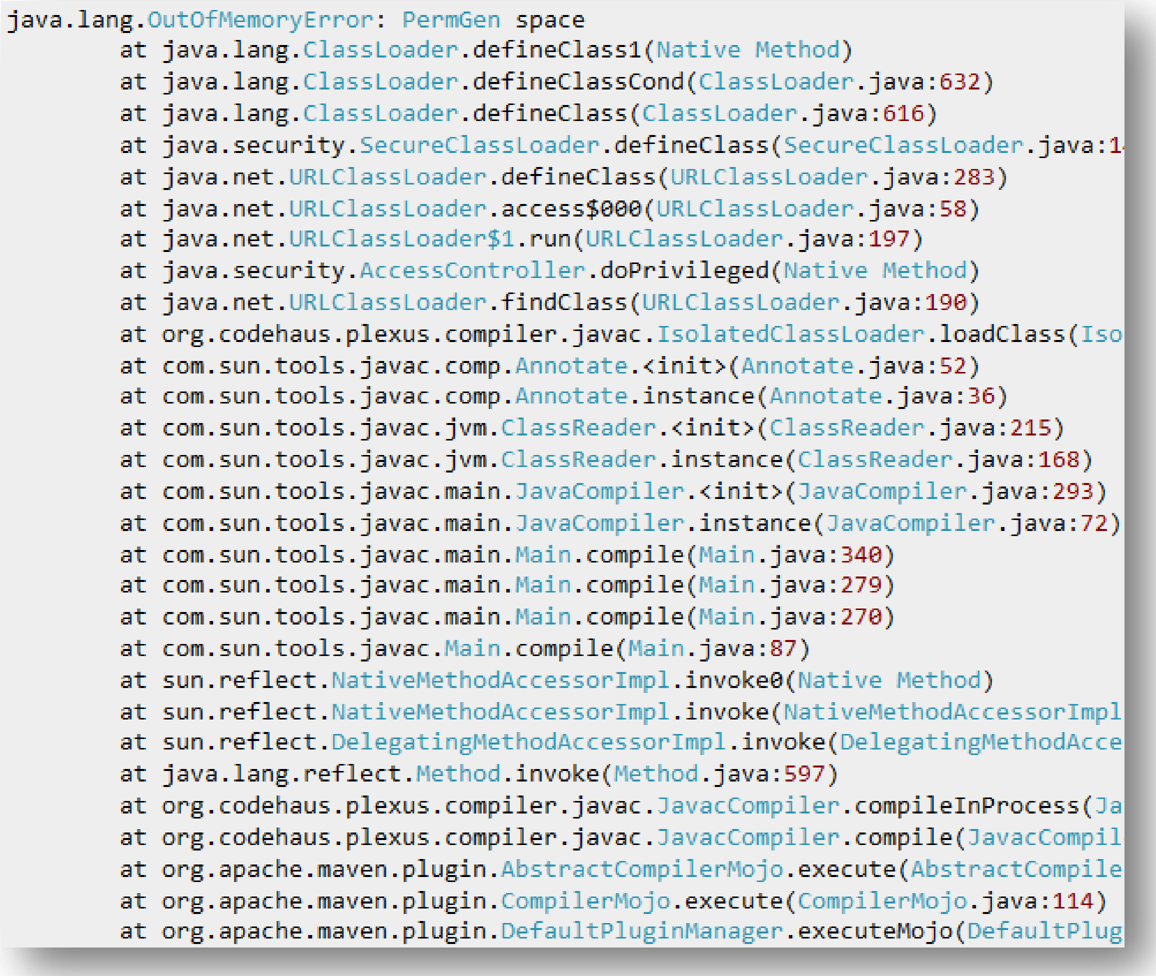
> *"Thrown when the Java Virtual Machine cannot allocate an object because it is out of memory, **and no more memory could be made available by the garbage collector.**"*
>
> *-- Javadoc java.lang.OutOfMemoryError*

> *"An Error is a subclass of Throwable that indicates serious problems that a reasonable application **should not try to catch**."*
>
> *-- Javadoc java.lang.Error*

java.lang.OutOfMemoryError

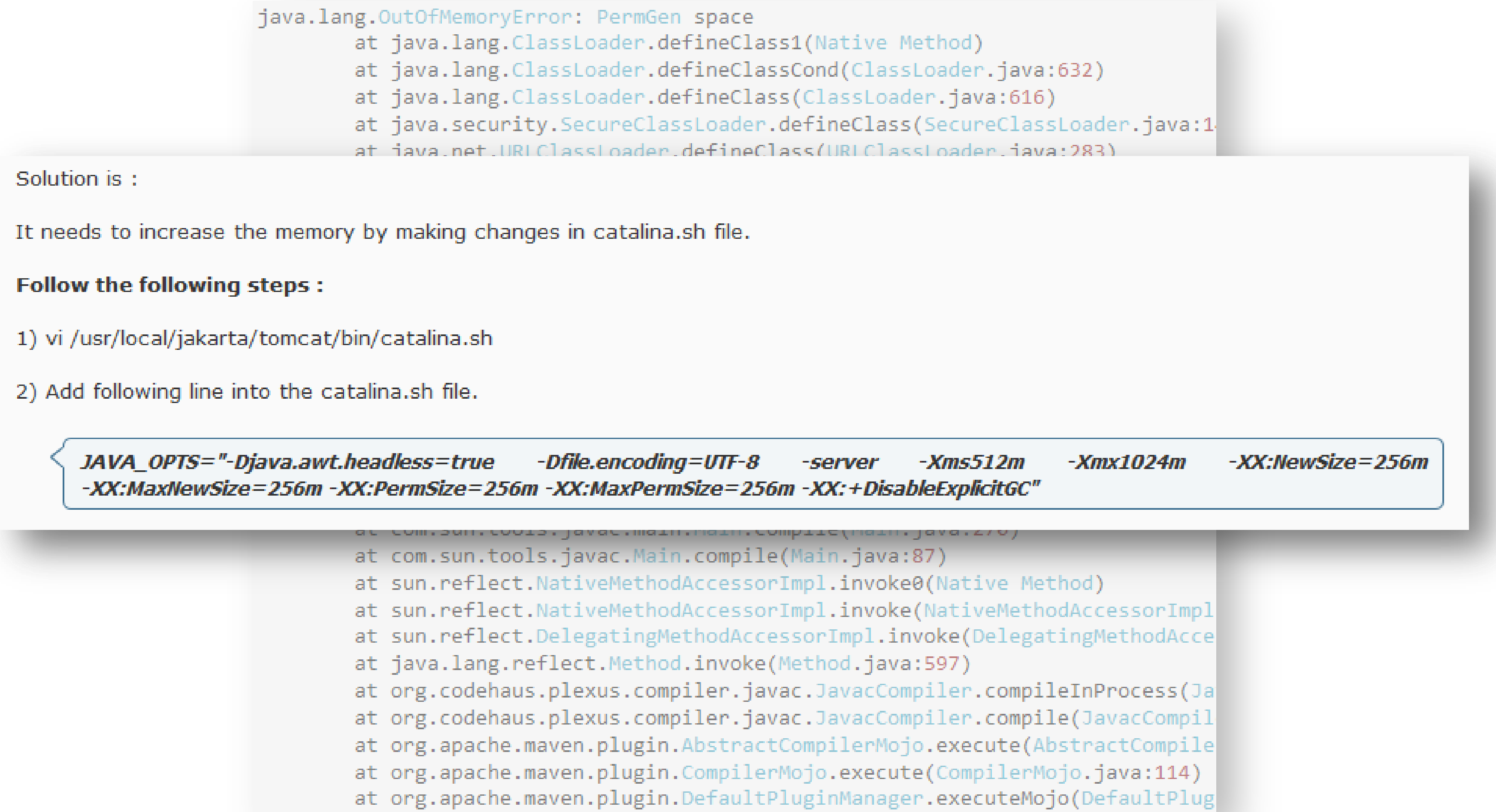
PermGen Space / Metaspace

OutOfMemoryError: PermGen Space



```
java.lang.OutOfMemoryError: PermGen space
        at java.lang.ClassLoader.defineClass1(Native Method)
        at java.lang.ClassLoader.defineClassCond(ClassLoader.java:632)
        at java.lang.ClassLoader.defineClass(ClassLoader.java:616)
        at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:1
        at java.net.URLClassLoader.defineClass(URLClassLoader.java:283)
        at java.net.URLClassLoader.access$000(URLClassLoader.java:58)
        at java.net.URLClassLoader$1.run(URLClassLoader.java:197)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:190)
        at org.codehaus.plexus.compiler.javac.IsolatedClassLoader.loadClass(Iso
        at com.sun.tools.javac.comp.Annotate.<init>(Annotate.java:52)
        at com.sun.tools.javac.comp.Annotate.instance(Annotate.java:36)
        at com.sun.tools.javac.jvm.ClassReader.<init>(ClassReader.java:215)
        at com.sun.tools.javac.jvm.ClassReader.instance(ClassReader.java:168)
        at com.sun.tools.javac.main.JavaCompiler.<init>(JavaCompiler.java:293)
        at com.sun.tools.javac.main.JavaCompiler.instance(JavaCompiler.java:72)
        at com.sun.tools.javac.main.Main.compile(Main.java:340)
        at com.sun.tools.javac.main.Main.compile(Main.java:279)
        at com.sun.tools.javac.main.Main.compile(Main.java:270)
        at com.sun.tools.javac.Main.compile(Main.java:87)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAcce
        at java.lang.reflect.Method.invoke(Method.java:597)
        at org.codehaus.plexus.compiler.javac.JavacCompiler.compileInProcess(Ja
        at org.codehaus.plexus.compiler.javac.JavacCompiler.compile(JavacCompil
        at org.apache.maven.plugin.AbstractCompilerMojo.execute(AbstractCompile
        at org.apache.maven.plugin.CompilerMojo.execute(CompilerMojo.java:114)
        at org.apache.maven.plugin.DefaultPluginManager.executeMojo(DefaultPlug
```

OutOfMemoryError: PermGen Space



```
java.lang.OutOfMemoryError: PermGen space
        at java.lang.ClassLoader.defineClass1(Native Method)
        at java.lang.ClassLoader.defineClassCond(ClassLoader.java:632)
        at java.lang.ClassLoader.defineClass(ClassLoader.java:616)
        at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:1
        at java.net.URLClassLoader.defineClass(URLClassLoader.java:283)
```

Solution is :

It needs to increase the memory by making changes in catalina.sh file.

Follow the following steps :

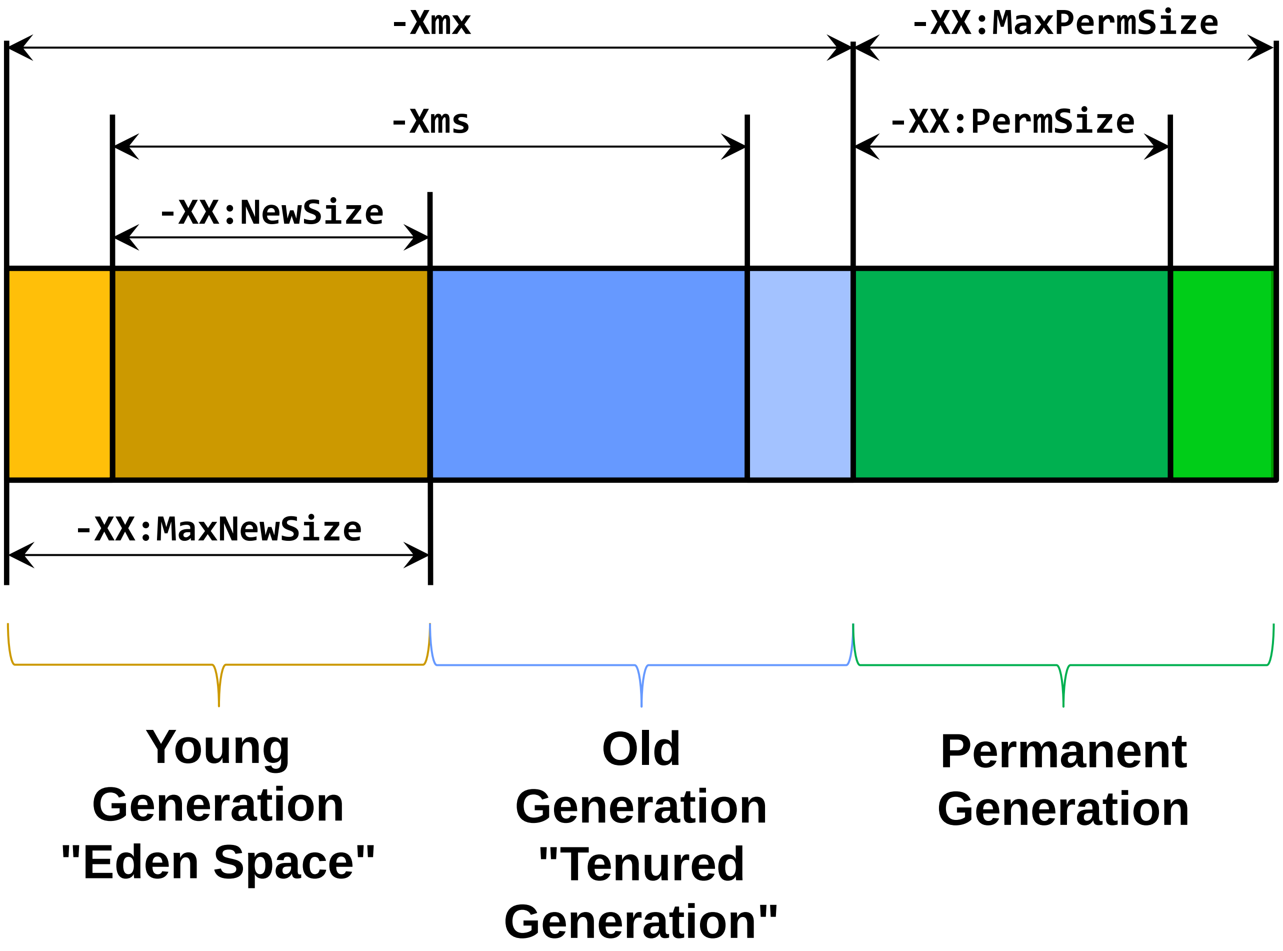
1) vi /usr/local/jakarta/tomcat/bin/catalina.sh

2) Add following line into the catalina.sh file.

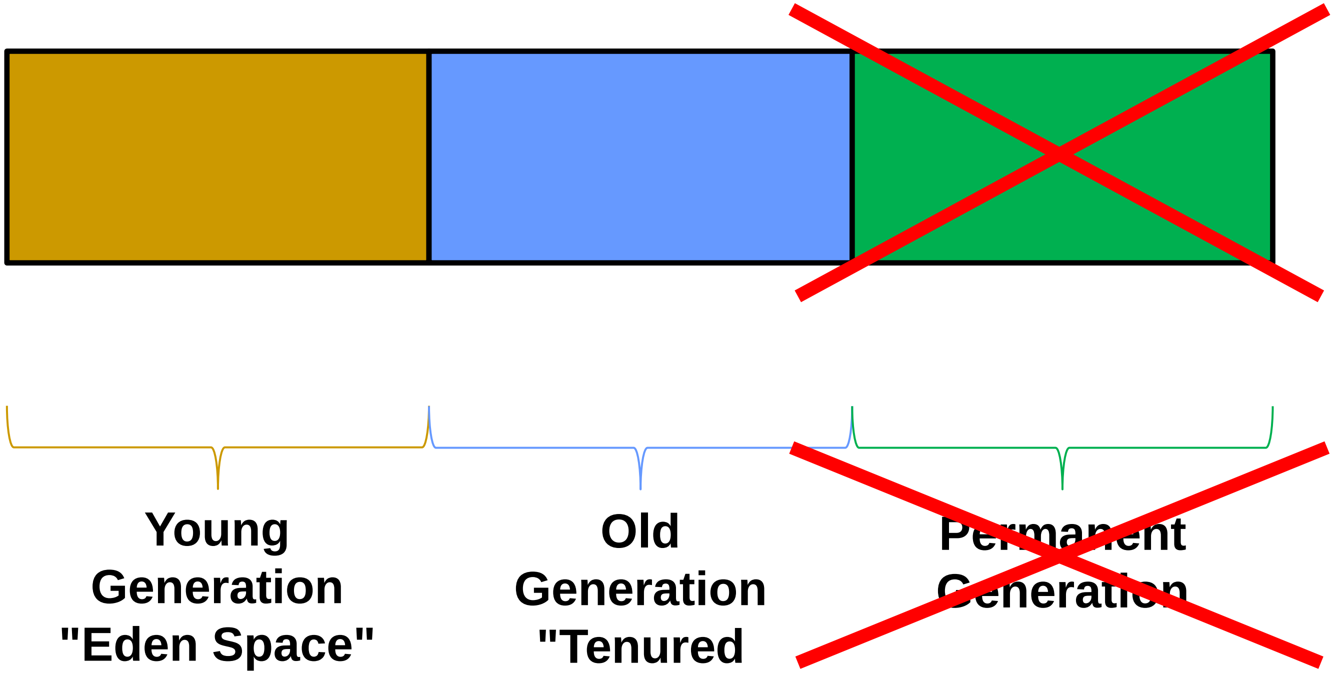
```
JAVA_OPTS="-Djava.awt.headless=true -Dfile.encoding=UTF-8 -server -Xms512m -Xmx1024m -XX:NewSize=256m
-XX:MaxNewSize=256m -XX:PermSize=256m -XX:MaxPermSize=256m -XX:+DisableExplicitGC"
```

```
        at com.sun.tools.javac.main.Main.compile(Main.java:276)
        at com.sun.tools.javac.Main.compile(Main.java:87)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAcce
        at java.lang.reflect.Method.invoke(Method.java:597)
        at org.codehaus.plexus.compiler.javac.JavacCompiler.compileInProcess(Ja
        at org.codehaus.plexus.compiler.javac.JavacCompiler.compile(JavacCompil
        at org.apache.maven.plugin.AbstractCompilerMojo.execute(AbstractCompile
        at org.apache.maven.plugin.CompilerMojo.execute(CompilerMojo.java:114)
        at org.apache.maven.plugin.DefaultPluginManager.executeMojo(DefaultPlug
```

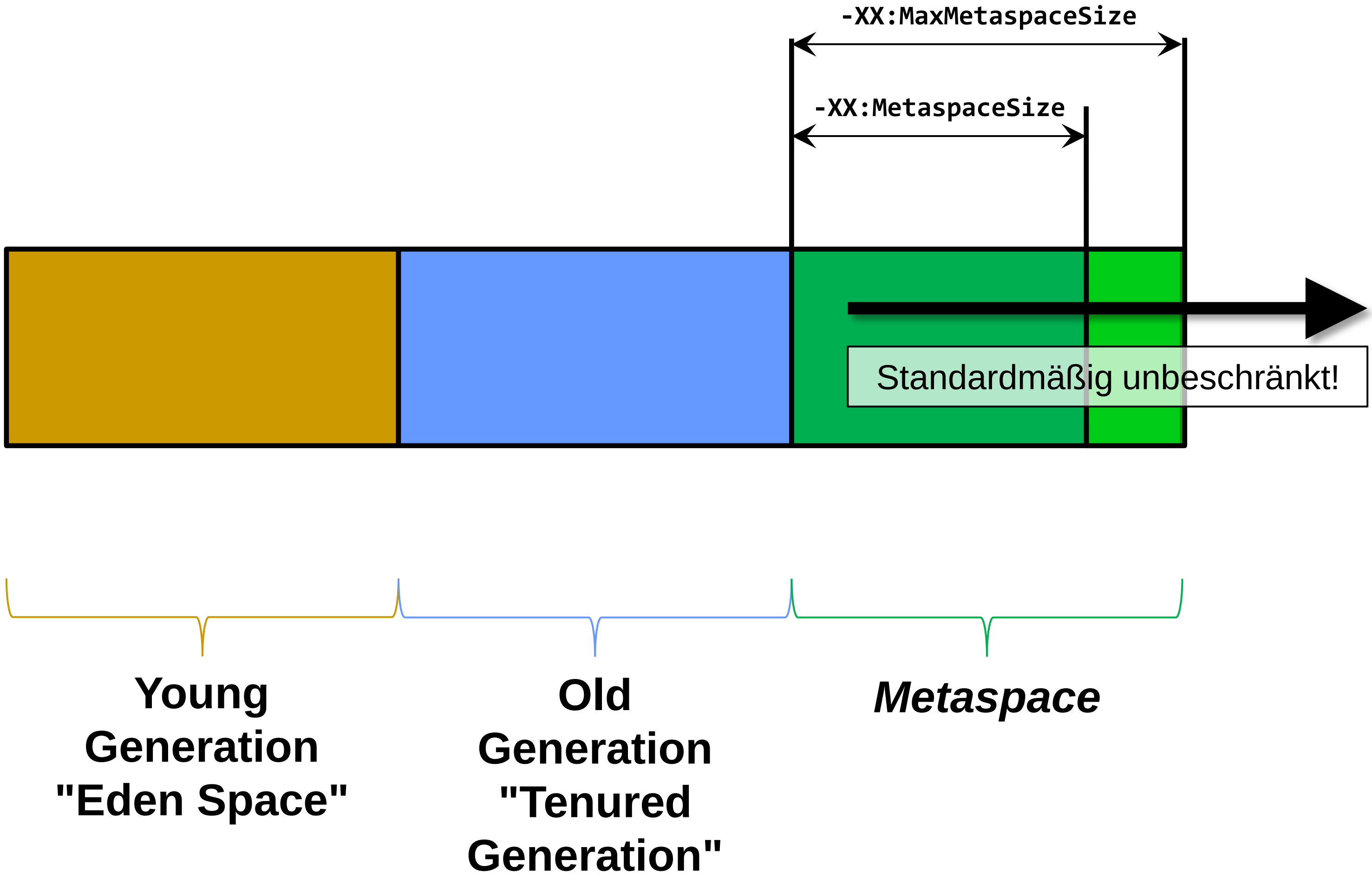
Java Virtual Machine - Speicher



Java Virtual Machine - Speicher - Java 8


Young Generation "Eden Space"
Old Generation "Tenured Generation"
Permanent Generation

Java Virtual Machine - Speicher - Java 8


-XX:MaxMetaspaceSize
-XX:MetaspaceSize
Standardmäßig unbeschränkt!
Young Generation "Eden Space"
Old Generation "Tenured Generation"
Metaspace

java.lang.management.*

Informationen direkt aus der VM

`java.lang.management.ManagementFactory`

```
  1:      ClassLoadingMXBean
  1:      MemoryMXBean
  1:      ThreadMXBean
  1:      RuntimeMXBean
  1:      OperatingSystemMXBean
  1:      PlattformLoggingMXBean
  0..1:   CompilationMXBean
  1..n:   GarbageCollectorMXBean
  1..n:   MemoryManagerMXBean
  1..n:   MemoryPoolMXBean
  1..n:   BufferPoolMXBean
```

Informationen direkt aus der VM

java.lang.management.MemoryMXBean

Beispielinhalt

```java
public class MemoryMXBeanDemo {

  public static void main(String[] args) {
    MemoryMXBean bean = ManagementFactory.getMemoryMXBean();
    System.out.println("Heap:\n" + bean.getHeapMemoryUsage());
  }

}
```

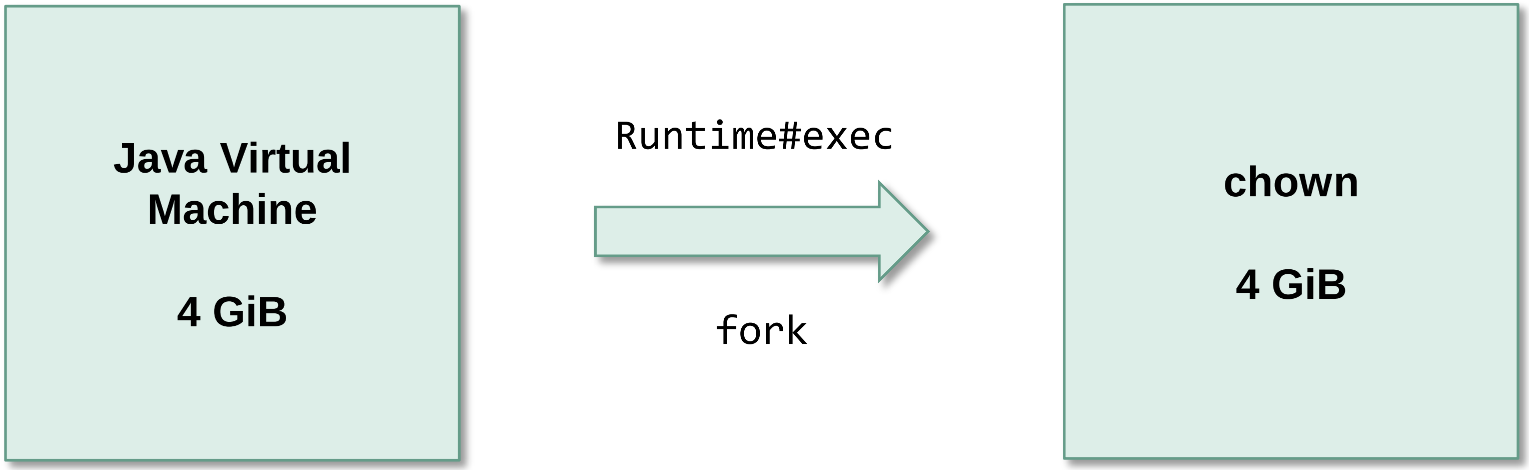
```
Heap:
init = 16777216(16384K) used = 386632(377K)
committed = 16252928(15872K) max = 259522560(253440K)
```

java.lang.Runtime#exec

java.lang.Runtime#exec

```java
public void uploadContent(byte[] content,
  String fileName) throws IOException {

   File uploadedFile = uploadContentToFile(content, fileName);

   Runtime.getRuntime().exec(new String[] {
     "chown",
     "test:test",
     uploadedFile.getAbsolutePath()
   });

 }
```

java.lang.Runtime#exec



> *"The fork operation creates a separate address space for the child. **The child process has an exact copy of all the memory segments of the parent process**, though if copy-on-write semantics are implemented actual physical memory **may** not be assigned"*
>
> *-- http://en.wikipedia.org/wiki/Fork_(operating_system)*

java.lang.Runtime#exec

- bis Java 6: fork
- Java 7 / Java 8: fork / vfork

> *"[...] we are currently using **vfork() on Linux***
> *and **fork() on other Unix systems** [...]"*
>
> *-- OpenJDK 7: UNIXProcess_md.c*

java.lang.Runtime#exec

Ein-/Ausgabe

java.lang.Runtime#exec stdout / stderr

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

  Process process = Runtime.getRuntime().exec(
    new String[] {
      "/bin/foo" // Liefert viel Output an stdout
    }
  );

  process.waitFor();

}
```

java.lang.Runtime#exec stdout / stderr

> *"All its standard I/O (i.e. stdin, stdout, stderr) operations will be redirected to the parent process, where they can be accessed via the streams obtained using the methods getOutputStream(), getInputStream(), and getErrorStream().*
>
> *[…]*
>
> *Failure to promptly write the input stream or read the output stream of the subprocess* ***may cause the subprocess to block****, or even deadlock."*
>
> *-- Javadoc java.lang.Runtime*

java.lang.Runtime#exec stdout / stderr

```
Process process = Runtime.getRuntime().exec(
  new String[] {
    "/bin/foo" // Liefert viel Output an stdout
  }
);

try(InputStream stdout = process.getInputStream()) {
  for(int data = stdout.read(); data > -1; data = stdout.read()) {
    doSomething(data);
  }
}

process.waitFor();
```

Hier nur Behandlung von stdout
Was ist mit stderr?

java.lang.Runtime#exec stdout / stderr

```
ProcessBuilder processBuilder = new ProcessBuilder(
  "/bin/foo" // Liefert viel Output an stdout
);
processBuilder.redirectErrorStream(true);
Process process = processBuilder.start();

try(InputStream stdout = process.getInputStream()) {
  for(int data = stdout.read(); data > -1; data = stdout.read()) {
    doSomething(data);
  }
}

process.waitFor();
```

java.lang.ThreadLocal

java.lang.ThreadLocal

```java
public class ThreadLocalExample {

  ThreadLocal<Integer> counter = new ThreadLocal<>();

  public void example() {
    for(int i=0; i < 10; i++) {
      new Thread(new Runnable() {
        @Override public void run() {
          ThreadLocalExample.this.exampleInThread();
        }
      }).start();
    }
  }

  synchronized void exampleInThread() {
    for(int i=0; i < 10; i++) {
      Integer oldValue = counter.get();
      counter.set(oldValue == null ? 1 : (oldValue + 1));
    }
    System.out.println(Thread.currentThread() + " " + counter.get());
  }

}
```

```
Thread[Thread-0,5,main] 10
Thread[Thread-1,5,main] 10
Thread[Thread-2,5,main] 10
Thread[Thread-3,5,main] 10
Thread[Thread-4,5,main] 10
Thread[Thread-5,5,main] 10
Thread[Thread-6,5,main] 10
Thread[Thread-7,5,main] 10
Thread[Thread-8,5,main] 10
Thread[Thread-9,5,main] 10
```

ThreadLocal - Anwendungsfall

```
public class ExampleServlet extends BaseServletFromFramework {










  @Override
  protected void doSomethingWithinFramework(FrameworkObject o) {
    String currentUser = ???
    if("admin".equals(currentUser)) {
      // Do some stuff
    }
  }

}
```

ThreadLocal - Anwendungsfall - Idee

```java
public class ExampleServlet extends BaseServletFromFramework {

  private String myUser = null;

  @Override
  protected void service(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
    this.myUser = req.getRemoteUser();
    super.service(req, resp);
  }

  @Override
  protected void doSomethingWithinFramework(FrameworkObject o) {
    String currentUser = this.myUser;
    if("admin".equals(currentUser)) {
      // Do some stuff
    }
  }

}
```

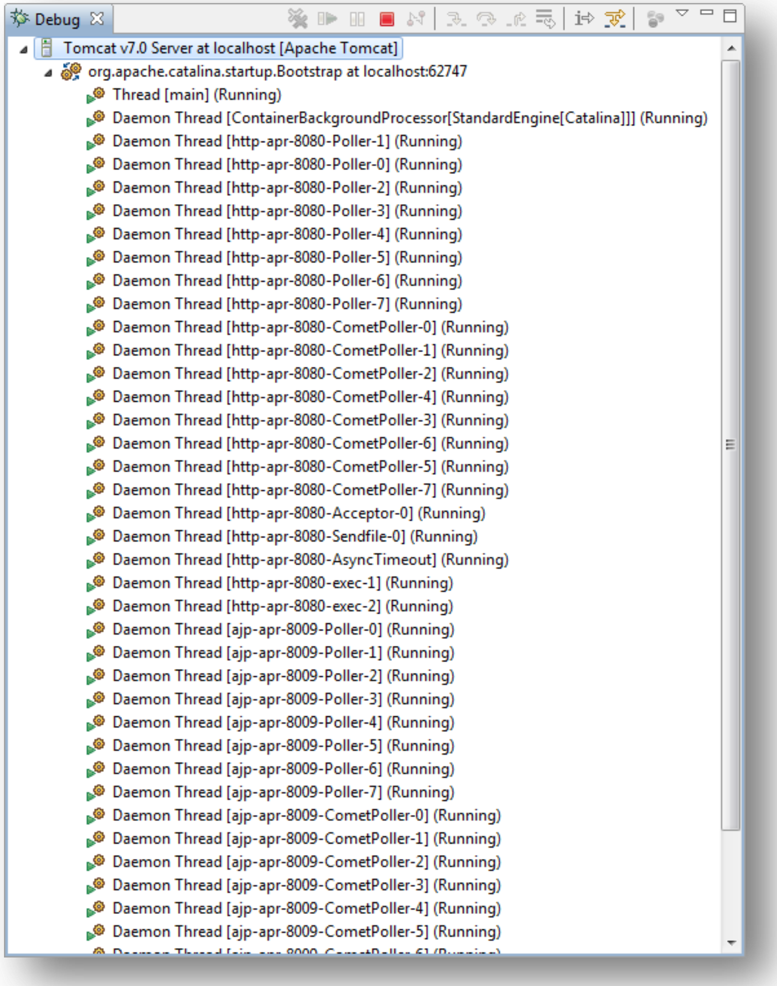
ThreadLocal - Anwendungsfall - Idee

```
public class ExampleServlet extends BaseServletFromFramework {

 private String myUser = null;

 @Override
 protected void service(HttpServl       HttpServletResponse resp)
  throws ServletException, IOEx
  this.myUser = req.getRemoteU
  super.service(req, resp);
 }

 @Override
 protected void doSomethi               ect o) {
  String currentUser =
  if("admin".equals(c
   // Do some stuff
  }
 }

}
```

ThreadLocal - Anwendungsfall

```java
public class ExampleServlet extends BaseServletFromFramework {

  private ThreadLocal<String> myUser = new ThreadLocal<>();

  @Override
  protected void service(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
    this.myUser.set(req.getRemoteUser());
    super.service(req, resp);
  }

  @Override
  protected void doSomethingWithinFramework(FrameworkObject o) {
    String currentUser = this.myUser.get();
    if("admin".equals(currentUser)) {
      // Do some stuff
    }
  }

}
```

java.lang.ThreadLocal



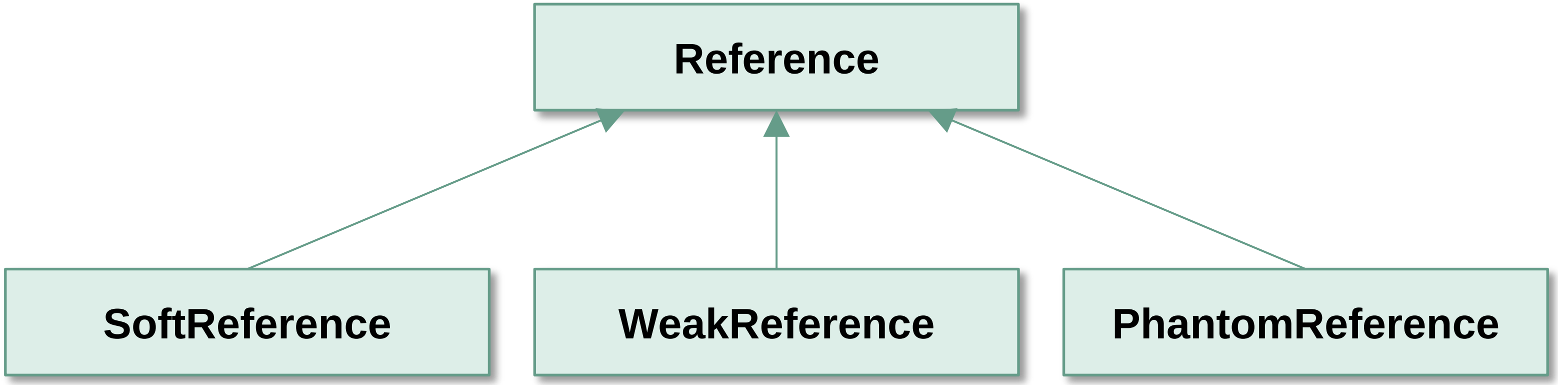
```
Tomcat v7.0 Server at localhost [Apache Tomcat]
  org.apache.catalina.startup.Bootstrap at localhost:62747
    Thread [main] (Running)
    Daemon Thread [ContainerBackgroundProcessor[StandardEngine[Catalina]]] (Running)
    Daemon Thread [http-apr-8080-Poller-1] (Running)
    Daemon Thread [http-apr-8080-Poller-0] (Running)
    Daemon Thread [http-apr-8080-Poller-2] (Running)
    Daemon Thread [http-apr-8080-Poller-3] (Running)
    Daemon Thread [http-apr-8080-Poller-4] (Running)
    Daemon Thread [http-apr-8080-Poller-5] (Running)
    Daemon Thread [http-apr-8080-Poller-6] (Running)
    Daemon Thread [http-apr-8080-Poller-7] (Running)
    Daemon Thread [http-apr-8080-CometPoller-0] (Running)
    Daemon Thread [http-apr-8080-CometPoller-1] (Running)
    Daemon Thread [http-apr-8080-CometPoller-2] (Running)
    Daemon Thread [http-apr-8080-CometPoller-4] (Running)
    Daemon Thread [http-apr-8080-CometPoller-3] (Running)
    Daemon Thread [http-apr-8080-CometPoller-6] (Running)
    Daemon Thread [http-apr-8080-CometPoller-5] (Running)
    Daemon Thread [http-apr-8080-CometPoller-7] (Running)
    Daemon Thread [http-apr-8080-Acceptor-0] (Running)
    Daemon Thread [http-apr-8080-Sendfile-0] (Running)
    Daemon Thread [http-apr-8080-AsyncTimeout] (Running)
    Daemon Thread [http-apr-8080-exec-1] (Running)
    Daemon Thread [http-apr-8080-exec-2] (Running)
    Daemon Thread [ajp-apr-8009-Poller-0] (Running)
    Daemon Thread [ajp-apr-8009-Poller-1] (Running)
    Daemon Thread [ajp-apr-8009-Poller-2] (Running)
    Daemon Thread [ajp-apr-8009-Poller-3] (Running)
    Daemon Thread [ajp-apr-8009-Poller-4] (Running)
    Daemon Thread [ajp-apr-8009-Poller-5] (Running)
    Daemon Thread [ajp-apr-8009-Poller-6] (Running)
    Daemon Thread [ajp-apr-8009-Poller-7] (Running)
    Daemon Thread [ajp-apr-8009-CometPoller-0] (Running)
    Daemon Thread [ajp-apr-8009-CometPoller-1] (Running)
    Daemon Thread [ajp-apr-8009-CometPoller-2] (Running)
    Daemon Thread [ajp-apr-8009-CometPoller-3] (Running)
    Daemon Thread [ajp-apr-8009-CometPoller-4] (Running)
    Daemon Thread [ajp-apr-8009-CometPoller-5] (Running)
    Daemon Thread [ajp-apr-8009-CometPoller-6] (Running)
```

- ThreadLocal Werte bleiben bestehen solange der Thread aktiv ist
- Bei Threads im Pool kann dies sehr sehr lange sein!

java.lang.ref.Reference

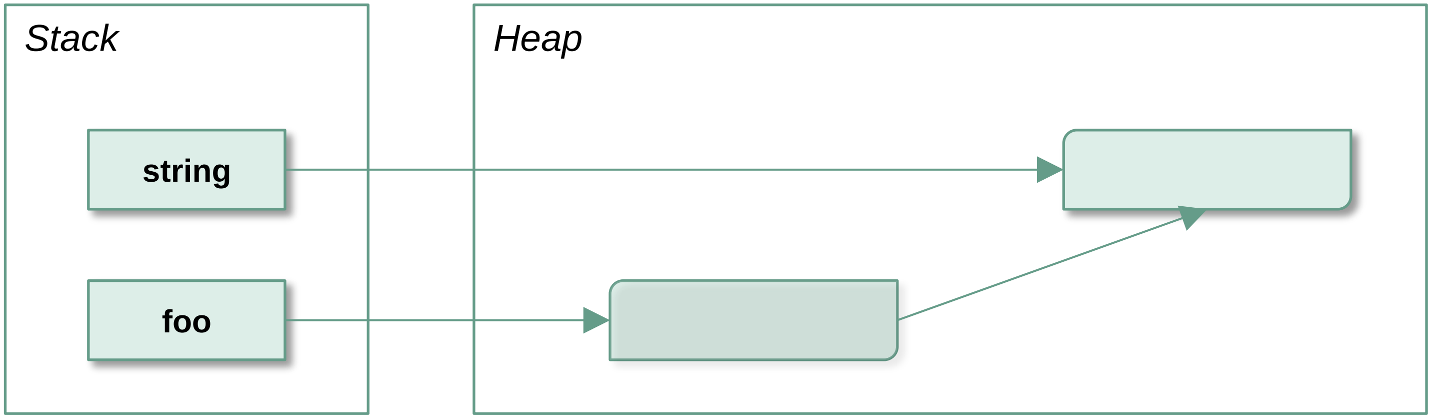
java.lang.ref.Reference

> *"Abstract base class for reference objects. This class defines the operations common to all reference objects. Because reference objects are implemented in close cooperation with the garbage collector, this class may not be subclassed directly."*
>
> *-- Javadoc java.lang.ref.Reference*



java.lang.ref.Reference

```
String string = "Hello world";
Foo foo = new Foo(string);
foo = null;
this.doSomeOtherStuff();
```

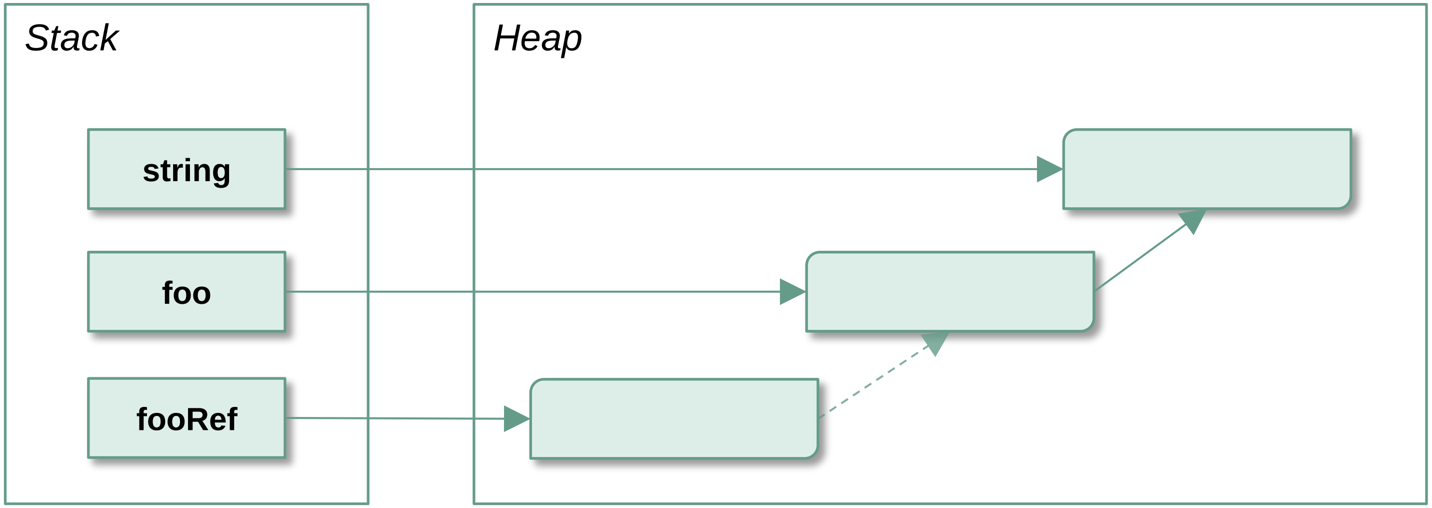


java.lang.ref.Reference

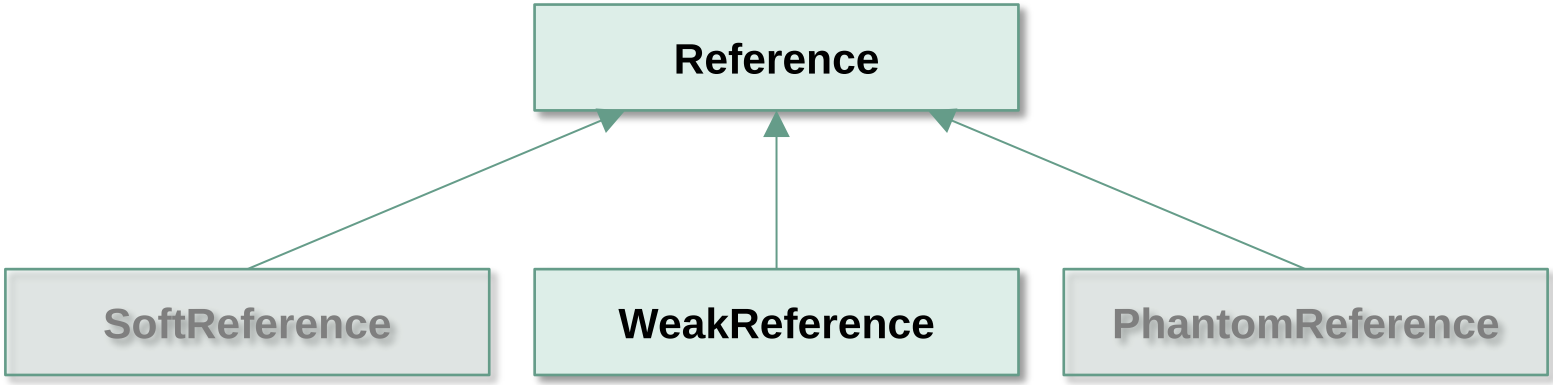
```java
String string = "Hello world";
Foo foo = new Foo(string);
WeakReference<Foo> fooRef = new WeakReference<>(foo);
System.out.println("1: " + fooRef.get());

foo = null;
this.doSomeOtherStuff();

System.out.println("2: " + fooRef.get());
```

```
1: Foo@620968f9
```

```
2: null
```



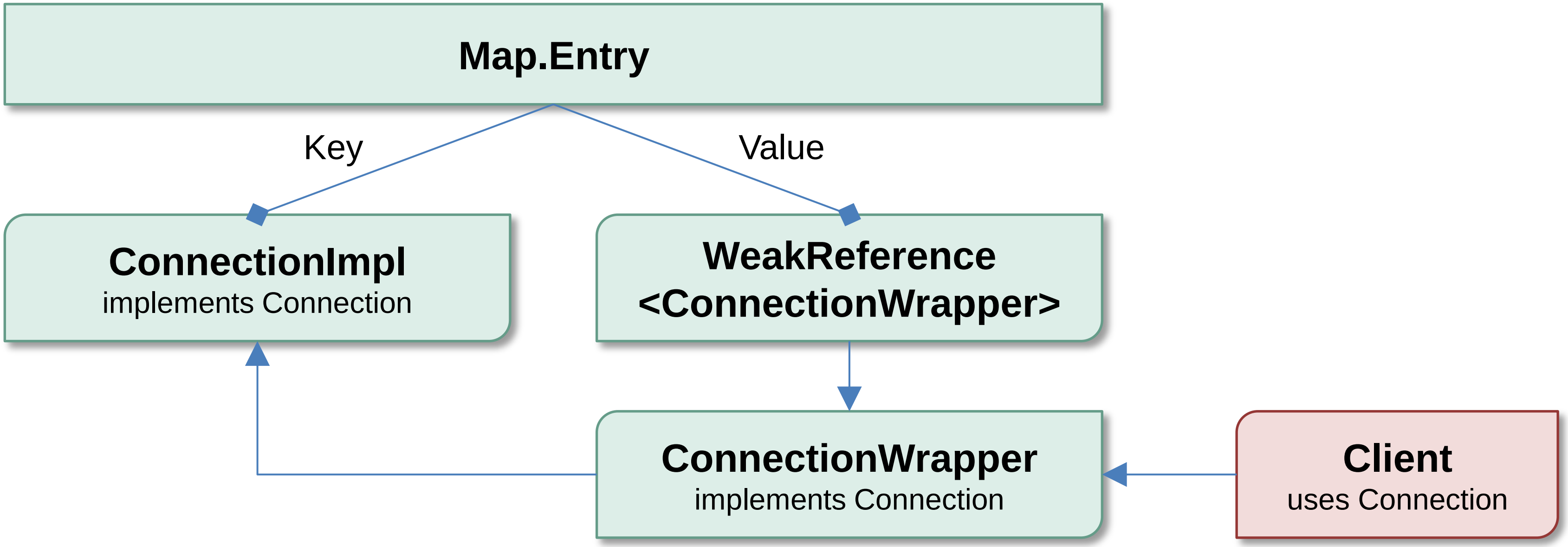
java.lang.ref.WeakReference



> *"Weak reference objects, which do not prevent their referents from being made finalizable, finalized, and then reclaimed. "*
>
> *-- Javadoc java.lang.ref.WeakReference*

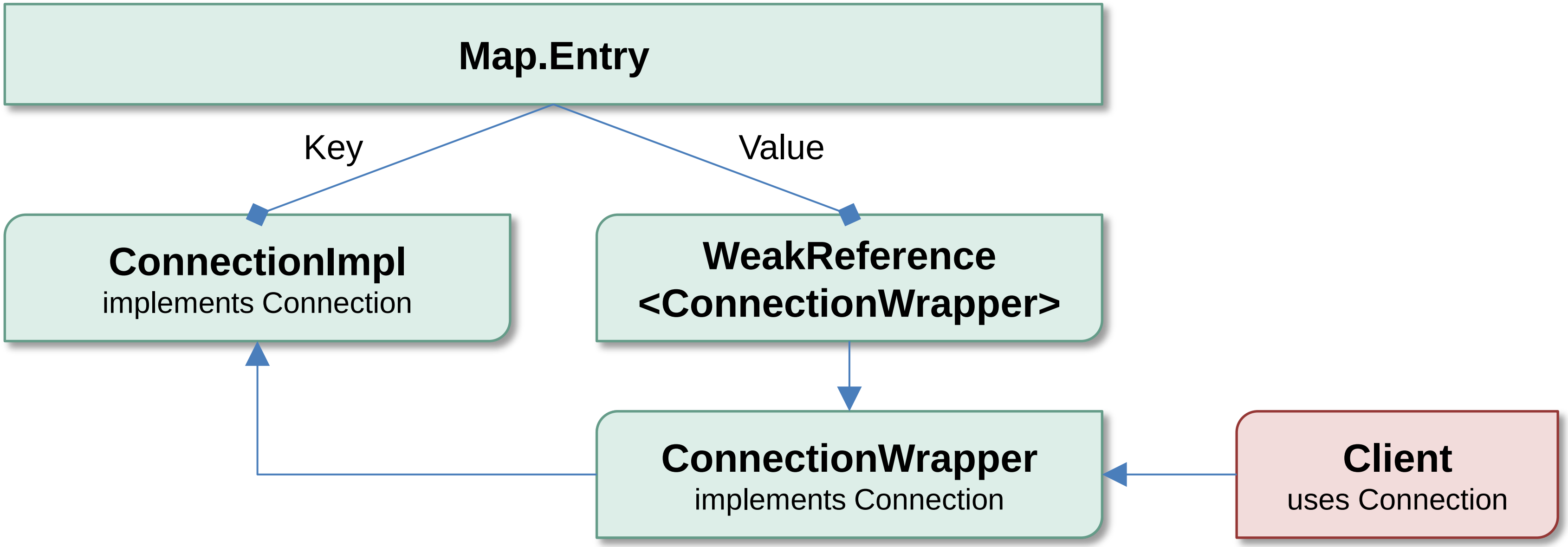
WeakReference: Anwendungsfall

- Connection Pool
  - "Merken" von ausgegebenen und noch verwendeten Connections in einer internen Map

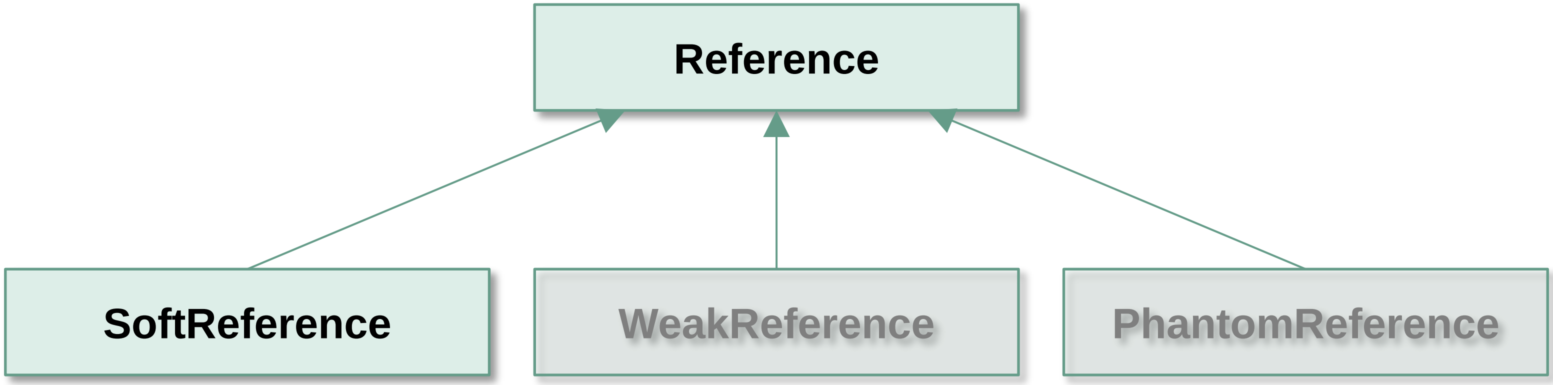


WeakReference: Anwendungsfall

- Connection Pool
  - "Merken" von ausgegebenen und noch verwendeten Connections in einer internen Map



java.lang.ref.SoftReference



Reference

SoftReference

WeakReference

PhantomReference

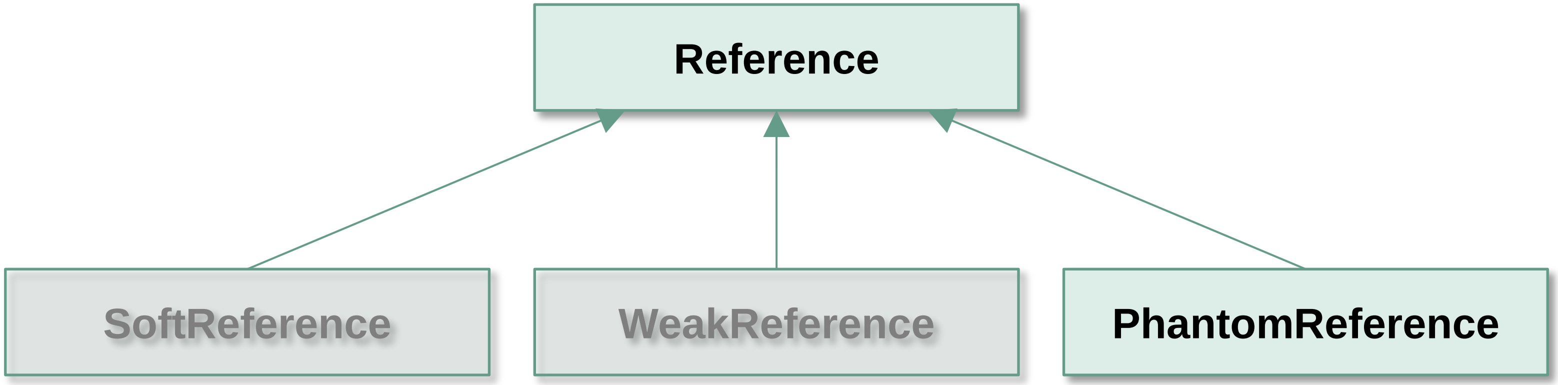
"Soft reference objects, which are cleared ***at the discretion of the garbage collector in response to memory demand****. Soft references are most often used to implement memory-sensitive caches."*

-- Javadoc java.lang.ref.SoftReference

SoftReference: Anwendungsfall

```java
public class SoftReferenceExample {

  private Map<String, SoftReference<Expensive>> cache =
    new HashMap<>();

  public Expensive lookup(String key) {
    SoftReference<Expensive> ref = this.cache.get(key);
    Expensive expensive = ref == null ? null : ref.get();
    if(expensive == null) {
      expensive = Expensive.createExpensiveObject();
      this.cache.put(key, new SoftReference<Expensive>(expensive));
    }
    return expensive;
  }

}
```

java.lang.ref.PhantomReference



"Phantom reference objects, which are enqueued after the collector determines that their referents may otherwise be reclaimed. Phantom references are most often used for scheduling pre-mortem cleanup actions in a more flexible way than is possible with the Java finalization mechanism."

-- Javadoc java.lang.ref.PhantomReference

PhantomReference: Achtung!

```java
Foo foo = new Foo("Hello world");
PhantomReference<Foo> fooRef =
  new PhantomReference<>(foo);
System.out.println("1: " + fooRef.get());

foo = null;
this.doSomeOtherStuff();

System.out.println("2: " + fooRef.get());
```

```
1: null
```

```
2: null
```

> *"[…] the referent of a phantom reference may not be retrieved: The get method of a phantom reference always returns null. "*
>
> *-- Javadoc java.lang.ref.PhantomReference*

PhantomReference: Anwendungsfall

- Aufräumarbeiten, die im Finalizer nicht ausgeführt werden können (oder sollen)
- Off-Heap-Speicherverwaltung (java.nio.DirectByteBuffer)
- Sicheres Löschen von temporären Dateien (Tomcat / Wicket)
- Ableiten von PhantomReference und Hinzufügen von für das Aufräumen wichtiger Properties

PhantomReference: Anwendungsfall

```java
public class MyPhantomReference extends PhantomReference<Foo> {

  private File file = null;

  public MyPhantomReference(Foo referent, ReferenceQueue<? super Foo> q,
    File file) {
    super(referent, q);
    this.setFile(bar);
  }

  public File getFile() {
    return this.file;
  }
  private void setFile(File file) {
    this.file = file;
  }

}
```

java.lang.ref.ReferenceQueue

```java
Foo foo = new Foo("Hello world");
ReferenceQueue<Foo> refQueue = new ReferenceQueue<>();
WeakReference<Foo> fooRef = new WeakReference<>(foo, refQueue);

foo = null;

this.doSomeOtherStuff();

Reference<Foo> refQueueRef = refQueue.remove();
System.out.println("1: " + refQueueRef);
System.out.println("2: " + refQueueRef.get());
```

```
1: java.lang.ref.WeakReference@62b103dd
2: null
```

java.lang.Object#finalize

java.lang.Object#finalize

```
public class ResurrectObject {

  private static Set<Object> objects = new HashSet<>();

  @Override
  protected void finalize() throws Throwable {
    objects.add(this);
    super.finalize();
  }

}
```

java.lang.Object#finalize

> *"Called by the garbage collector on an object when garbage collection determines that there are* ***no more references to the object****. A subclass overrides the finalize method to dispose of system resources or to perform other cleanup. […] After the finalize method has been invoked for an object, no further action is taken until the Java virtual machine has* ***again*** *determined that there is no longer any means by which this object can be accessed […] at which point the object may be discarded.* ***The finalize method is never invoked more than once by a Java virtual machine for any given object.****"*
>
> *-- Javadoc java.lang.Object*

- Überschriebene finalize Methode bedingt zwei Garbage Collector Durchläufe zum Entfernen des Objektes!

Fragen?
Anmerkungen?

Christian Robert
Senior Software Engineer
christian.robert@anderscore.com

anderScore GmbH
Frankenwerft 35
50667 Köln
www.anderScore.com

FrOSCon
Free and Open Source Software Conference

anderScore
trust in competence

Vielen Dank!

Christian Robert
Senior Software Engineer
christian.robert@anderscore.com

anderScore GmbH
Frankenwerft 35
50667 Köln
www.anderScore.com