| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/MemoryMXBean.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/lang/management/MemoryManagerMXBean.html)   [**NEXT CLASS**](http://docs.google.com/java/lang/management/MemoryNotificationInfo.html) | [**FRAMES**](http://docs.google.com/index.html?java/lang/management/MemoryMXBean.html)    [**NO FRAMES**](http://docs.google.com/MemoryMXBean.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

## **java.lang.management**

Interface MemoryMXBean

public interface **MemoryMXBean**

The management interface for the memory system of the Java virtual machine.

A Java virtual machine has a single instance of the implementation class of this interface. This instance implementing this interface is an [MXBean](http://docs.google.com/ManagementFactory.html#MXBean) that can be obtained by calling the [ManagementFactory.getMemoryMXBean()](http://docs.google.com/java/lang/management/ManagementFactory.html#getMemoryMXBean()) method or from the [platform MBeanServer](http://docs.google.com/java/lang/management/ManagementFactory.html#getPlatformMBeanServer()) method.

The ObjectName for uniquely identifying the MXBean for the memory system within an MBeanServer is:

[java.lang:type=Memory](http://docs.google.com/java/lang/management/ManagementFactory.html#MEMORY_MXBEAN_NAME)

#### Memory

The memory system of the Java virtual machine manages the following kinds of memory:

#### 1. Heap

The Java virtual machine has a *heap* that is the runtime data area from which memory for all class instances and arrays are allocated. It is created at the Java virtual machine start-up. Heap memory for objects is reclaimed by an automatic memory management system which is known as a *garbage collector*.

The heap may be of a fixed size or may be expanded and shrunk. The memory for the heap does not need to be contiguous.

#### 2. Non-Heap Memory

The Java virtual machine manages memory other than the heap (referred as *non-heap memory*).

The Java virtual machine has a *method area* that is shared among all threads. The method area belongs to non-heap memory. It stores per-class structures such as a runtime constant pool, field and method data, and the code for methods and constructors. It is created at the Java virtual machine start-up.

The method area is logically part of the heap but a Java virtual machine implementation may choose not to either garbage collect or compact it. Similar to the heap, the method area may be of a fixed size or may be expanded and shrunk. The memory for the method area does not need to be contiguous.

In addition to the method area, a Java virtual machine implementation may require memory for internal processing or optimization which also belongs to non-heap memory. For example, the JIT compiler requires memory for storing the native machine code translated from the Java virtual machine code for high performance.

#### Memory Pools and Memory Managers

[Memory pools](http://docs.google.com/java/lang/management/MemoryPoolMXBean.html) and [memory managers](http://docs.google.com/java/lang/management/MemoryManagerMXBean.html) are the abstract entities that monitor and manage the memory system of the Java virtual machine.

A memory pool represents a memory area that the Java virtual machine manages. The Java virtual machine has at least one memory pool and it may create or remove memory pools during execution. A memory pool can belong to either the heap or the non-heap memory.

A memory manager is responsible for managing one or more memory pools. The garbage collector is one type of memory manager responsible for reclaiming memory occupied by unreachable objects. A Java virtual machine may have one or more memory managers. It may add or remove memory managers during execution. A memory pool can be managed by more than one memory manager.

#### Memory Usage Monitoring

Memory usage is a very important monitoring attribute for the memory system. The memory usage, for example, could indicate:

* the memory usage of an application,
* the workload being imposed on the automatic memory management system,
* potential memory leakage.

The memory usage can be monitored in three ways:

* Polling
* Usage Threshold Notification
* Collection Usage Threshold Notification

Details are specified in the [MemoryPoolMXBean](http://docs.google.com/java/lang/management/MemoryPoolMXBean.html) interface.

The memory usage monitoring mechanism is intended for load-balancing or workload distribution use. For example, an application would stop receiving any new workload when its memory usage exceeds a certain threshold. It is not intended for an application to detect and recover from a low memory condition.

#### Notifications

This MemoryMXBean is a [NotificationEmitter](http://docs.google.com/javax/management/NotificationEmitter.html) that emits two types of memory [notifications](http://docs.google.com/javax/management/Notification.html) if any one of the memory pools supports a [usage threshold](http://docs.google.com/MemoryPoolMXBean.html#UsageThreshold) or a [collection usage threshold](http://docs.google.com/MemoryPoolMXBean.html#CollectionThreshold) which can be determined by calling the [MemoryPoolMXBean.isUsageThresholdSupported()](http://docs.google.com/java/lang/management/MemoryPoolMXBean.html#isUsageThresholdSupported()) and [MemoryPoolMXBean.isCollectionUsageThresholdSupported()](http://docs.google.com/java/lang/management/MemoryPoolMXBean.html#isCollectionUsageThresholdSupported()) methods.

* [usage threshold exceeded notification](http://docs.google.com/java/lang/management/MemoryNotificationInfo.html#MEMORY_THRESHOLD_EXCEEDED) - for notifying that the memory usage of a memory pool is increased and has reached or exceeded its  [usage threshold](http://docs.google.com/MemoryPoolMXBean.html#UsageThreshold) value.
* [collection usage threshold exceeded notification](http://docs.google.com/java/lang/management/MemoryNotificationInfo.html#MEMORY_COLLECTION_THRESHOLD_EXCEEDED) - for notifying that the memory usage of a memory pool is greater than or equal to its  [collection usage threshold](http://docs.google.com/MemoryPoolMXBean.html#CollectionThreshold) after the Java virtual machine has expended effort in recycling unused objects in that memory pool.

The notification emitted is a [Notification](http://docs.google.com/javax/management/Notification.html) instance whose [user data](http://docs.google.com/javax/management/Notification.html#setUserData(java.lang.Object)) is set to a [CompositeData](http://docs.google.com/javax/management/openmbean/CompositeData.html) that represents a [MemoryNotificationInfo](http://docs.google.com/java/lang/management/MemoryNotificationInfo.html) object containing information about the memory pool when the notification was constructed. The CompositeData contains the attributes as described in [MemoryNotificationInfo](http://docs.google.com/java/lang/management/MemoryNotificationInfo.html#from(javax.management.openmbean.CompositeData)).

#### NotificationEmitter

The MemoryMXBean object returned by [ManagementFactory.getMemoryMXBean()](http://docs.google.com/java/lang/management/ManagementFactory.html#getMemoryMXBean()) implements the [NotificationEmitter](http://docs.google.com/javax/management/NotificationEmitter.html) interface that allows a listener to be registered within the MemoryMXBean as a notification listener. Below is an example code that registers a MyListener to handle notification emitted by the MemoryMXBean.

class MyListener implements javax.management.NotificationListener {  
 public void handleNotification(Notification notif, Object handback) {  
 // handle notification  
 ....  
 }  
 }  
  
 MemoryMXBean mbean = ManagementFactory.getMemoryMXBean();  
 NotificationEmitter emitter = (NotificationEmitter) mbean;  
 MyListener listener = new MyListener();  
 emitter.addNotificationListener(listener, null, null);

**Since:** 1.5 **See Also:** [JMX Specification.](http://docs.google.com/javax/management/package-summary.html),  [Ways to Access MXBeans](http://docs.google.com/package-summary.html#examples)

| **Method Summary** | |
| --- | --- |
| void | [**gc**](http://docs.google.com/java/lang/management/MemoryMXBean.html#gc())()            Runs the garbage collector. |
| [MemoryUsage](http://docs.google.com/java/lang/management/MemoryUsage.html) | [**getHeapMemoryUsage**](http://docs.google.com/java/lang/management/MemoryMXBean.html#getHeapMemoryUsage())()            Returns the current memory usage of the heap that is used for object allocation. |
| [MemoryUsage](http://docs.google.com/java/lang/management/MemoryUsage.html) | [**getNonHeapMemoryUsage**](http://docs.google.com/java/lang/management/MemoryMXBean.html#getNonHeapMemoryUsage())()            Returns the current memory usage of non-heap memory that is used by the Java virtual machine. |
| int | [**getObjectPendingFinalizationCount**](http://docs.google.com/java/lang/management/MemoryMXBean.html#getObjectPendingFinalizationCount())()            Returns the approximate number of objects for which finalization is pending. |
| boolean | [**isVerbose**](http://docs.google.com/java/lang/management/MemoryMXBean.html#isVerbose())()            Tests if verbose output for the memory system is enabled. |
| void | [**setVerbose**](http://docs.google.com/java/lang/management/MemoryMXBean.html#setVerbose(boolean))(boolean value)            Enables or disables verbose output for the memory system. |

| **Method Detail** |
| --- |

### getObjectPendingFinalizationCount

int **getObjectPendingFinalizationCount**()

Returns the approximate number of objects for which finalization is pending.

**Returns:**the approximate number objects for which finalization is pending.

### getHeapMemoryUsage

[MemoryUsage](http://docs.google.com/java/lang/management/MemoryUsage.html) **getHeapMemoryUsage**()

Returns the current memory usage of the heap that is used for object allocation. The heap consists of one or more memory pools. The used and committed size of the returned memory usage is the sum of those values of all heap memory pools whereas the init and max size of the returned memory usage represents the setting of the heap memory which may not be the sum of those of all heap memory pools.

The amount of used memory in the returned memory usage is the amount of memory occupied by both live objects and garbage objects that have not been collected, if any.

**MBeanServer access**:

The mapped type of MemoryUsage is CompositeData with attributes as specified in [MemoryUsage](http://docs.google.com/java/lang/management/MemoryUsage.html#from(javax.management.openmbean.CompositeData)).

**Returns:**a [MemoryUsage](http://docs.google.com/java/lang/management/MemoryUsage.html) object representing the heap memory usage.

### getNonHeapMemoryUsage

[MemoryUsage](http://docs.google.com/java/lang/management/MemoryUsage.html) **getNonHeapMemoryUsage**()

Returns the current memory usage of non-heap memory that is used by the Java virtual machine. The non-heap memory consists of one or more memory pools. The used and committed size of the returned memory usage is the sum of those values of all non-heap memory pools whereas the init and max size of the returned memory usage represents the setting of the non-heap memory which may not be the sum of those of all non-heap memory pools.

**MBeanServer access**:

The mapped type of MemoryUsage is CompositeData with attributes as specified in [MemoryUsage](http://docs.google.com/java/lang/management/MemoryUsage.html#from(javax.management.openmbean.CompositeData)).

**Returns:**a [MemoryUsage](http://docs.google.com/java/lang/management/MemoryUsage.html) object representing the non-heap memory usage.

### isVerbose

boolean **isVerbose**()

Tests if verbose output for the memory system is enabled.

**Returns:**true if verbose output for the memory system is enabled; false otherwise.

### setVerbose

void **setVerbose**(boolean value)

Enables or disables verbose output for the memory system. The verbose output information and the output stream to which the verbose information is emitted are implementation dependent. Typically, a Java virtual machine implementation prints a message whenever it frees memory at garbage collection.

Each invocation of this method enables or disables verbose output globally.

**Parameters:**value - true to enable verbose output; false to disable. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and the caller does not have ManagementPermission("control").

### gc

void **gc**()

Runs the garbage collector. The call gc() is effectively equivalent to the call:

System.gc()

**See Also:**[System.gc()](http://docs.google.com/java/lang/System.html#gc())

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/MemoryMXBean.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/lang/management/MemoryManagerMXBean.html)   [**NEXT CLASS**](http://docs.google.com/java/lang/management/MemoryNotificationInfo.html) | [**FRAMES**](http://docs.google.com/index.html?java/lang/management/MemoryMXBean.html)    [**NO FRAMES**](http://docs.google.com/MemoryMXBean.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).