Memory leaks in Java

In C, programmers totally control allocation and deallocation of dynamically created objects. And if a programmer does not destroy objects, [memory leak happens in C](https://www.geeksforgeeks.org/what-is-memory-leak-how-can-we-avoid/),

Java does automatic Garbage collection. However there can be situations where garbage collector does not collect objects because there are references to them. There might be situations where where an application creates lots of objects and does not use them. Just because every objects has valid references, [garbage collector in Java](https://www.geeksforgeeks.org/garbage-collection-java/)can’t destroys the objects. Such types of useless objects are called as Memory leaks. If allocated memory goes beyond limit, program will be terminated by rising OutOfMemoryError. Hence if an object is no longer required, it is highly recommended to make that object eligible for garbage collector. Otherwise We should use some tools that do memory management to identifies useless objects or memeory leaks like:

* HP OVO
* HP J METER
* JProbe
* IBM Tivoli

|  |
| --- |
| // Java Program to illustrate memory leaks  import java.util.Vector;  public class MemoryLeaksDemo  {      public static void main(String[] args)      {          Vector v = new Vector(214444);          Vector v1 = new Vector(214744444);          Vector v2 = new Vector(214444);          System.out.println("Memory Leaks");      }  } |

/////

## What is HPjmeter?

HPjmeter is a free, supported Java performance analysis tool that graphically displays important metrics related to the performance of Java applications. The metrics displayed include:

* Method call counts
* Method execution times
* Method call graph
* Number of allocated objects
* Allocation sites
* Object reference graph
* Lock contention
* and many others

HPjmeter is designed to display the collected metrics to allow you to easily identify performance bottlenecks and quickly tune your Java applications. Metrics can be collected using either -Xrunhprof (JVMPI, available on all platforms) or -Xeprof (Low-Intrusion Extended Profiling available on HP's Unix, HP-UX).

## How To Use HPjmeter

HPjmeter reads the profile data produced by the Java VM (or a profiling agent). Therefore, using HPjmeter always requires two steps:

1. Run your application with profiling enabled (for example, add -Xrunhprof:<options> to the command line)
2. Run HPjmeter to analyze the file produced when the profiled program exits.

HPjmeter can read a large number of profile data formats. The metrics available in HPjmeter depend on the contents of the profile data file, and can be controlled by the profiler agent options. In most cases, the options for analysis of the application execution time will be different from the options used for heap analysis. For example, to collect the CPU time data using sampling when running MyApp program, one can use:

java -Xrunhprof:cpu=samples,thread=y,depth=12,cutoff=0 MyApp

The hprof profiler is available for most Java 2 platforms, although its robustness may vary. For more information about the hprof options please see the on-line help in HPjmeter's **Help -> User's Guide**, "Profiling with -Xrunhprof."

Download HPjmeter (no cost) from the [Hewlett-Packard Java website](http://www.hp.com/java). Just unzip the downloaded bundle to get the jar file, and invoke it with:

java -mx128m -jar HPjmeter.jar