Dealing with “java.lang.OutOfMemoryError: PermGen space” Error

Perm Gen Space is piece of memory allocated in running JVM based application

Perm Gen Error is from the Out of memory error

JVM works with 2 memory types

* Stack and Heap

Stack

* Store Primitive types and Object address

Heap

* Stores the value of Objects
* Sores static content of Java application(static methods ,static variables ,reference to static objects and class files )
* Default size is 82 MB for 64 bit version , 64 MB for 32 bit JVM
* Classloaders will be main cause of this Perm Gen Error
* Problem araises when application gets undeployed
* Server container keeps reference of one or more classes
* And classloader itself cannot be garbage collected.
* Another cause is application threads running even after application gets undeployed

[JVM parameter](https://www.baeldung.com/jvm-parameters) –XX:MaxPermSize

The –Xnoclassgc JVM parameter, when specified at the start of the JVM, explicitly removes class files from the list of entities to be trashed. In application servers and with modern frameworks that load and unload classes thousands of times per application's lifecycle, this can bring to very fast exhaustion of the PermGen space.

In older versions of Java, classes are a permanent part of the heap, meaning that once loaded, they remain in memory. By specifying the CMSClassUnloadingEnabled (for Java 1.5 or CMSPermGenSweepingEnabled for Java 1.6) JVM Parameter, it is possible to enable the garbage collection of classes. If we happen to be working with Java 1.6, UseConcMarkSweepGC must also be set to true. Otherwise, the CMSClassUnloadingEnabled argument would be ignored.

[Permgen has been entirely replaced by Metaspace](https://www.baeldung.com/java-permgen-metaspace), which has an automatically resizable space and an advanced feature that enables the cleaning of dead classes.

It should be trivial noticing that in case of a memory leak, none of the solutions provided can suffice. Memory will finish, no matter how great the size. Even Metaspace has a limited amount of memory available. Deep HEAP analysis is sometimes the only solution and can be conducted with tools like VisualGC or JPROFILER.