**Java Memory Management**

Diagram

Description automatically generated

There are few memory types in java,

* **Heap Memory**
* **Stack Memory**
* **Metaspace**

Stack memory size is very less than the heap memory. This whole memory will be lived in the JRE.

**Heap Memory**

There are two parts inside the Heap area called **Young** **Generation**and **Old Generation**. There are 3 parts inside the Young generation area called Eden and two survivor( Survivor1, Survivor2 ) memories.

In the heap area **objects, JRE Classes**will be stored. The Garbage Collector will run only on Heap Memory. ( GC will destroy objects which don’t has references and frees memories )

**Memory Pool** — This stays in the Heap area. In the memory pool **immutable objects,**the **String pool** will be stored.

1. When Eden memory completes( full ), Minor Garbage Collector( Minor GC ) will perform and Minor GC will move old memory parts(variables) to Survivor memories.
2. Then after some time, the Young generation may be filled, Then Old objects will be moved to the Old Generation.
3. Then Old Generation will be filled and there is no space available in the Heap area. Then Major Garbage Collector( Major GC ) will be performed. This Major GC will take some longer time than Minor GC.

Minor GC always performs in the Young Generation and the major GC always performs in the Old Generation.

**Stack Memory**

This is used to execute the threads. This contains methods-specific values( l**ocal variables**),**object references and method blocks**. This uses the Last In First Out( LIFO ) way to free the memory(which means the newest/latest memory locations will be destroyed).

* Local variables — local variables are variables that define in a method, including method parameters. Java Stack memory only contains the byte, short, char, int, long, float and double(primitives) local variables. If we create an object ( Ex: String ) as a local variable that object is stored in the Heap memory area, but the object reference stays in the Stack memory area.

**Object Reference** — Variable names which point to objects in the Heap Memory. In the below example “my” is an object reference.

MyClass my = new MyClass();

**Method Block** — This is just a name for methods when talking about Java Memory Management. In the below example “myMethod” will be a method block. When after executing a method that memory location will be freed and may be located for another method block.

public void myMethod() {  
 int i = 0;  
}

**Metaspace**

Metaspace is a new memory space starting from the Java 8 version; it has replaced the older PermGen memory space. The most significant difference is how it handles memory allocation.

Specifically, this native memory region grows automatically by default.

We also have new flags to tune the memory:

* *MetaspaceSize* and *MaxMetaspaceSize*— we can set the Metaspace upper bounds.
* *MinMetaspaceFreeRatio —*is the minimum percentage of class metadata capacity free after garbage collection.
* *MaxMetaspaceFreeRatio*— is the maximum percentage of class metadata capacity free after a garbage collection to avoid a reduction in the amount of space.

Additionally, the garbage collection process also gains some benefits from this change. The garbage collector now automatically triggers the cleaning of the dead classes once the class metadata usage reaches its maximum metaspace size.

**Runtime Constant Pool** — This will be generated at the run time and stays inside the metaspace area. In here **static variables, constants**will be stored.

**Explanation with sample code**

public class Test {  
 int a, b;  
 static int *c* = 20;  
}  
  
public class MyClass{  
 Test t1 = new Test();  
  
 public int testMethod(int x){  
 int i = x + 3;  
 //some stuff  
 }  
}

* **t1 ( reference )** — Stack memory
* **Test object ( new Test() )** — Heap memory
* **int a, b** — Heap memory ( inside the object. Because these are bound to the object )
* **int c** — Metaspace ( static variables are not bound to an object )
* **testMethod**— Stack Memory ( this is also a reference — for a method )
* **int x** — Stack Memory ( this is a method parameter hence it is a local variable )
* **int i**— Stack Memory ( local variable. This is not bound to the object )