**Java Memory Model Notes**

Java memory is mainly divided into two types:

* **Heap** space memory
* **Stack** space memory

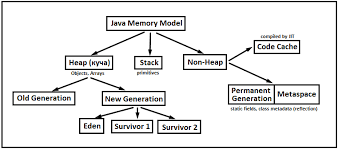


Fig: Block diagram of java memory model

**Heap**: Java Heap space is used by java runtime to allocate memory to Objects and JRE classes. Object are created in the Heap space.

Heap memory is further divided into smaller parts called generations:

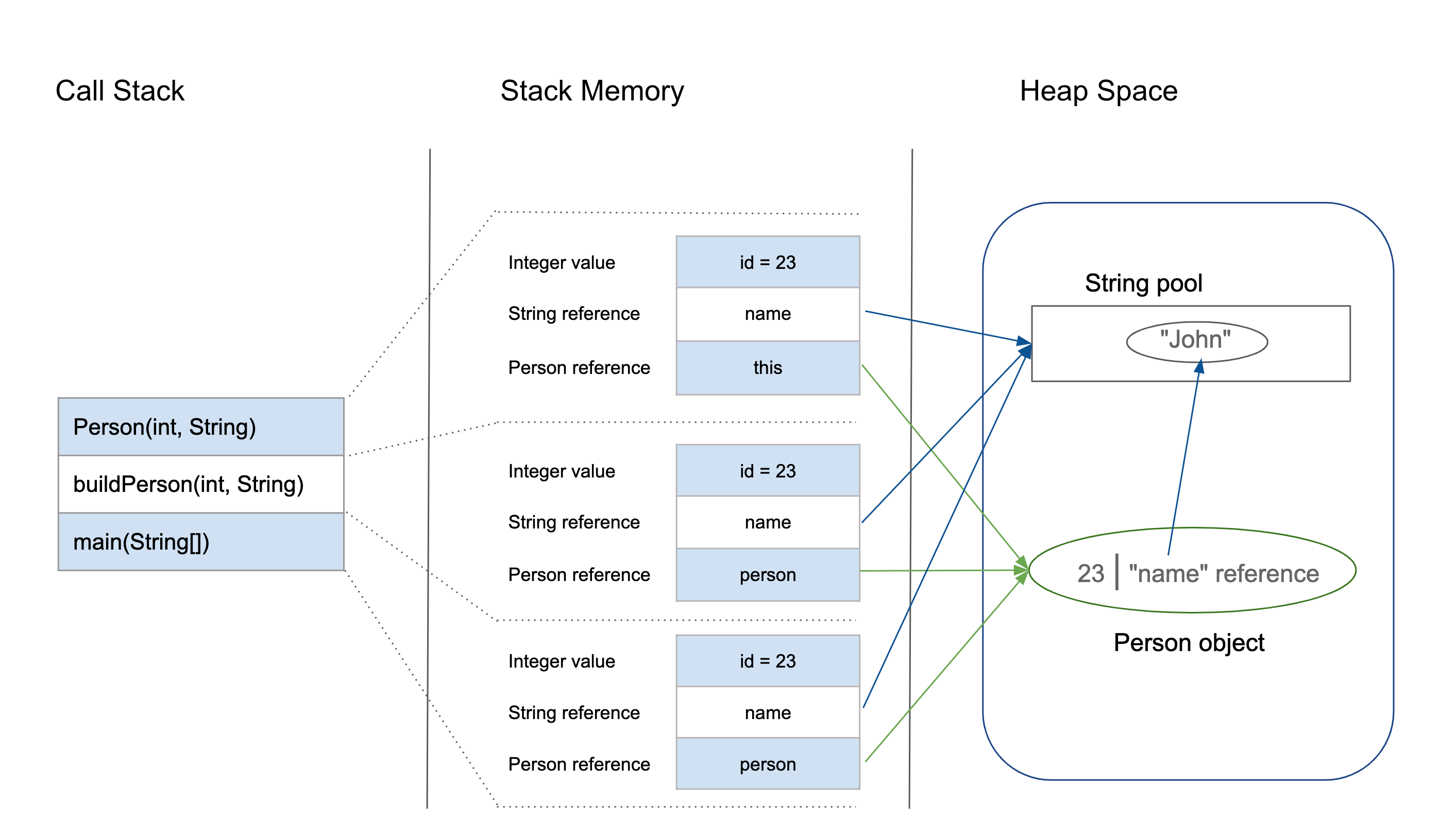
* **Young Generation:** The young generation is the place where all the new objects are created. When the young generation is filled, garbage collection is performed. This garbage collection is called **Minor GC**.

Young Generation is divided into three parts – **Eden Memory** and two **Survivor Memory** spaces.

* **Eden Space:** When we create an object, the memory will be allocated from the Eden Space.
* **Survivor Space:** This contains the objects that have survived from the Young garbage collection or Minor garbage collection. We have two equally divided survivor spaces called S0 and S1.
* **Old or Tenured Generation –** this is where long surviving objects are stored. When objects are stored in the Young Generation, a duration for the object is set and when that threshold is reached, the object is moved to the old generation.
* **Permanent Generation –** this consists of JVM metadata for the runtime classes and application methods

**GC:** **Garbage Collection** runs on the heap memory to free the memory used by objects that don’t have any reference. Any object created in the heap space has global access and can be referenced from anywhere of the application.

Note: Garbage collection in heap area is mandatory.



* Most of the newly created objects are located in the Eden memory space.
* When Eden space is filled with objects, Minor GC is performed and all the survivor objects are moved to one of the survivor spaces.
* Minor GC also checks the survivor objects and move them to the other survivor space. So at a time, one of the survivor space is always empty.
* Objects that are survived after many cycles of GC, are moved to the Old generation memory space. Usually, it is done by setting a duration of time for the young generation objects before they are ready to move to Old generation.

One of the basic ways of garbage collection involves three steps:

1. **Marking**: This is the first step where garbage collector identifies which objects are in use and which ones are not in use.
2. **Normal Deletion**: Garbage Collector removes the unused objects and reclaim the free space to be allocated to other objects.
3. **Deletion with Compacting**: For better performance, after deleting unused objects, all the survived objects can be moved to be together. This will increase the performance of allocation of memory to newer objects.

**Stack:** Java Stack memory is used for the execution of a thread. They contain method-specific values that are used as parameters and references to other objects in the heap that is getting referred from the method.

Stack memory is always referenced in LIFO (Last-In-First-Out) order. Whenever a method is invoked, a new block is created in the stack memory for the method to hold local primitive values and reference to other objects in the method.

As soon as the method ends, the block becomes unused and becomes available for the next method.  
Stack memory size is very less compared to Heap memory

**Permanent Generation:** Permanent Generation or “Perm Gen” contains the application metadata required by the JVM to describe the classes and methods used in the application. Note that Perm Gen is not part of Java Heap memory.

Perm Gen is populated by JVM at runtime based on the classes used by the application. Perm Gen also contains Java SE library classes and methods. Perm Gen objects are garbage collected in a full garbage collection.