How Java Stores Array

In Array, only a fixed set of elements can be stored. It is an index-based data structure, which starts from the 0th position. The first element will take place in index 0, and the 2nd element will take place in index 1, and so on.

Arrays are of two types:

- Single Dimensional arrays: Norml Arrays
- Multi Dimensional Arrays: Arrays storing arrays

Storage of Arrays

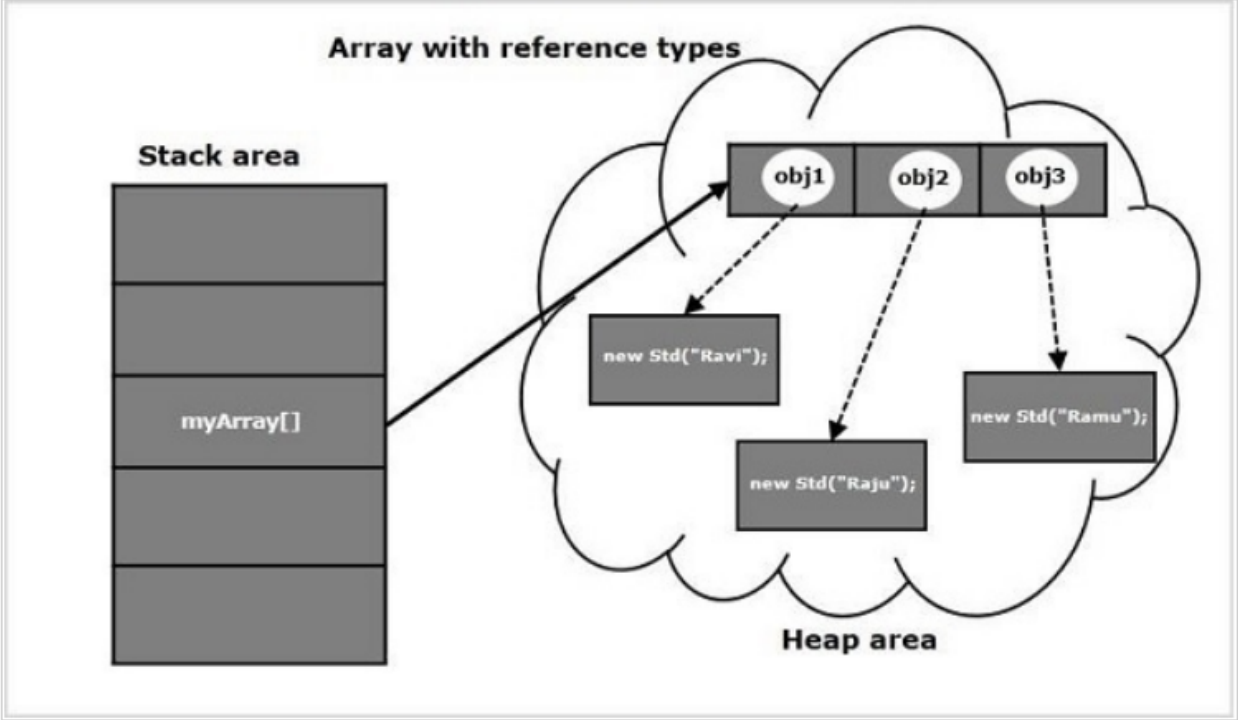
The reference types in java are stored in the heap area. Since arrays are reference types these are also stored in the heap area.

In addition to primitive datatypes arrays also store reference types: Another arrays (multi-dimensional), Objects. In this case the object array/multi-dimensional stores the references of the objects/arrays init, which points to the location of the objects/arrays.

For suppose, if we have a class with name Std with a constructor that accepts the name of a student and if we have defined an array of this class and populated it as shown below.

```
class Std {
   private String name;
   public Std(String name){
      this.name = name;
   }
}
public class Sample {
   public static void main(String args[]) throws Exception {
      //Creating an array to store objects of type Std
      Std myArray[] = new Std[3];
      //Populating the array
      myArray [0] = new Std("Ravi");
      myArray [1] = new Std("Raju");
      myArray [2] = new Std("Ramu");
   }
}
```

The memory of the array myArray might be like −



JVM memory locations

JVM has five memory locations namely −

- Heap − Runtime storage allocation for objects (reference types).
- Stack − Storage for local variables and partial results. A stack contains frames and allocates one for each thread. Once a thread gets completed, this frame also gets destroyed. It also plays roles in method invocation and returns.
- PC Registers − Program Counter Registers contains the address of an instruction that JVM is currently executing.
- Execution Engine − It has a virtual processor, interpreter to interpret bytecode instructions one by one and a JIT, just in time compiler.
- Native method stacks – It contains all the native methods used by the application.