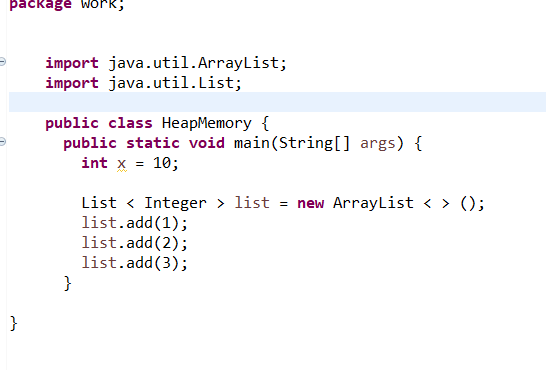
**JAVA MEMEORY MANAGEMENT**

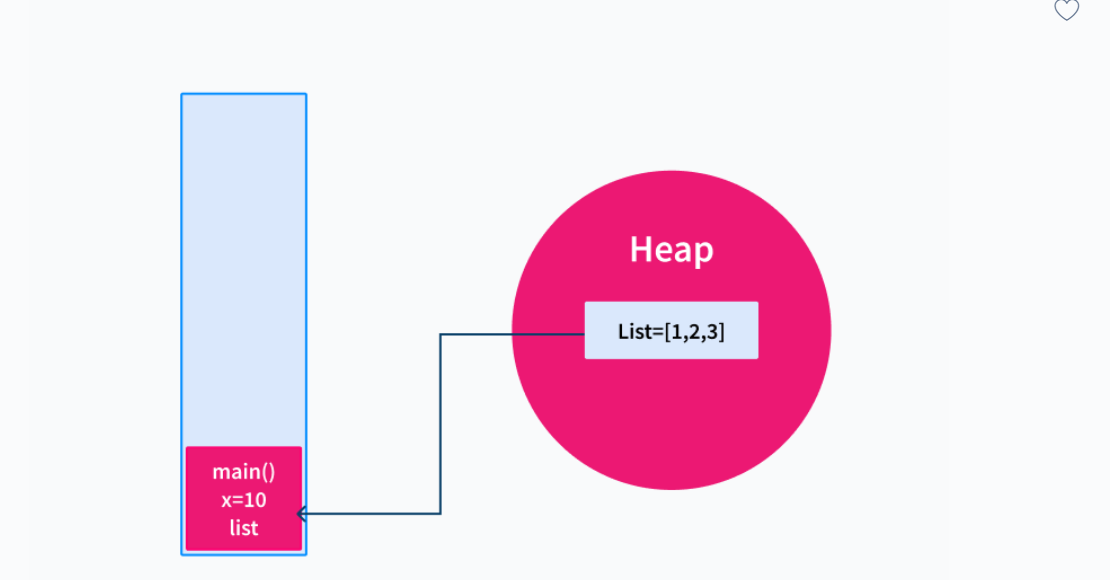
 It’s essential that the memory is managed thoroughly without any leaks. Allocation and deallocation of memory is a critical task and requires a lot of care and consideration. However in Java, unlike other programming language, the JVM and to be specific Garbage Collector has the role of managing memory allocation so that the programmer needs not to.

**Heap memory:**

Heap memory is used to store the objects that are created during the execution of a Java program. The reference to the objects that are created is stored in stack memory.  Heap follows dynamic memory allocation (memory is allocated during execution or runtime) and provides random access, unlike stack, which follows Last-In-First-Out (LIFO) order. The unused objects in the heap memory are cleared automatically by the Garbage Collector.



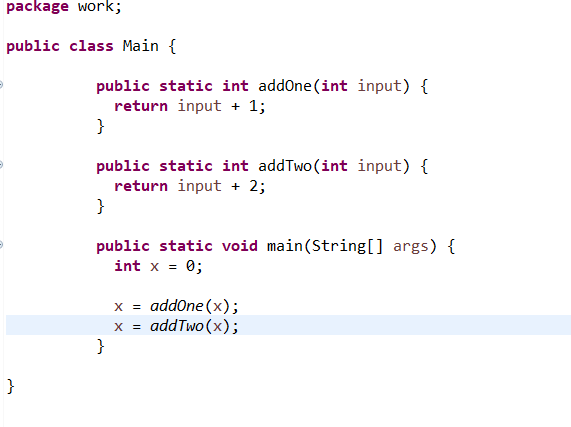
In the above example, the variable x is allocated in the stack, whereas the object list is allocated in the heap. Only the reference to the list object is stored in a stack.



**What is Stack Memory in Java?**

Stack memory is the space allocated for a process where all the function calls, primitive data types like int, double, etc., and local and reference variables of the functions are stored. Stack memory is always accessed in a Last-In-First-Out (LIFO) manner. In the stack memory, a new memory block is created for every method that is executed.

The values in the stack exist for as long as the function that created them is running. The size of the stack memory is fixed and cannot grow or shrink once created.



1. When the program is executed, the main method is executed first by the JVM. When the main method is called, a block is allocated for it in the stack. The variables in the method are created, assigned values, and then stored in the block.
2. When the addOne method is called from the main method, a new block is allocated in the stack. The variables for the method are created and stored in the block. Upon the completion of the execution of the method, the value is returned to the calling method(here it is the main method), and the block is cleared.
3. Similarly, when the addTwo method is called, a new block is allocated for it, and the variables are created and stored. When the method finishes execution, the value is returned to the calling method, and the block is cleared.
4. Finally, the main method completes its execution, and the block is cleared from the stack.

| **Property** | **Stack Memory** | **Heap Memory** |
| --- | --- | --- |
| Size | The size of stack memory is **smaller** | The size of heap memory is **larger** |
| Order | Stack memory is accessed in **Last-In-First-Out (LIFO)** manner | Heap memory is **dynamically allocated** and does not follow any order |
|  |  |  |
| Speed | Access to stack memory is **faster** because of Last-In-First-Out (LIFO) ordering | Access to memory is **slower** because it does not follow any order and is allocated dynamically |
| Resizing | Resizing of variables is **not allowed** in stack | Resizing of variables is **allowed** in a heap |
| Allocation | Memory is allocated and deallocated automatically when a method starts and completes its execution respectively | Memory is allocated when objects are created and deallocated by the garbage collector when they are no longer in use |
| Storage | Local variables and object references inside the function are stored in stack | The newly created objects and the JRE classes are stored in a heap |

**When to Use Stack and Heap Memory in Java**

We can decide when to use stack and heap memory in java based on the above-listed properties. Stack memory can be used

* When the variables are not used outside the method scope.
* When the size required for the variables is small.
* When accessing variables should be faster.

Heap memory can be used.

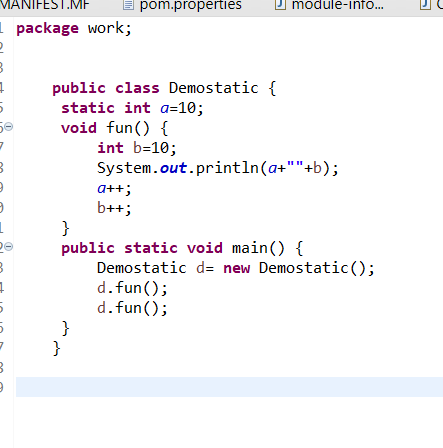
* When the variables are used outside the method scope.
* When the size required for the variables is large.

## Static Variable in Java

The static variables are those variables that are common to all the instances of the class; only a single copy of the static variable is created and shared among all the instances of the class. Because it is a class-level variable, memory allocation of such variables only happens once when the class is loaded in the memory.

## What is the Use of Static Keyword in Java

The main use of the static keyword in java is used to manage the memory management in java. Whenever we place a static keyword before the initialization of a particular class’s methods or variables, these static methods and variables belong to the class instead of their instances(Objects).



#### **What is Garbage Collection?**

Java garbage collection is an automatic process. Automatic garbage collection is the process of looking at heap memory, identifying which objects are in use and which are not, and deleting the unused objects. An in-use object, or a referenced object, means that some part of your program still maintains a pointer to that object. An unused or unreferenced object is no longer referenced by any part of your program. So the memory used by an unreferenced object can be reclaimed. The programmer does not need to mark objects to be deleted explicitly

#### **Advantages of Garbage Collection in Java**

The advantages of Garbage Collection in Java are:

* It makes java memory-efficient because the garbage collector removes the unreferenced objects from heap memory.
* It is automatically done by the garbage collector(a part of JVM), so we don’t need extra effort.

**Collection:** Collection is a [interface](https://www.geeksforgeeks.org/interfaces-in-java/) present in java.util.package. It is used to represent a group of individual objects as a single unit.

**Collections** is a utility class present in java.util.package. It defines several utility methods like sorting and searching which is used to operate on collection. It has all static methods. These methods provide much-needed convenience to developers, allowing them to effectively work with [Collection Framework](https://www.geeksforgeeks.org/collections-in-java-2/).