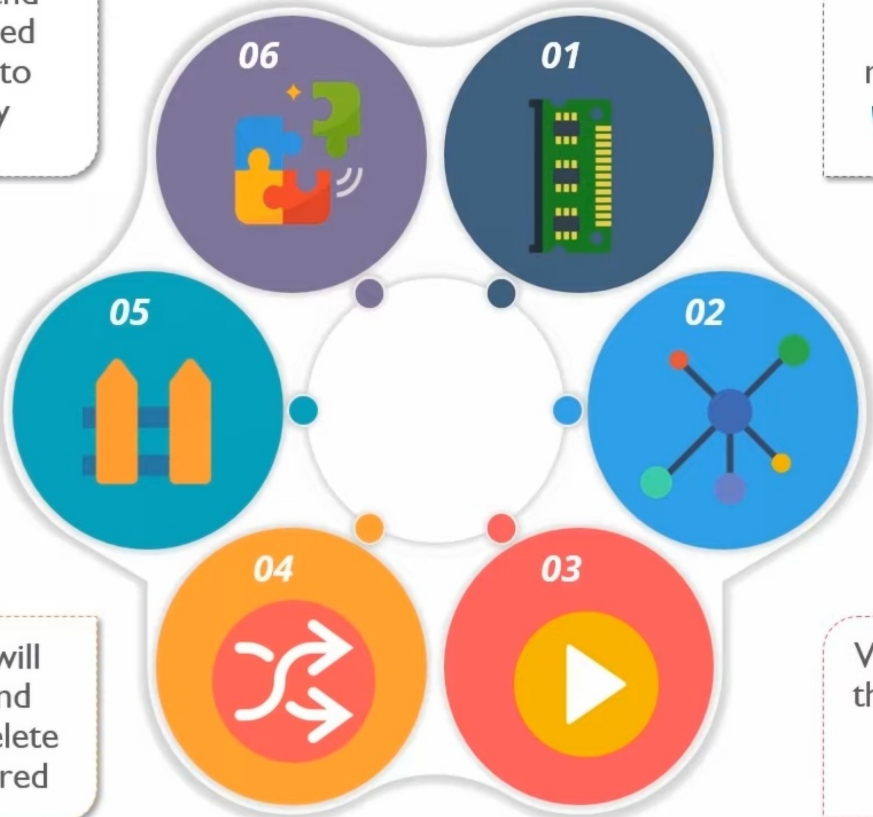
# Memory Allocation – Stack



01 Java stack is part of the memory where *temporary variables*(local), are stored

02 It is used to execute a thread and may have certain short-lived values as well as references to other objects

03 Variables that are stored on the stack exist for as long as the function that created them are running

04 The size of the stack will vary since methods and functions create and delete local variables as required

05 Stack has size limits, which can vary according to the operating system you use

06 Memory is allocated and then subsequently freed without you needing to manage the memory allocation

# Memory Allocation – Heap

Heap is the area in the memory which is used to store objects

Memory is not managed automatically, nor is it as tightly managed like stack

You would need to free allocated memory yourself when these blocks are no longer needed

There is no size limit in the heap

Compared to stack, objects in the heap are much slower to access

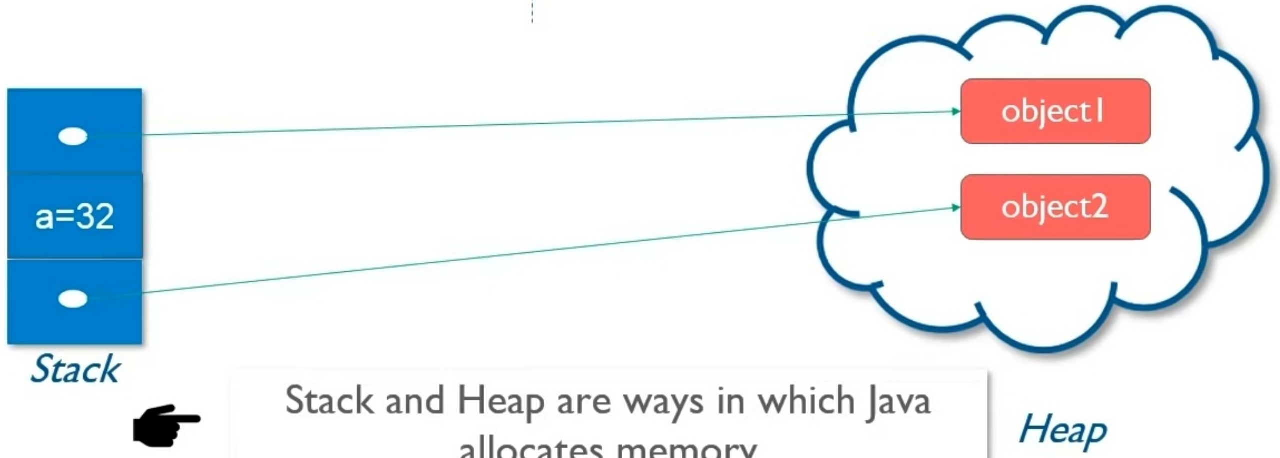
# Stack Vs Heap

## *Stack*

- Stack is used for static memory allocation
- Variables that are allocated on the stack are accessible directly from memory, thus these can run very fast
- Memory allocation happens when the program is compiled

## *Heap*

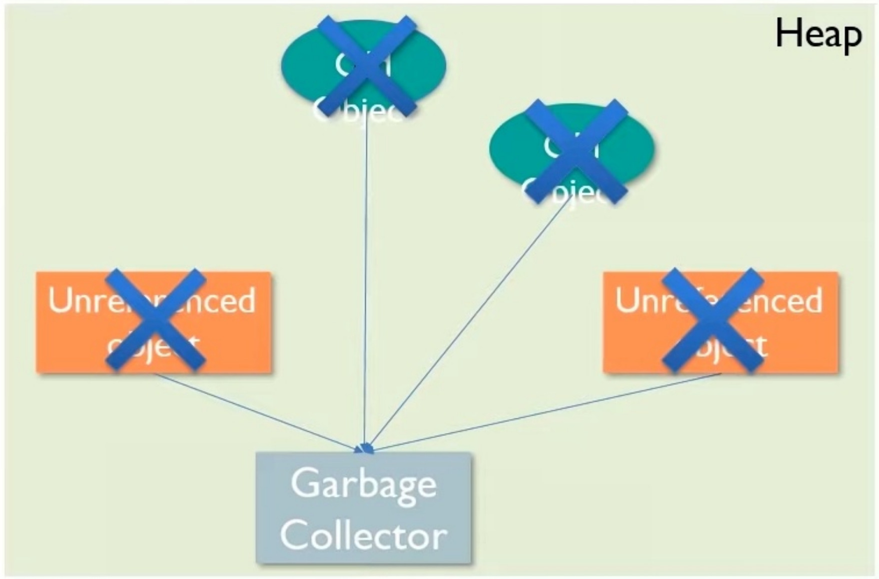
- Heap is used for dynamic memory allocation
- Accessing objects on the heap takes more time
- Memory allocation begins at runtime



Stack and Heap are ways in which Java allocates memory

# Garbage Collection

- *Garbage* means *unreferenced objects*
- *Garbage Collection* is a process of destroying the unreferenced objects
- In Java, Garbage Collection is performed automatically, thus providing better memory management
- If Heap memory is full, Garbage Collection starts from the older objects



## Ways in which Objects are Unreferenced

- Nulling the reference:

```
Student s=new Student();
s=null;
```

- Assigning a reference to another :

```
Student s1=new Student();
Student s2=new Student();
s1=s2; // now the first object will be available for Garbage Collection
```

- Anonymous object :

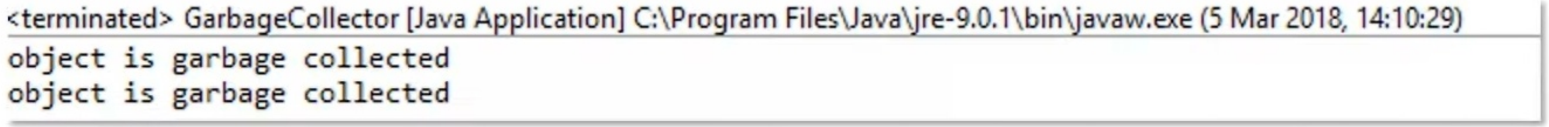
```
new Student();
```

# finalize() and gc() method

- The *finalize()* method is invoked before an unreferenced object is removed
- The *gc()* method is used to invoke the garbage collector to perform clean-up processing
- The *gc()* is found in *System* and *Runtime* classes

```
public class TestGarbage1{
 public void finalize(){System.out.println("object is garbage collected");}
 public static void main(String args[]){
  TestGarbage1 s1=new TestGarbage1();
  TestGarbage1 s2=new TestGarbage1();
  s1=null;
  s2=null;
  System.gc();
 }
}
```

Output:



```
<terminated> GarbageCollector [Java Application] C:\Program Files\Java\jre-9.0.1\bin\javaw.exe (5 Mar 2018, 14:10:29)
object is garbage collected
object is garbage collected
```