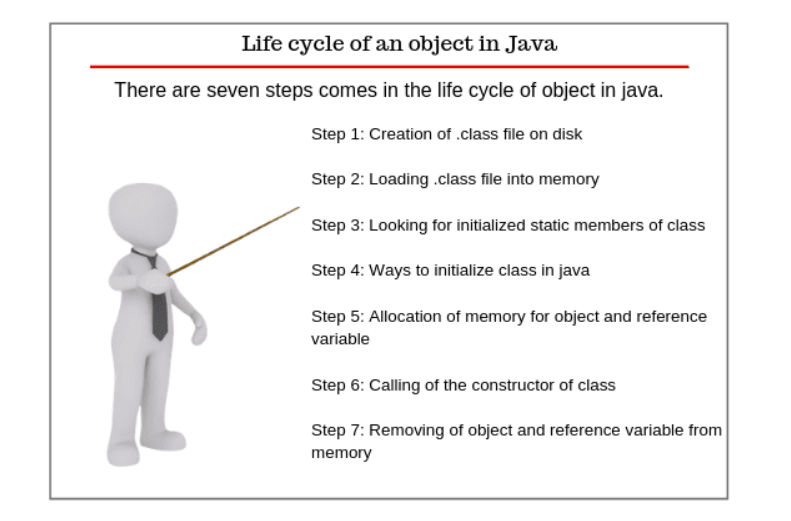
**Life Cycle of Object in Java**



**Step 1:** Creation of .class file on disk

Java programs run on the Java virtual machine (JVM).

When we compile the Java class, it is transformed into byte code which

is platform and machine-independent.

The compiled classes are stored as a .class file on a disk.

**Step 2:** Loading .class file into memory

After that, the Java runtime finds out that class on the disk that is in

the form of a .class file. Java classloader loads that class into memory

and then Java runtime reads it into the memory.

**Step 3:** Looking for initialized static members of class

Now Java looks for all initialized static members of the class such as

static method, static field, and static block.

You always remember that all the static members of the class do not

belong to any particular instance of the class. It belongs to the class itself

and is shared by all the objects created from the class.

**Step 4:** Ways to initialize class in java

A class can be initialized in two ways in Java. First, when you access a

static field or static method of the class.

For example, when you run the main method in a class,

the class is initialized because the main method is static and the

second way is when object or instance of the class is created using

the new keyword, the class is initialized.

**Step 5:** Allocation of memory for object and reference variable

In stage 5, Java allocates the memory on the heap for the object and memory on the stack for object reference variable.

**Step 6:** Calling of the constructor of class

After allocating the memory, JVM calls the constructor of the class

which is like a method but it is called only once when the object is created.

Thus, the object lives its life and providing access to its fields and

methods whatever we want and need to access them.

**Step 7:** Removing of object and reference variable from memory

When the accessing of field and methods are completed, the object and

its reference are removed from the memory by the JVM.

At this time the object has died.

Actually, when the object is no longer in use,

Java runtime calls the garbage collector to destroy all the objects.

Thus, objects are born, live, and die.