**Can Java Abstract Class Have a Constructor?**

Yes, an abstract class in Java can have a constructor. While you cannot instantiate an abstract class directly, its constructor can be invoked when a concrete subclass is created, allowing for the proper initialization of fields in the abstract class.

# Difference Between Stack Memory and Heap Memory

* Stack Memory: This is used for static memory allocation and the execution of threads. It stores primitive values and references to heap objects. Each thread has its own stack that is created when the thread is started. The stack is LIFO (Last In First Out) structured.
* Heap Memory: This is used for dynamic memory allocation for Java objects and JRE classes. Objects are created on the heap, and heap memory is shared among all threads. Garbage Collection operates primarily on the heap to free up memory used by objects that are no longer reachable.

# Final and Abstract Methods and Classes

* Final Class: A final class cannot be subclassed. This is done to prevent any inheritance from the class.
* Abstract Class: An abstract class cannot be instantiated on its own and must be subclassed to be used. Abstract classes are designed to be base classes.
* Final Method: A final method cannot be overridden by subclasses. This is used to prevent a specific method from being changed.
* Abstract Method: An abstract method has no implementation in the class that declares it and must be implemented by subclasses.

# How Garbage Collector Works in Java

Java’s Garbage Collector (GC) works by identifying and disposing of objects that are no longer needed by a program. It primarily uses the “mark-and-sweep” algorithm, where it first “marks” objects that are reachable through a set of active references, and then “sweeps” away objects that are not marked (i.e., unreachable), freeing up memory.

# Shallow Copy and Deep Copy

* Shallow Copy: A shallow copy of an object copies the values of the object’s properties as they are, including references to other objects. If the object contains references to other objects, the references are copied but the actual objects are not, meaning multiple references to the same object can exist.
* Deep Copy: A deep copy of an object will duplicate everything. This means creating new instances for each element that the object references, thus no two references point to the same object.