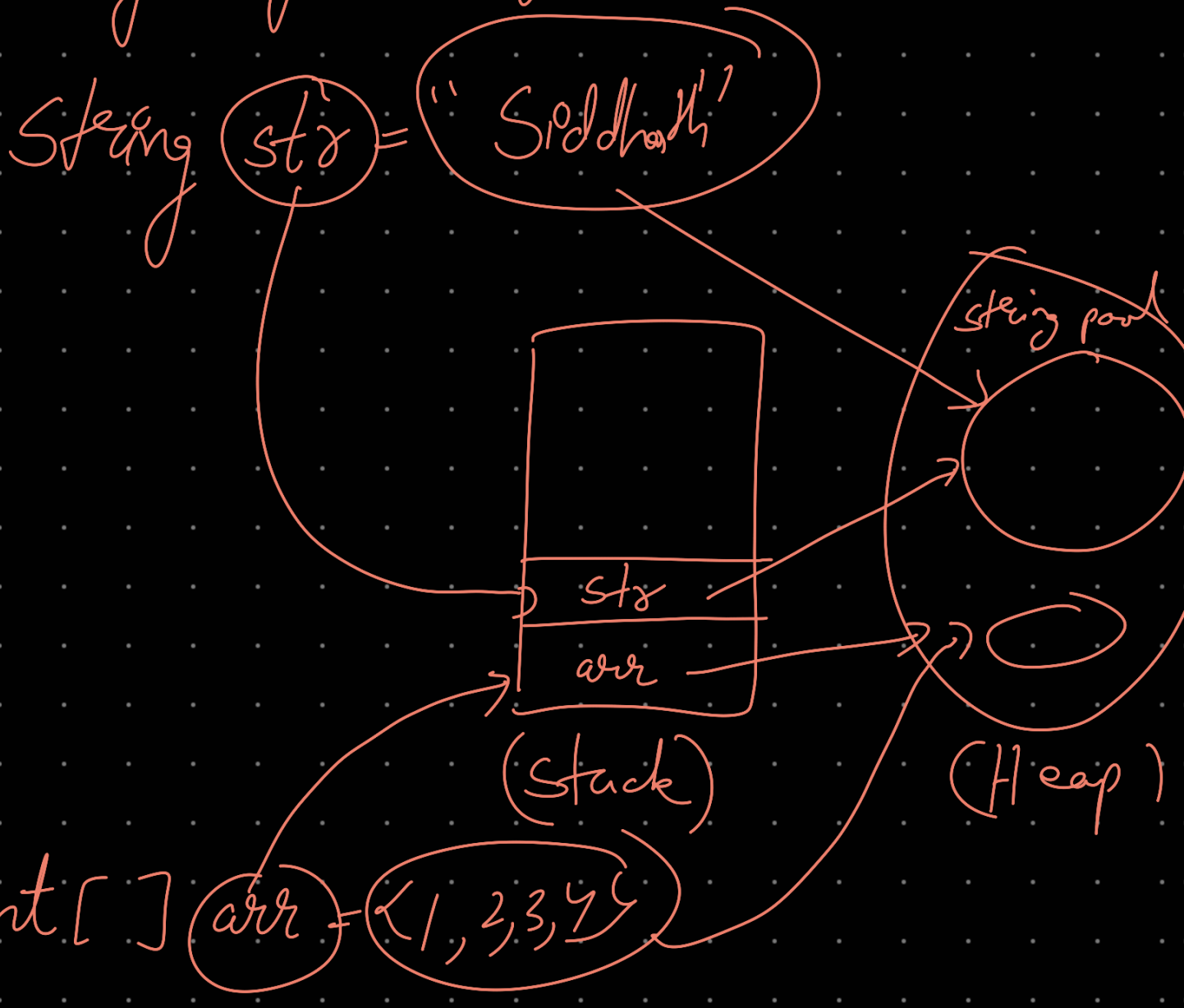
[Java Memory Management]

* int, short, byte, char, bool
  - stored in Stack

* String, obj, Array



Student st = new Student()

st → Kind of pointer present in stack

new Student() → in heap

We haven't explicitly deleted all these references, GC will delete

* GC is code running in different thread of JVM

JVM process → Some other threads

Main thread → Our code runs here → It can create User threads

GC threads

Both threads are Synchronised

so Both are sharing Heap memory

[Types of References]

① Person p = new Person();

p → Strong reference

Not garbage collected

Lead to Memory leaks

② Person p = new Person()

WeakReference<Person> weak = new WeakReference<>(p)

Weak reference

Used for objects that needs to be garbage collected when no strong reference exists

③ `Person p = new Person()`

`SoftReference<Person> soft = new SoftReference(p)`

soft reference

* When memory is sufficient, can be accessed using soft reference

Heap is divided

- Young Generation → Minor GC
- Old Gen or Tenured → Major GC
- permanent Gen

[Garbage collection Algorithm]

1. Mark and Sweep
2. Mark and Sweep with Compaction

Version of GC

- Serial GC
- parallel GC → (Default in Java 8)
- Concurrent Mark and Sweep (CMS)

↳ Gc | Gc C → Java > 8 version

* When Gc runs it locks Heap Memory, so all other User threads is paused for that time because of Synchronization

↳ It's an overhead, that is why C++ is still faster

↳ known as "Stop the world"

↳ Gc is expensive

↳ Specially serial Gc

⇓

Enhanced by parallel Gc