In a hypothetical theater:

* It is called **Mutex** if only one person is allowed to watch the play.
* It is called **Semaphore** if N number of people are allowed to watch the play. If anybody leaves the Theater during the play then other person can be allowed to watch the play.
* It is called **CountDownLatch** if no one is allowed to enter until every person vacates the theater. Here each person has free will to leave the theater.
* It is called **CyclicBarrier** if the play will not start until every person enters the theater. Here a showman can not start the show until all the persons enter and grab the seat. Once the play is finished the same barrier will be applied for next show.

Here, a person is a *thread*, a play is a *resource*.

**CountDownLatch:** If we want all of our threads to do

something + countdown

so that **other waiting** (for count to reach zero) **threads** can proceed, we can use countdown latch. All prior threads who actually did the countdown can go on in this situation but there is no guarantee that **line processed after latch.countdown() will be after waiting for other threads to reach at latch.countdown()** but it has a guarantee that **other waiting threads** will only start further after latch.await() has reached zero.

**CyclicBarrier:** If we want all our thread to

do something + await at common point + do something

(each await call will decrease wait time for threads to carry on further)

CyclicBarrier functionality can be achieved by CountDownLatch only once by calling latch.countdown() followed by latch.await() by all the threads.

but again you cant reset/reuse the countdownlatch.

Best example where I used CyclicBarrier is to initialize multiple caches (warmed by multiple threads) and then starting further processing, and I wanted to reinitialize other caches again in Sync.

1. Semaphores – Restrict the number of threads that can access a resource. Example, limit max 10 connections to access a file simultaneously.
2. Mutex – Only one thread to access a resource at once. Example, when a client is accessing a file, no one else should have access the same file at the same time.