<https://stackoverflow.com/questions/7335950/semaphore-vs-monitors-whats-the-difference/7336799#7336799>

Q: What are the major differences between a Monitor and a Semaphore?

A1:

A **Monitor** is an object designed to be accessed from multiple threads. The member functions or methods of a monitor object will enforce mutual exclusion, so only one thread may be performing any action on the object at a given time. If one thread is currently executing a member function of the object then any other thread that tries to call a member function of that object will have to wait until the first has finished.

A **Semaphore** is a lower-level object. You might well use a semaphore to implement a monitor. A semaphore essentially is just a counter. When the counter is positive, if a thread tries to acquire the semaphore then it is allowed, and the counter is decremented. When a thread is done then it releases the semaphore, and increments the counter.

If the counter is already zero when a thread tries to acquire the semaphore then it has to wait until another thread releases the semaphore. If multiple threads are waiting when a thread releases a semaphore then one of them gets it. The thread that releases a semaphore need not be the same thread that acquired it.

A monitor is like a public toilet. Only one person can enter at a time. They lock the door to prevent anyone else coming in, do their stuff, and then unlock it when they leave.

A semaphore is like a bike hire place. They have a certain number of bikes. If you try and hire a bike and they have one free then you can take it, otherwise you must wait. When someone returns their bike then someone else can take it. If you have a bike then you can give it to someone else to return --- the bike hire place doesn't care who returns it, as long as they get their bike back.