Learning Objectives in this Part of the Lesson

- Understand the concept of semaphores
Learning Objectives in this Part of the Lesson

- Understand the concept of semaphores
- Be aware of the two types of semaphores
Learning Objectives in this Part of the Lesson

• Understand the concept of semaphores
• Be aware of the two types of semaphores
• Note a human-known use of semaphores
Introduction to Semaphores
Introduction to Semaphores

• A semaphore is conceptually an “object” that can be atomically incremented & decremented to control access to a shared resource

See en.wikipedia.org/wiki/Semaphore_(programming)
Introduction to Semaphores

- A semaphore is conceptually an “object” that can be atomically incremented & decremented to control access to a shared resource
- e.g., originally used to control access to a shared railroad track

See en.wikipedia.org/wiki/Railway_semaphore_signal
Concurrent programs use semaphores to coordinate interactions between multiple threads.

See tutorials.jenkov.com/java-concurrency/semaphores.html
Concurrent programs use semaphores to coordinate interactions between multiple threads, e.g.,

- A semaphore can control the access of threads to a limited # of resources

See www.youtube.com/watch?v=RAv71VbdkBc for the Semaphore anthem ;-}
Introduction to Semaphores

• Concurrent programs use semaphores to coordinate interactions between multiple threads, e.g.,
  • A semaphore can control the access of threads to a limited # of resources
  • It records a count ("permits") of how many units of a resource are available
Introduction to Semaphores

• Concurrent programs use semaphores to coordinate interactions between multiple threads, e.g.,
  • A semaphore can control the access of threads to a limited # of resources
  • It records a count (“permits”) of how many units of a resource are available
  • It provides operations to adjust the permit count atomically as units are acquired or released
Introduction to Semaphores

- Concurrent programs use semaphores to coordinate interactions between multiple threads, e.g.,
  - A semaphore can control the access of threads to a limited # of resources
  - It records a count ("permits") of how many units of a resource are available
  - It provides operations to adjust the permit count atomically as units are acquired or released
Concurrent programs use semaphores to coordinate interactions between multiple threads, e.g.,

- A semaphore can control the access of threads to a limited # of resources
- It records a count (“permits”) of how many units of a resource are available
- It provides operations to adjust the permit count atomically as units are acquired or released
Introduction to Semaphores

- Concurrent programs use semaphores to coordinate interactions between multiple threads, e.g.,
  - A semaphore can control the access of threads to a limited # of resources
  - It records a count (“permits”) of how many units of a resource are available
  - It provides operations to adjust the permit count atomically as units are acquired or released
  - Threads can wait (timed or blocking) until a unit of the resource is available
Concurrent programs use semaphores to coordinate interactions between multiple threads, e.g.,

- A semaphore can control the access of threads to a limited number of resources.
- It records a count ("permits") of how many units of a resource are available.
- It provides operations to adjust the permit count atomically as units are acquired or released.
- Threads can wait (timed or blocking) until a unit of the resource is available.
- When a thread is done with a resource, the permit count is incremented atomically, and another waiting thread can acquire it.
Introduction to Semaphores

Concurrent programs use semaphores to coordinate interactions between multiple threads, e.g.,

- A semaphore can control the access of threads to a limited # of resources
- It records a count ("permits") of how many units of a resource are available
- It provides operations to adjust the permit count atomically as units are acquired or released
- Threads can wait (timed or blocking) until a unit of the resource is available
- When a thread is done with a resource the permit count is incremented atomically & another waiting thread can acquire it

This example “fully brackets” the acquiring & releasing of permits, i.e., the thread that acquires a semaphore is the same as the one that releases it.
Introduction to Semaphores

• There are two types of semaphores
Introduction to Semaphores

- There are two types of semaphores
  - **Counting semaphores**

See javarevisited.blogspot.com/2012/05/counting-semaphore-example-in-java-5.html
Introduction to Semaphores

• There are two types of semaphores
  • **Counting semaphores**
    • Have # of permits defined by a counter (N) with precise meaning
Introduction to Semaphores

• There are two types of semaphores
  • **Counting semaphores**
    • Have # of permits defined by a counter (N) with precise meaning
      • **Negative**
        • exactly -N threads queued waiting to acquire semaphore
There are two types of semaphores

- **Counting semaphores**
  - Have # of permits defined by a counter (N) with precise meaning
    - **Negative**
    - **Zero** == no waiting threads
      - an acquire operation will block the invoking thread until the counter N is positive
There are two types of semaphores

**Counting semaphores**
- Have \# of permits defined by a counter (N) with precise meaning
  - **Negative**
  - **Zero** == no waiting threads
  - **Positive** == no waiting threads
    - an acquire operation will not block the invoking thread

A valid semaphore implementation needn’t be precisely aligned with these states
Introduction to Semaphores

- There are two types of semaphores
  - Counting semaphores
  - Binary semaphores

Introduction to Semaphores

• There are two types of semaphores
  • Counting semaphores
  • **Binary semaphores**
    • Have only 2 states: acquired (0) & not acquired (1)
There are two types of semaphores

- **Counting semaphores**
- **Binary semaphores**
  - Have only 2 states: acquired (0) & not acquired (1)
  - Restrict the counter \( N \) to the values 0 & 1

In practice, binary semaphores are often implemented via counting semaphores
Introduction to Semaphores

• We’ll analyze examples of counting & binary semaphores later
Introduction to Semaphores

- We’ll analyze examples of counting & binary semaphores later, e.g.
- The PalantiriSimulator app uses a counting semaphore

See [github.com/douglasraigschmidt/LiveLessons/tree/master/PalantiriManagerApplication](https://github.com/douglasraigschmidt/LiveLessons/tree/master/PalantiriManagerApplication)
Introduction to Semaphores

- We’ll analyze examples of counting & binary semaphores later, e.g.
  - The PalantiriSimulator app use a counting semaphore
  - This use case demonstrates fully-bracketed semaphores
Introduction to Semaphores

- We’ll analyze examples of counting & binary semaphores later, e.g.
  - The PalantiriSimulator app uses a counting semaphore
  - The Ping/Ping app uses a pair of binary semaphores

See [github.com/douglaascarigschmidt/LiveLessons/tree/master/PingPongApplication](https://github.com/douglaascarigschmidt/LiveLessons/tree/master/PingPongApplication)
• We’ll analyze examples of counting & binary semaphores later, e.g.
  • The PalantiriSimulator app uses a counting semaphore
  • The Ping/Pong app uses a pair of binary semaphores
  • This use case demonstrates non-bracketed semaphores
Human Known Use of Semaphores
Human Known Uses of Semaphores

- A human known use of counting semaphores applies them to schedule access to beach volleyball courts

See [en.wikipedia.org/wiki/Corona_del_Mar_State_Beach](en.wikipedia.org/wiki/Corona_del_Mar_State_Beach)
A human known use of counting semaphores applies them to schedule access to beach volleyball courts.

A bag full of balls is used to limit the number of teams that can concurrently play volleyball.
End of Introduction to Java Semaphores