Introduction to Java Monitor Objects

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

- Understand what monitors are & know how Java built-in monitor objects can ensure mutual exclusion & coordination between threads

1. Enter monitor object
2. Acquire lock
3. wait()
4. notifyAll()
5. Release lock
6. Leave monitor object

Learning Objectives in this Part of the Lesson

- Understand what monitors are & know how Java built-in monitor objects can ensure mutual exclusion & coordination between threads
- Note a human known use of monitors
Overview of Monitors
Overview of Monitors

• A monitor is a synchronization mechanism designed in the early 1970s

See en.wikipedia.org/wiki/Monitor_(synchronization)
Overview of Monitors

- A monitor provides three capabilities to concurrent programs:

  1. **Critical Section**

Overview of Monitors

- A monitor provides three capabilities to concurrent programs

  1. Only one thread at a time has mutually exclusive access to a critical section

See [en.wikipedia.org/wiki/Critical_section](en.wikipedia.org/wiki/Critical_section)
A monitor provides three capabilities to concurrent programs:

1. Only one thread at a time has mutually exclusive access to a critical section.
2. Threads running in a monitor can block awaiting certain conditions to become true.
A monitor provides three capabilities to concurrent programs:

1. Only one thread at a time has mutually exclusive access to a critical section.
2. Threads running in a monitor can block awaiting certain conditions to become true.
3. A thread can notify one or more threads that conditions they’re waiting on have been met.
Overview of Built-in Java Monitor Objects
Overview of Java Built-in Monitor Objects

- All objects in Java can be used as built-in monitor objects, which support two types of thread synchronization

See [en.wikipedia.org/wiki/Monitor_(synchronization)#Implicit_condition_variable_monitors](en.wikipedia.org/wiki/Monitor_(synchronization)#Implicit_condition_variable_monitors)
Overview of Java Built-in Monitor Objects

• All objects in Java can be used as built-in monitor objects, which support two types of thread synchronization

• **Mutual exclusion** – allows concurrent access & updates to shared data without race conditions

![Diagram showing mutual exclusion with synchronized methods m1() and m2() for Thread1 and Thread2.](image-url)
Overview of Java Built-in Monitor Objects

- All objects in Java can be used as built-in monitor objects, which support two types of thread synchronization
- **Mutual exclusion** – allows concurrent access & updates to shared data without race conditions

Every Java object has a single "intrinsic lock" associated with it

Java’s execution environment supports mutual exclusion via an entrance queue & synchronized methods/statements
Overview of Java Built-in Monitor Objects

- All objects in Java can be used as built-in monitor objects, which support two types of thread synchronization
  - **Mutual exclusion** – allows concurrent access & updates to shared data without race conditions
  - **Coordination** – Ensures computations run properly, e.g., in the right order, at the right time, under the right conditions, etc.
Overview of Java Built-in Monitor Objects

• All objects in Java can be used as built-in monitor objects, which support two types of thread synchronization
  
  - **Mutual exclusion** – allows concurrent access & updates to shared data without race conditions
  
  - **Coordination** – Ensures computations run properly, e.g., in the right order, at the right time, under the right conditions, etc.

A Java Monitor Object

- synchronized m1()
- synchronized m2()

Wait Queue
- wait()
- notify()
- notifyAll()

Every Java object has one “intrinsic condition” associated with it

Java’s execution environment supports coordination via a wait queue & notification mechanisms
Overview of Java Built-in Monitor Objects

- These mechanisms implement a variant of the *Monitor Object* pattern.

See [www.dre.vanderbilt.edu/~schmidt/PDF/monitor.pdf](www.dre.vanderbilt.edu/~schmidt/PDF/monitor.pdf)
Overview of Java Built-in Monitor Objects

• These mechanisms implement a variant of the *Monitor Object* pattern

• **Intent** – Ensure that only one method runs within an object & allow an object’s methods to cooperatively schedule their execution sequences
Human Known Use of Monitors
Human Know Use of Monitors

- A human known use of a monitor is an operating room in a hospital.
End of Introduction to Java Monitor Objects