Java Monitor Objects: Introduction
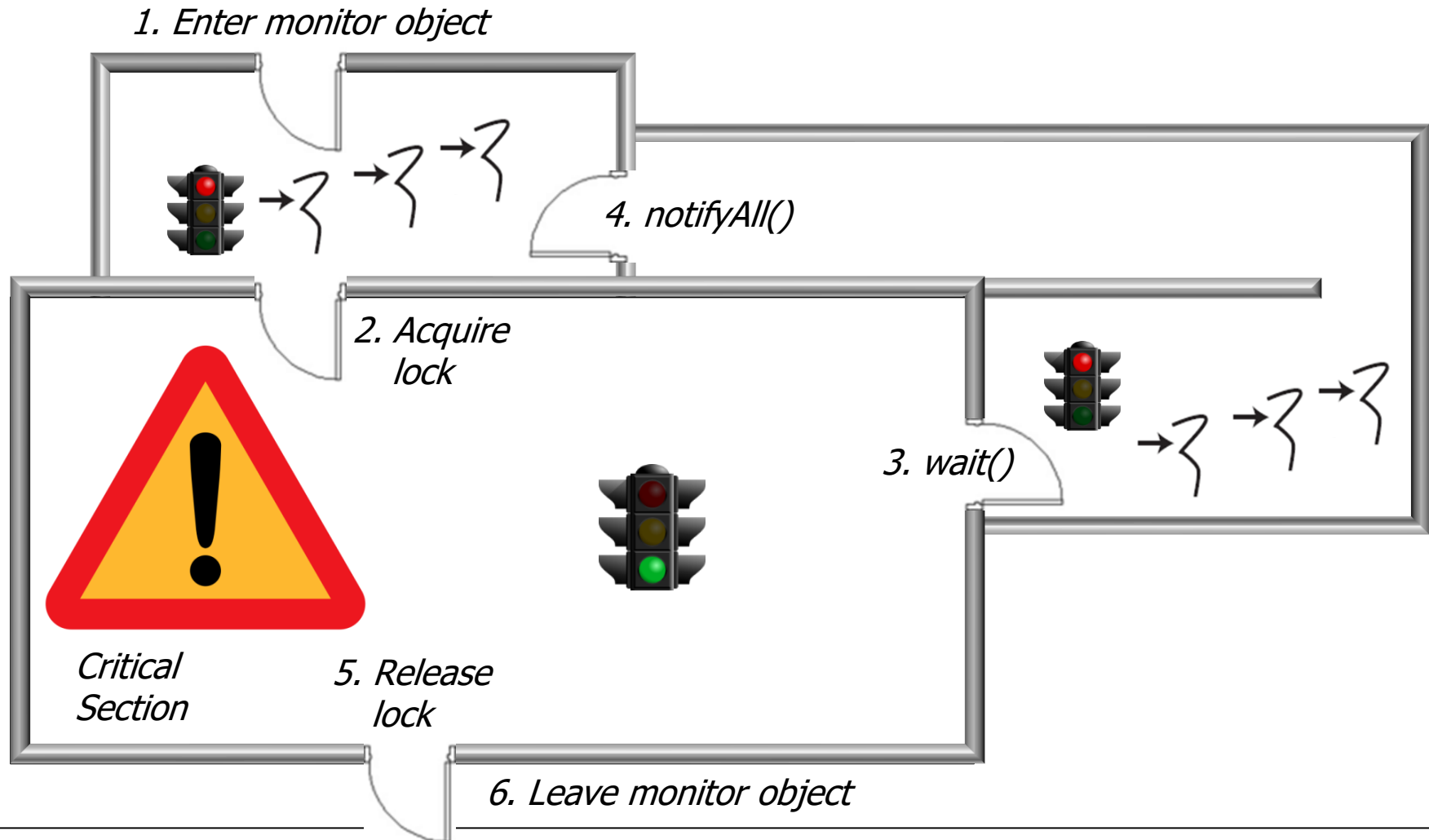
Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

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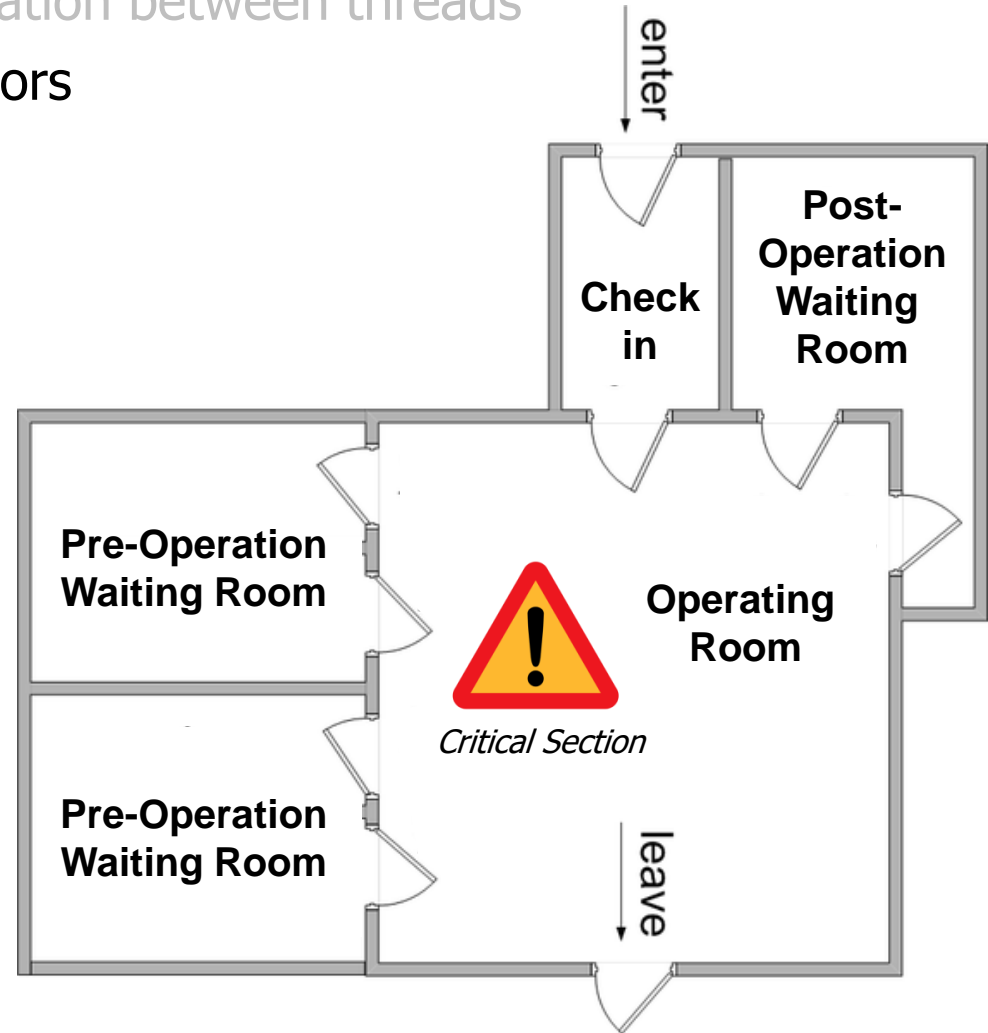
Learning Objectives in this Part of the Lesson

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

1. Enter monitor object

4. notifyAll()

2. Acquire lock

3. wait()

Critical Section

5. Release lock

6. Leave monitor object

See www.artima.com/insidejvm/ed2/threadsynch.html

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

enter

Post-Operation Waiting Room

Check in

Pre-Operation Waiting Room

Operating Room

Critical Section

Pre-Operation Waiting Room

leave

Overview of Monitors

Overview of Monitors

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

Billboard 1973
THE INTERNATIONAL NEWSWEEKLY OF MUSIC AND HOME ENTERTAINMENT

Top Rock'n'Roll Hits

1	Crocodile Rock *Elton John*	6	Will It Go Round In Circles *Billy Preston*
2	Let's Get It On *Marvin Gaye*	7	Frankenstein *The Edgar Winter Group*
3	Midnight Train To Georgia *Gladys Knight & The Pips*	8	Love Train *The O'Jays*
4	Bad, Bad Leroy Brown *Jim Croce*	9	Goodbye Yellow Brick Road *Elton John*
5	Brother Louie *Stories*	10	Ramblin Man *The Allman Brothers Band*

1973

SHARED CLASSES

PER BRINCH HANSEN

(1973)

The author discusses the close relationship between data and operations and suggests that a compiler should be able to check that data structures are accessed by meaningful procedures only. This idea leads to the introduction of shared classes—a programming notation for the monitor concept. The notation is illustrated by a message buffer for concurrent processes.

We will discuss the close relationship between data and operations and use it to define a very important form of resource protection.

If we consider variables of *primitive types* such as *integer* and *boolean*, it is quite possible that values of different types will be represented by identical bit strings at the machine level. For example both the *boolean* value *true* and the *integer* value 1 might be represented by the bit string

000...001

in single machine words.

So data of different types are distinguished not only by the representation of their values, but also by the operations associated with the types. An *integer*, for example, is a datum subject only to arithmetic operations, comparisons, and assignments involving other data subject to the same restrictions.

Now consider *structured types*. Take for example a variable that represents a message buffer which contains a sequences of messages sent, but not yet received. A *static* picture of process communication can be defined by

P. Brinch Hansen, *Operating System Principles*, Section 7.2 Class Concept, Prentice Hall, Englewood Cliffs, NJ, (July 1973), 226–232. Copyright © 2001 Per Brinch Hansen.

1

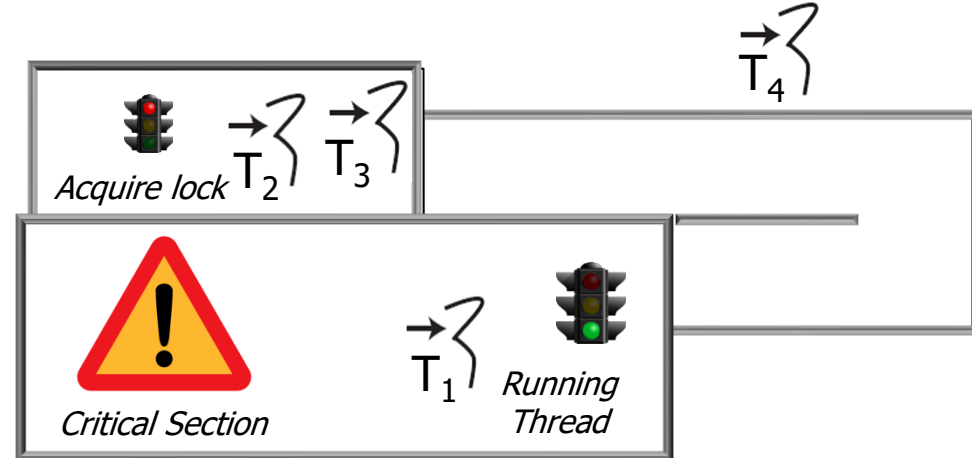See en.wikipedia.org/wiki/Monitor_(synchronization)

Overview of Monitors
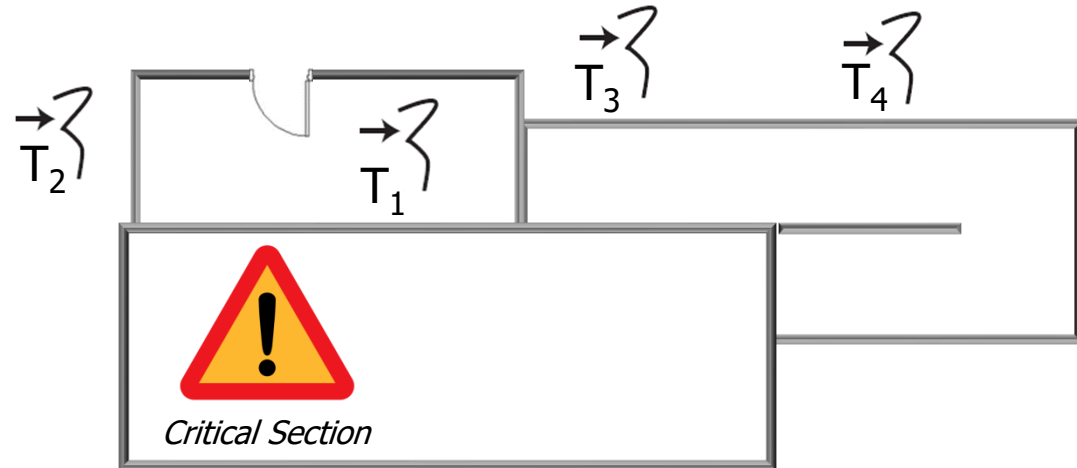
- A monitor provides three capabilities to concurrent programs

Critical Section

Overview of Monitors

- A monitor provides three capabilities to concurrent programs

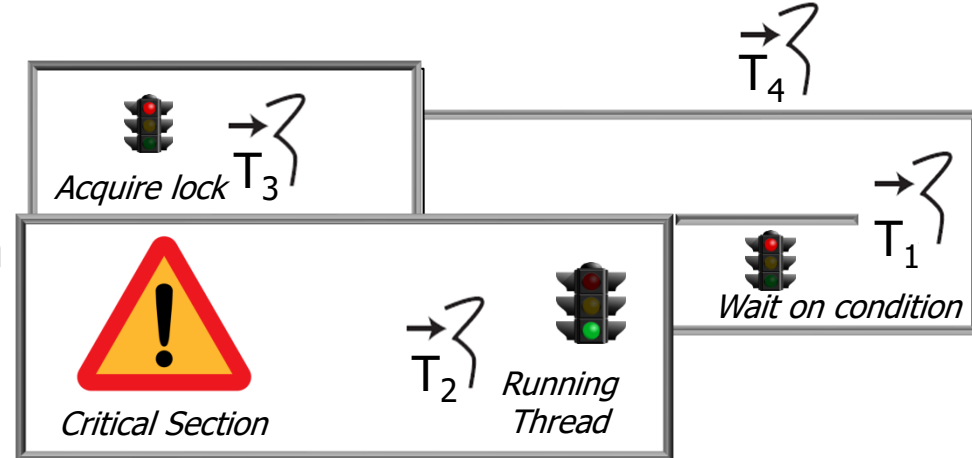 1. Only one thread at a time has mutually exclusive access to a critical section

Acquire lock T_2 T_3

T_4

Critical Section

T_1 *Running Thread*

See en.wikipedia.org/wiki/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

  2. Threads running in a monitor can block awaiting certain conditions to become true

  Acquire lock T_3

  T_4

  T_1

  Wait on condition

  Critical Section

  T_2 Running Thread

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

  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

T_3 T_4

Unblock on wait queue

Acquire lock

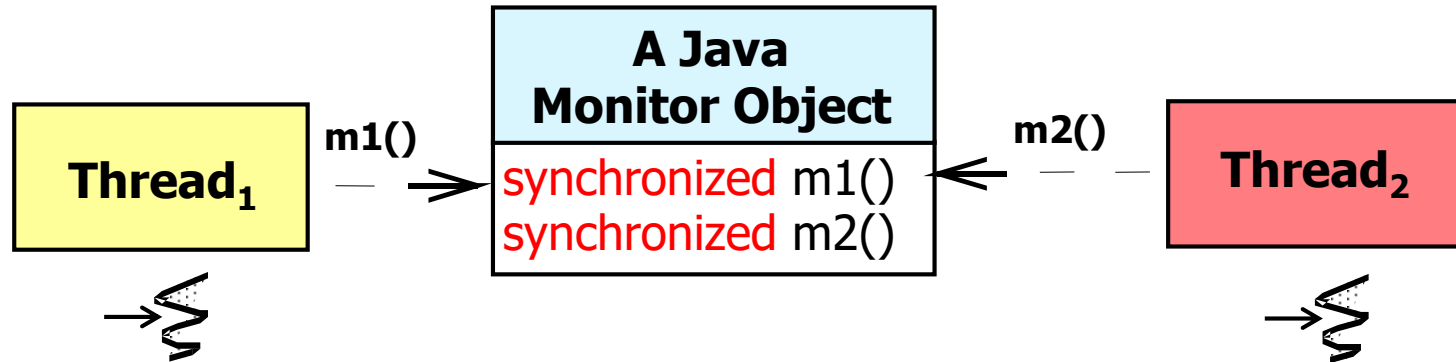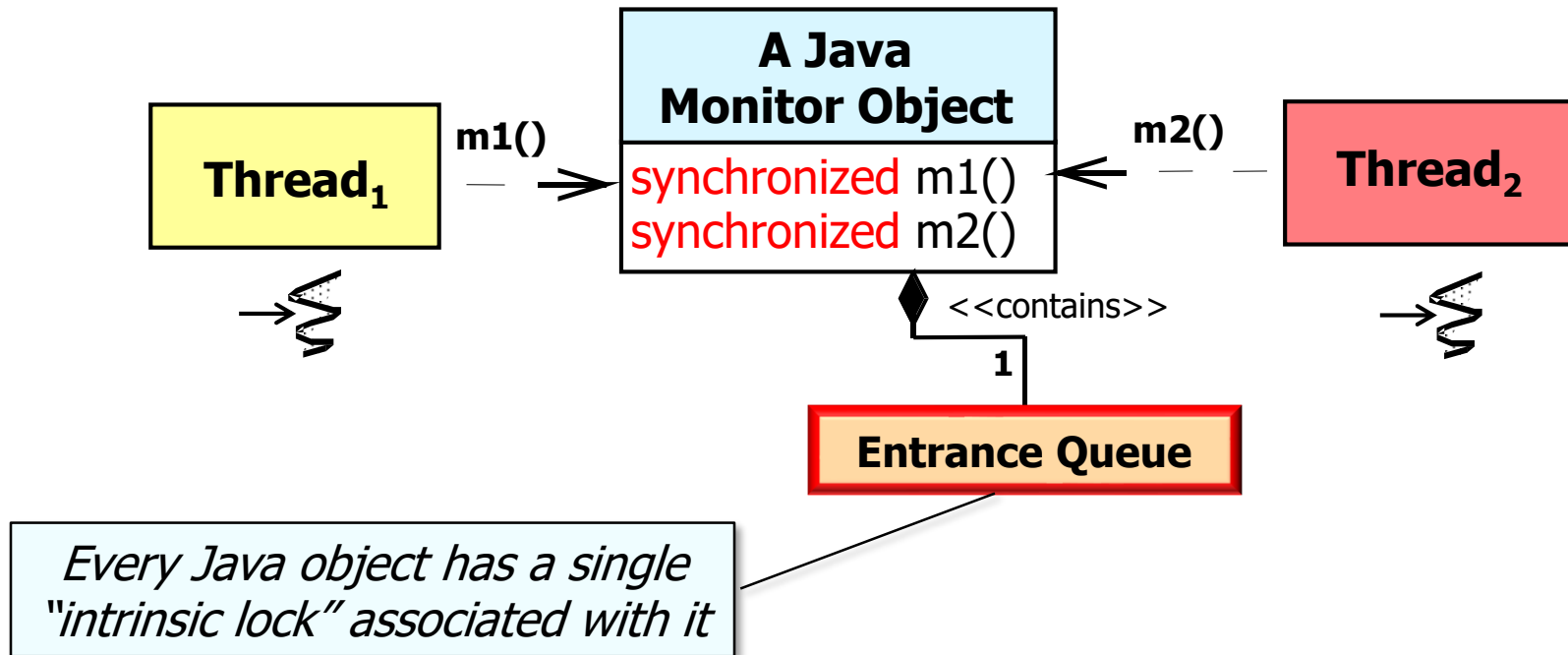T_1 Running Thread

Critical Section

T_2
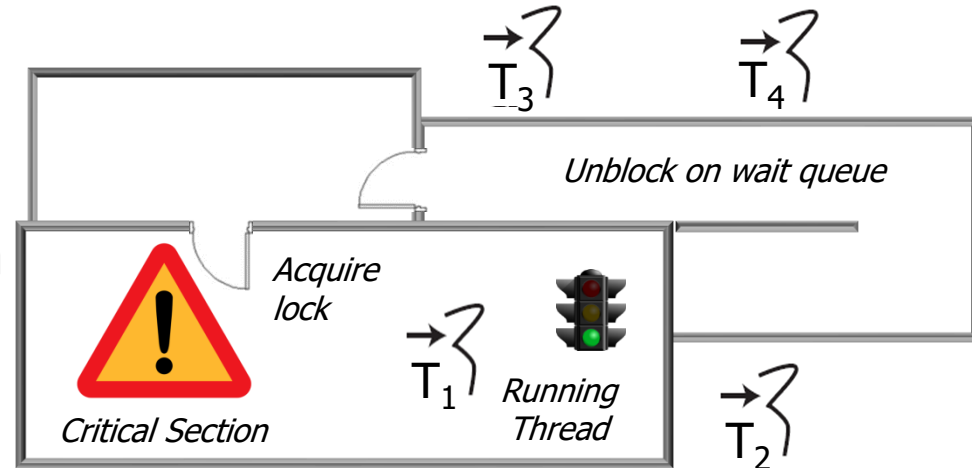
Overview of Built-in Java Monitor Objects

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

A Java Monitor Object

| Thread₁ | m1() | synchronized m1()
synchronized m2() | m2() | Thread₂ |
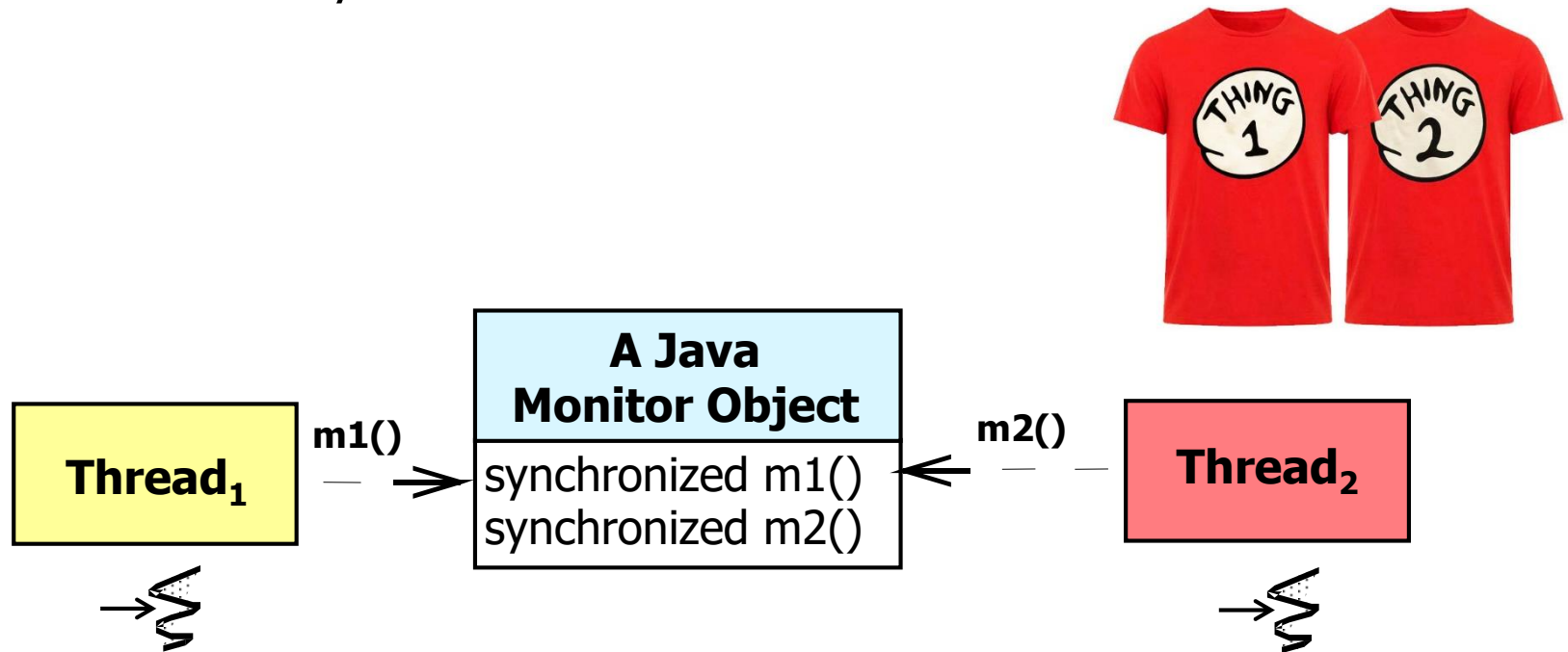
Overview of Java Built-in Monitor Objects

- 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

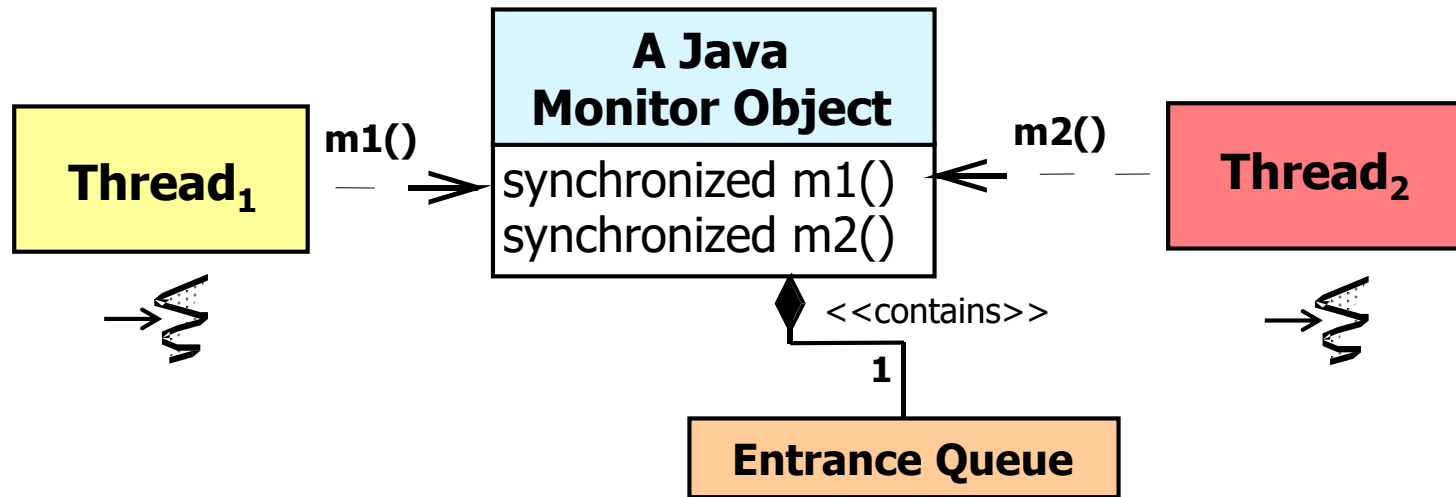# Overview of Java Built-in Monitor Objects

- 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
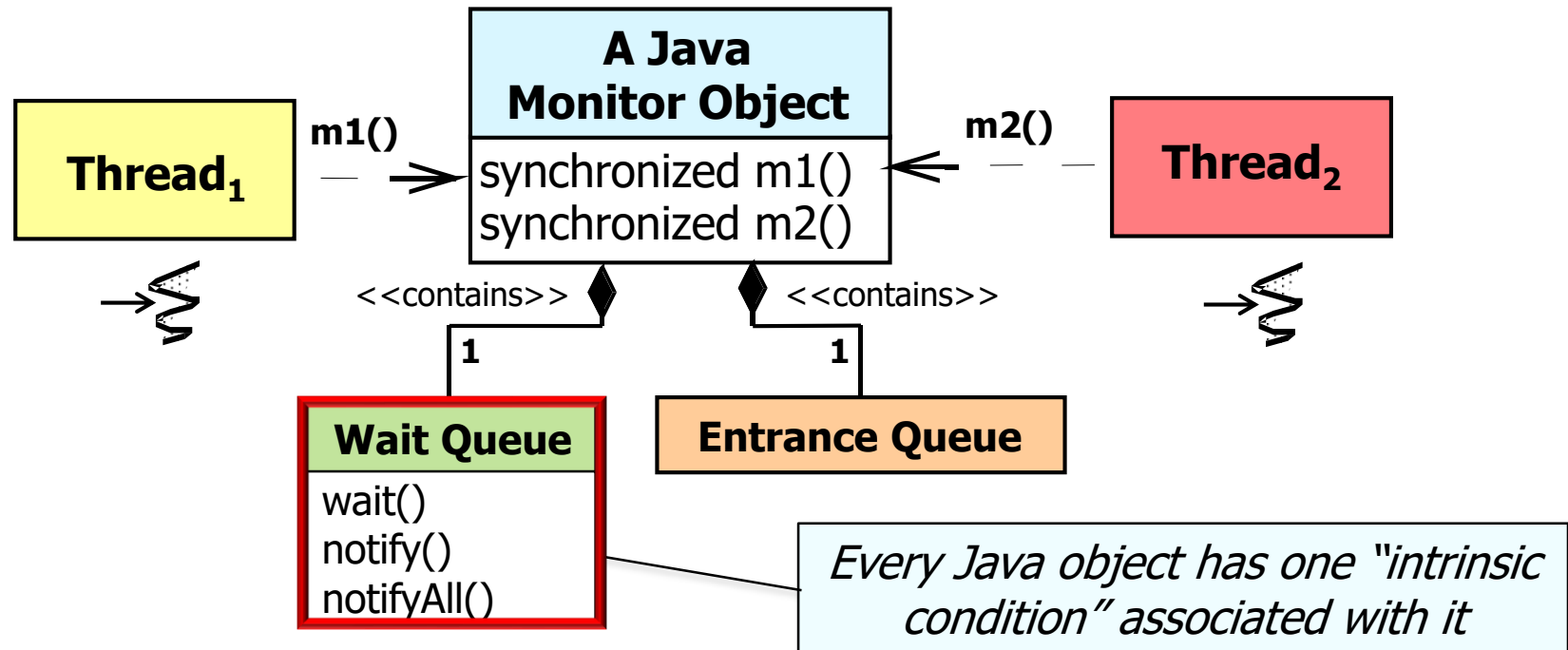
Overview of Java Built-in Monitor Objects

- 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.

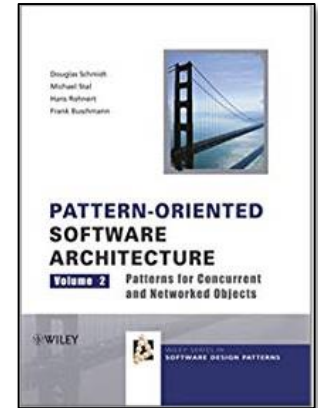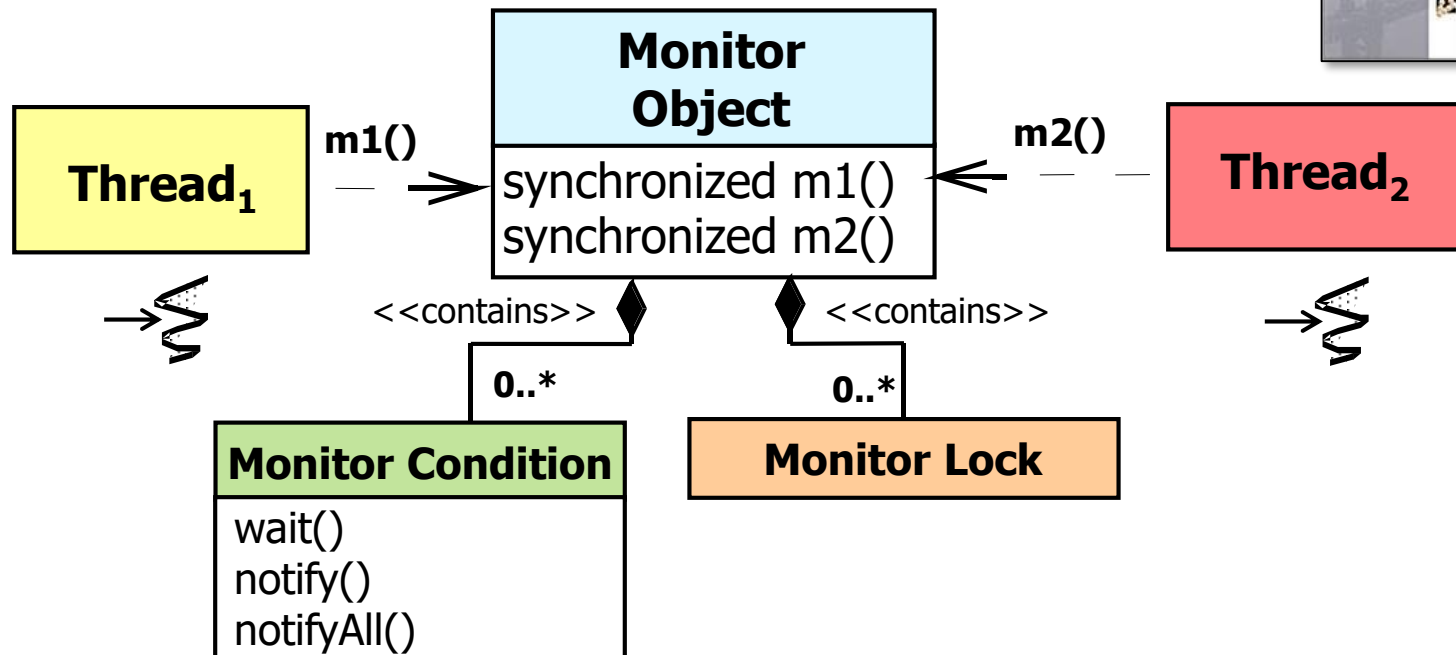# Overview of Java Built-in Monitor Objects

- 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.

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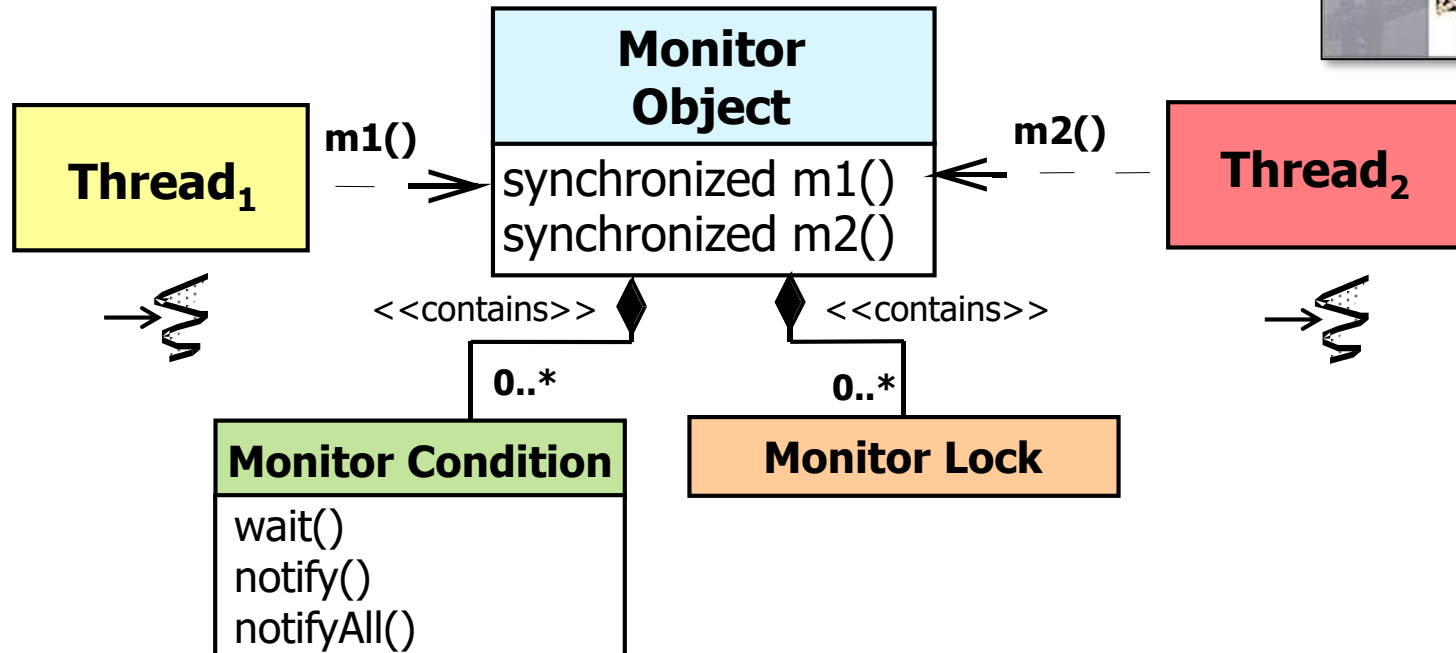
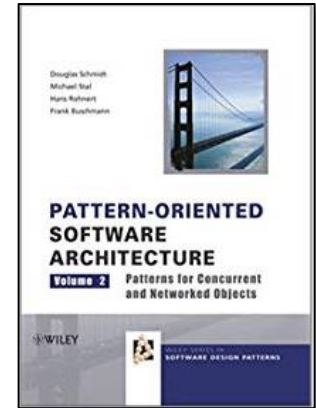# Overview of Java Built-in Monitor Objects

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

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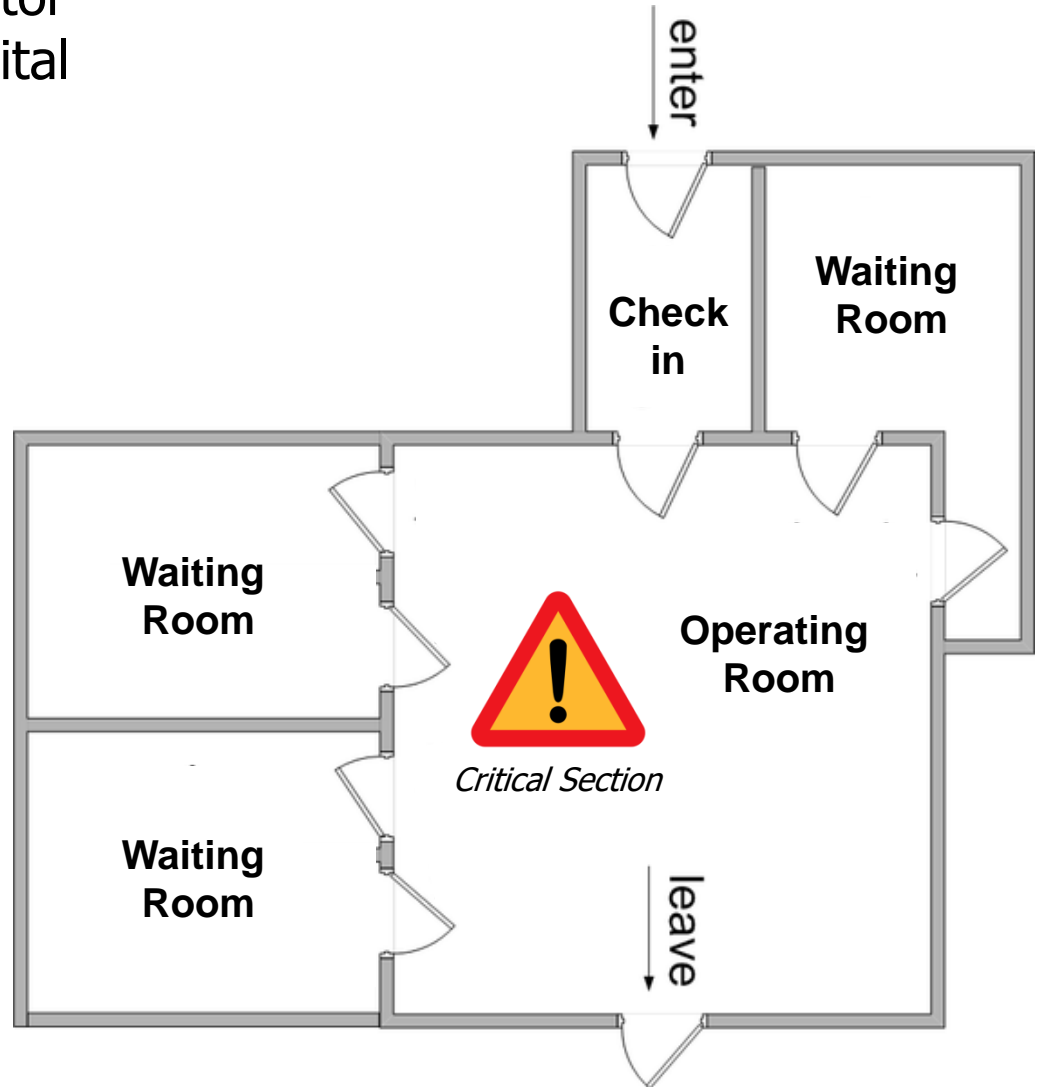
Overview of Java Built-in Monitor Objects

- 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

PATTERN-ORIENTED
SOFTWARE
ARCHITECTURE

Volume 2   Patterns for Concurrent and Networked Objects

WILEY

Monitor Object

Thread₁	m1() →	synchronized m1()

synchronized m1()
synchronized m2()

← m2()   **Thread₂**

<<contains>>   0..*   **Monitor Condition**

wait()
notify()
notifyAll()

<<contains>>   0..*   **Monitor Lock**

Human Known Use of Monitors

- A human known use of a monitor is an operating room in a hospital

enter

Check in

Waiting Room

Waiting Room

Operating Room

Critical Section

Waiting Room

leave

End of Java Monitor Objects: Introduction