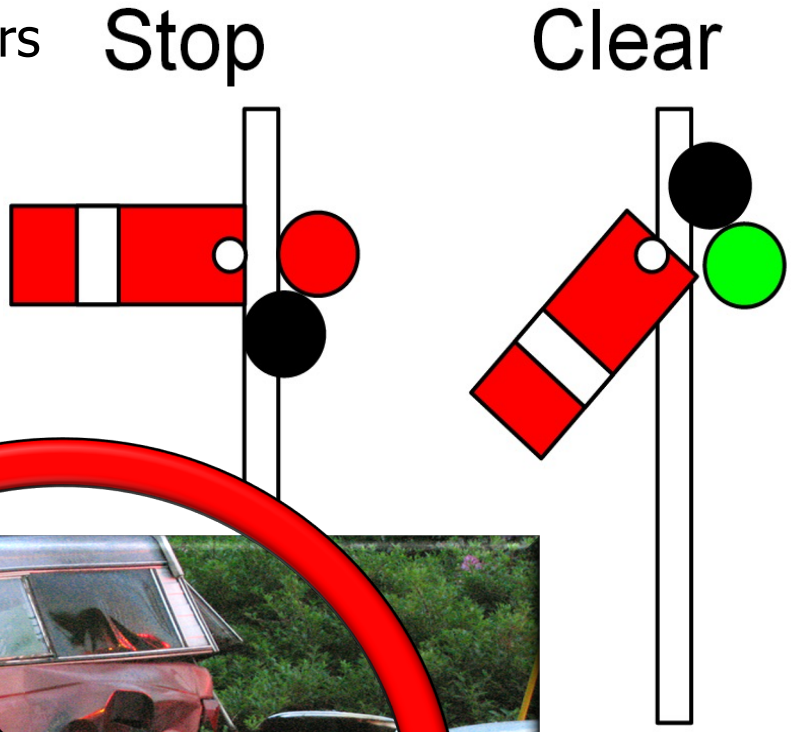
Overview of Java Synchronizers

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

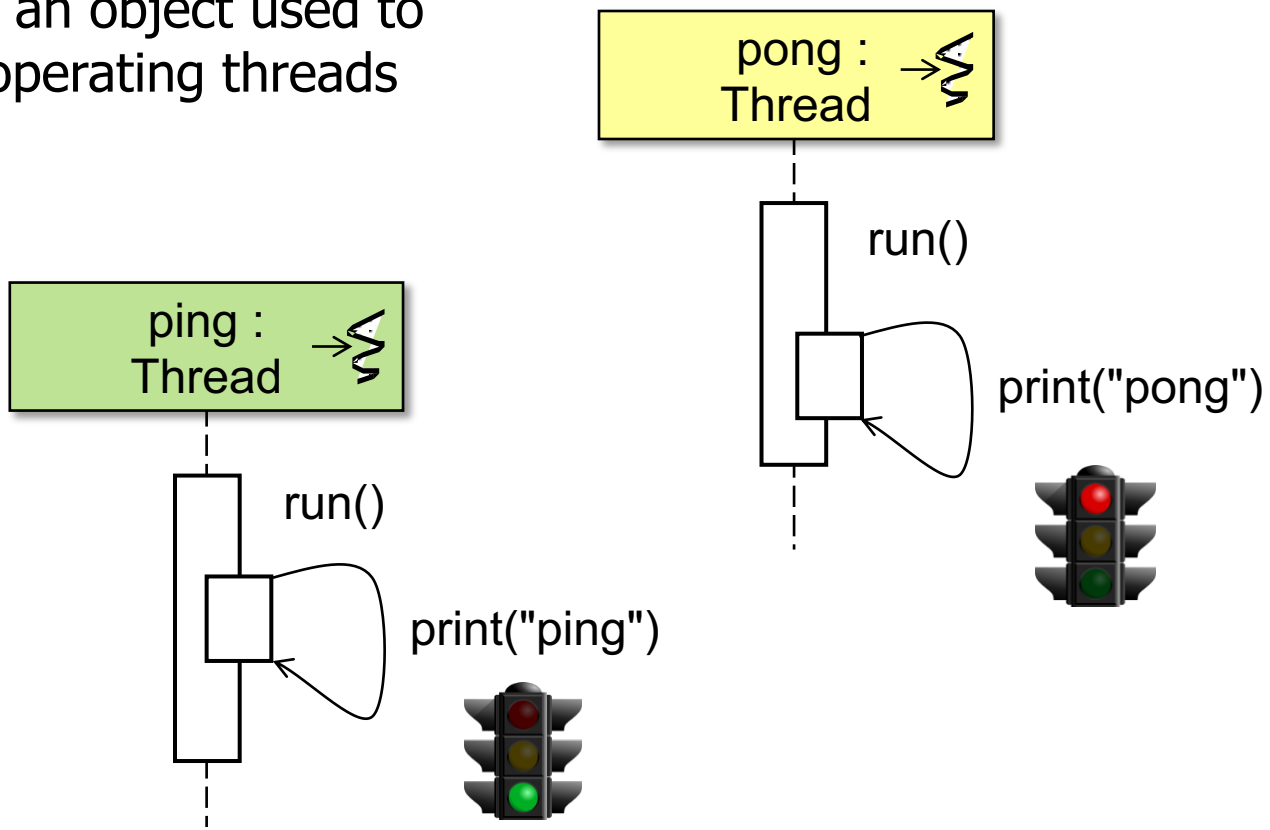
- Be aware of the Java memory model
- Understand the purpose of Java synchronizers



Overview of Java Synchronizers

Overview of Java Synchronizers

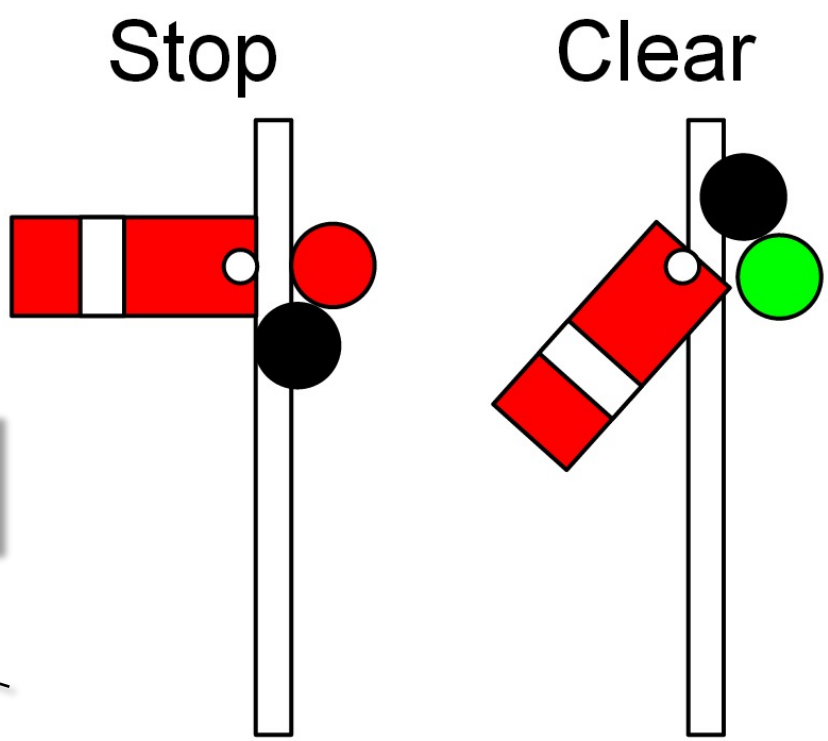
- A Java synchronizer is an object used to control the flow of cooperating threads based on its state



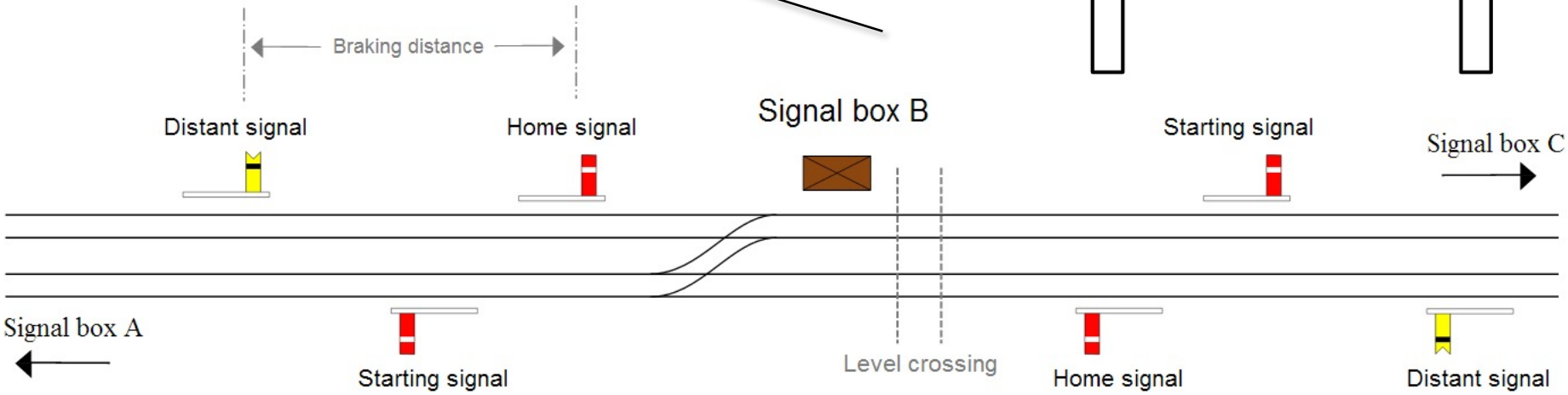
See en.wikipedia.org/wiki/Synchronization_(computer_science)

Overview of Java Synchronizers

- A Java synchronizer is an object used to control the flow of cooperating threads based on its state



Synchronization has long been essential to coordinate safety-critical physical processes



See en.wikipedia.org/wiki/Synchronization#Navigation_and_railways

Overview of Java Synchronizers

- Java synchronizers ensure interactions between threads obey certain properties

Overview of Java Synchronizers

- Java synchronizers ensure interactions between threads obey certain properties, e.g.
  - Don't corrupt shared mutable state

See henrikeichenhardt.blogspot.com/2013/06/why-shared-mutable-state-is-root-of-all.html

Overview of Java Synchronizers

- Java synchronizers ensure interactions between threads obey certain properties, e.g.
  - Don't corrupt shared mutable state

```
class NonAtomicOps {
  long mCounter = 0;

  void increment() {
    // Thread T1
    for (;;) mCounter++;
  }

  void decrement() {
    // Thread T2
    for (;;) mCounter--;
  }
  ...
}
```

Running increment() & decrement() concurrently yields undefined behavior since mCounter is shared mutable data

See tutorials.jenkov.com/java-concurrency/race-conditions-and-critical-sections.html

Overview of Java Synchronizers

- Java synchronizers ensure interactions between threads obey certain properties, e.g.
  - Don't corrupt shared mutable state

```
class AtomicOps {
  long mCounter = 0;

  synchronized void increment() {
    // Thread T1
    for (;;) mCounter++;
  }

  synchronized void decrement() {
    // Thread T2
    for (;;) mCounter--;
  }
  ...
}
```

Running increment() & decrement() concurrently yields correct behavior since mCounter is shared mutable data synchronized at the (coarse-grained) method level

See tutorials.jenkov.com/java-concurrency/synchronized.html

Overview of Java Synchronizers

- Java synchronizers ensure interactions between threads obey certain properties, e.g.
  - Don't corrupt shared mutable state

```
class AtomicOps {
  long mCounter = 0;

  void increment() { // Thread T1
    for (;;) synchronized
    { mCounter++; }
  }

  void decrement() { // Thread T2
    for (;;) synchronized
    { mCounter--; }
  }
  ...
}
```

Running increment() & decrement() concurrently yields correct behavior since mCounter is shared mutable data synchronized at the (fine-grained) statement level

See tutorials.jenkov.com/java-concurrency/synchronized.html

Overview of Java Synchronizers

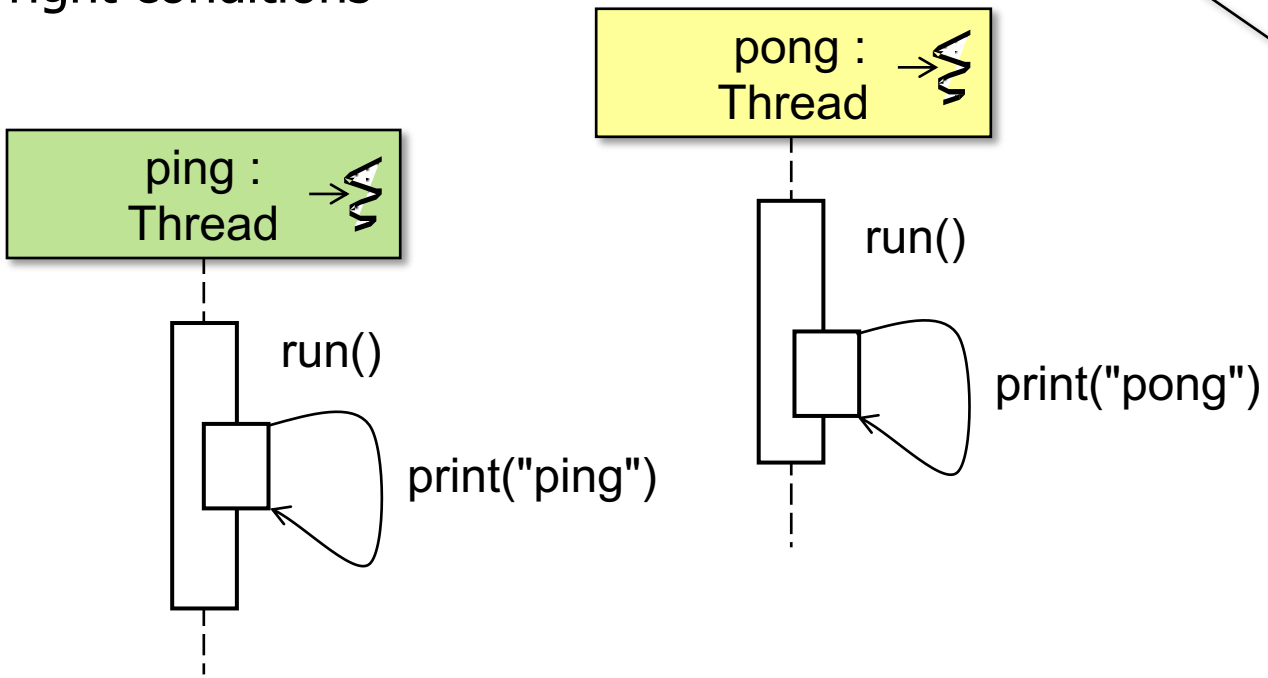
- Java synchronizers ensure interactions between threads obey certain properties, e.g.
  - Don't corrupt shared mutable state
  - Occur in the right order, at the right time, & under the right conditions

Overview of Java Synchronizers

- Java synchronizers ensure interactions between threads obey certain properties, e.g.
  - Don't corrupt shared mutable state
  - Occur in the right order, at the right time, & under the right conditions

The unsynchronized version is buggy

```
% java PingPongWrong
Ready...Set...Go!
Ping!(1)
Ping!(2)
Ping!(3)
Ping!(4)
Ping!(5)
Ping!(6)
Ping!(7)
Ping!(8)
Ping!(9)
Ping!(10)
Pong!(1)
Pong!(2)
Pong!(3)
Pong!(4)
Pong!(5)
Pong!(6)
Pong!(7)
Pong!(8)
Pong!(9)
Pong!(10)
Done!
```

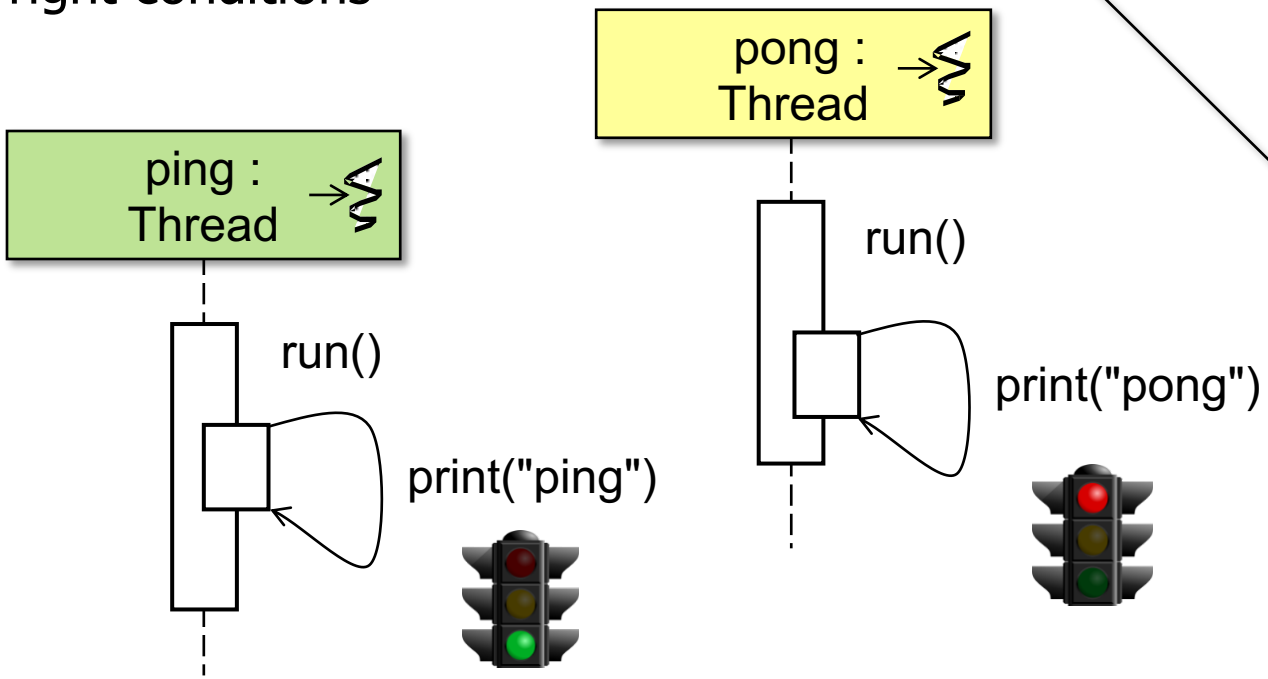


See upcoming lesson on "*Java Semaphore: Coordinating Threads*"

Overview of Java Synchronizers

- Java synchronizers ensure interactions between threads obey certain properties, e.g.
  - Don't corrupt shared mutable state
  - Occur in the right order, at the right time, & under the right conditions

The synchronized version coordinates the threads properly




```
% java PlayPingPong
Ready...Set...Go!
Ping!(1)
Pong!(1)
Ping!(2)
Pong!(2)
Ping!(3)
Pong!(3)
Ping!(4)
Pong!(4)
Ping!(5)
Pong!(5)
Ping!(6)
Pong!(6)
Ping!(7)
Pong!(7)
Ping!(8)
Pong!(8)
Ping!(9)
Pong!(9)
Ping!(10)
Pong!(10)
Done!
```

End of Overview of Java Synchronizers