

Thread Coordination with Notify and Wait

Objects and Threads

- Each object has a lock that can be obtained and released
 - Java uses an object's synchronized methods to control thread access the object
- Each object also has a waiting area for threads that need something other threads can provide
 - wait and notify

synchronization

t1 → synchronized void deposit(int x)
t2 → synchronized int withdraw()

the threads are essentially independent --
no order enforced

what if ordering required? (put must precede get)

t1 → synchronized void put(int x)
t2 → synchronized int get()

synchronization alone cannot guarantee an
ordering

coordination solution #1

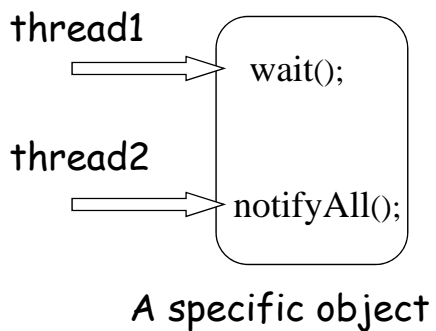
```
int value = -1; // no value yet

synchronized void put(int x) {
    value = x;
}

synchronized int get() {
    while (value == -1)
        { sleep(500); }
    return value; }
```

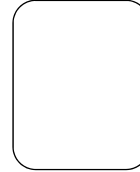
notify and wait

- wait ()
 - wait for in an object's waiting area
 - hope that another thread will execute notify or notifyAll
- notifyAll()
 - wake up all threads waiting in the object's waiting area

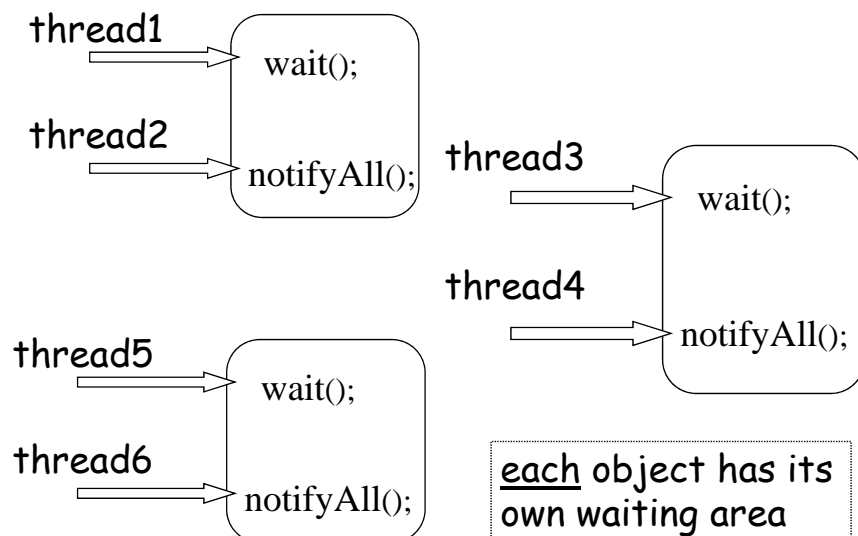


notify and wait (methods in class Object)

- All object's understand notify(), notifyAll() and wait()
- Use notify & wait in your code if:
 - you have get & put methods
 - you want consumer threads to wait on producer threads
 - you want producer threads to notify consumers



wait and notify is a communication mechanism



coordination solution #2

```
int value = -1; // no value yet

synchronized void put(int x) {
    value = x; notifyAll();
}

synchronized int get() {
    while (value == -1)
        { wait(); }
    return value; }
```

guarantees
that a put
will always
occur
before a
get

no busy
wait loop