A6
AND A START ON THREADS AND CONCURRENCY

Lecture 24 – CS2110 – Fall 2014

Administrative matters

PRELIM 2

- Thursday, 5:30pm, Statler Auditorium (even ids)
- Thursday, 7:30pm, Statler Auditorium (odd ids)
- Those authorized to have more time or a quieter space: 5:00PM onward, Statler 196

- Recitation this week: Those in recitation

  2110-208 Tu 1:25PM - 2:15PM in Olin Hall 218.

  Leon is out of town. Please go to room Olin 245

  instead and attend Eric Perdew's recitation.

Concurrency

- Modern computers have "multiple cores"
  - Instead of a single CPU on the chip
  - 4-8 common on laptops
  - And even with a single core (CPU) your program may have more than one thing "to do" at a time
  - Argues for having a way to do many things at once
- Finally, we often run many programs all at once
- And assignment A6 is filled with such concurrency!

What is a Thread?

- *A separate "execution" <u>that runs within a single program</u> and can perform a computational task independently and concurrently with other threads*
- Many applications do their work in just a single thread: the one that called main() at startup
  - But there may still be extra threads...
    - Garbage collection runs in a "background" thread
    - GUIs have a separate thread that listens for events and "dispatches" upcalls
- Today: learn to create new threads of our own and see threads in action in assignment A6.

What is a Thread?

- A thread is a kind of object that “independently computes”
  - Needs to be created, like any object
  - Then “started”. This causes some method (like main()) to be called. It runs side by side with other threads in the same program, and they see the same global data
- The Mac has an app, Activity Monitor, that shows you what apps are running and how many threads each has. We show you this on Gries’s laptop. The PC should have a similar app. Find it and play with it!
- On Gries’s computer at the moment, the Mail app 22 threads, Safari has 13. DropBox has 41. Eclipse has 34.

Concurrency

- *Concurrency* refers to a single program in which several threads are running simultaneously
  - Special problems arise
  - They see the same data and hence can interfere with each other, e.g. one thread modifies a complex structure like a heap while another is trying to read it

Class Thread in Java

- Threads are instances of class Thread
  - Can create many, but they consume space & time
- The Java Virtual Machine created the Thread that executes your method main.
- Threads have a priority
  - Higher priority Threads are executed preferentially
  - A newly created Thread has initial priority equal to the Thread that created it (but can change)

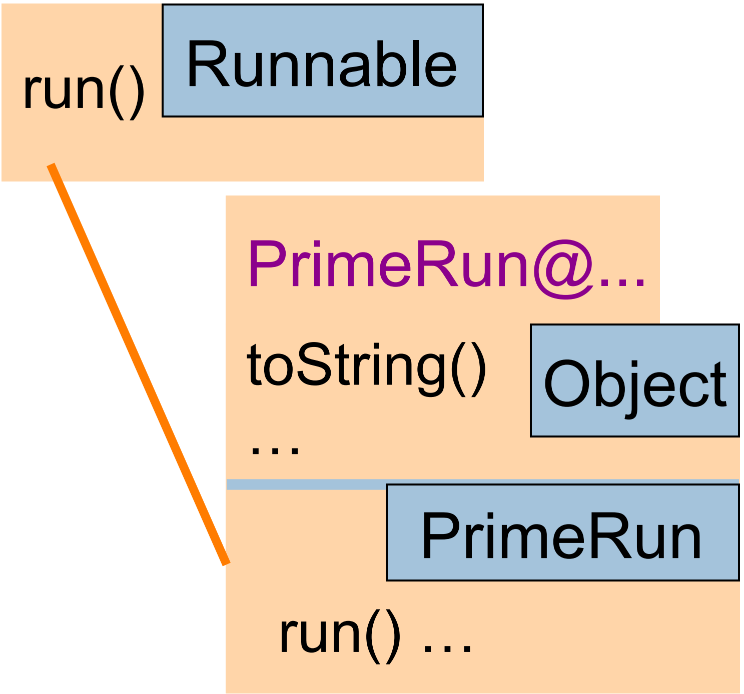
Runnable object, running in a new Thread

```
class PrimeRun
           implements Runnable {
    long a, b;

    PrimeRun(long a, long b) {
        this.a= a; this.b= b;
    }

    public void run() {
        // compute primes
        // in a..b
        ...
    }
}
```

```
PrimeRun p=
         new PrimeRun(143, 195);
new Thread(p).start();
```



Method start() will call p's method run() in the new thread of execution

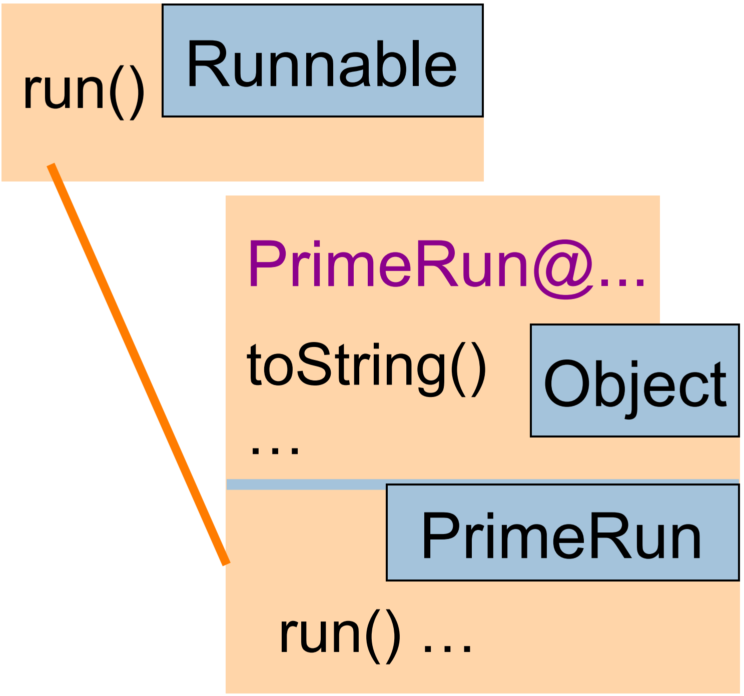
Runnable object, running in a new Thread

```
class PrimeRun
        implements Runnable {
   long a, b;

   PrimeRun(long a, long b) {
      this.a= a; this.b= b;
   }

   public void run() {
      // compute primes
      // in a..b
      ...
   }
}
```

```
PrimeRun p=
       new PrimeRun(143, 195);
p.run();
```



No new thread. run() runs in same thread as its caller

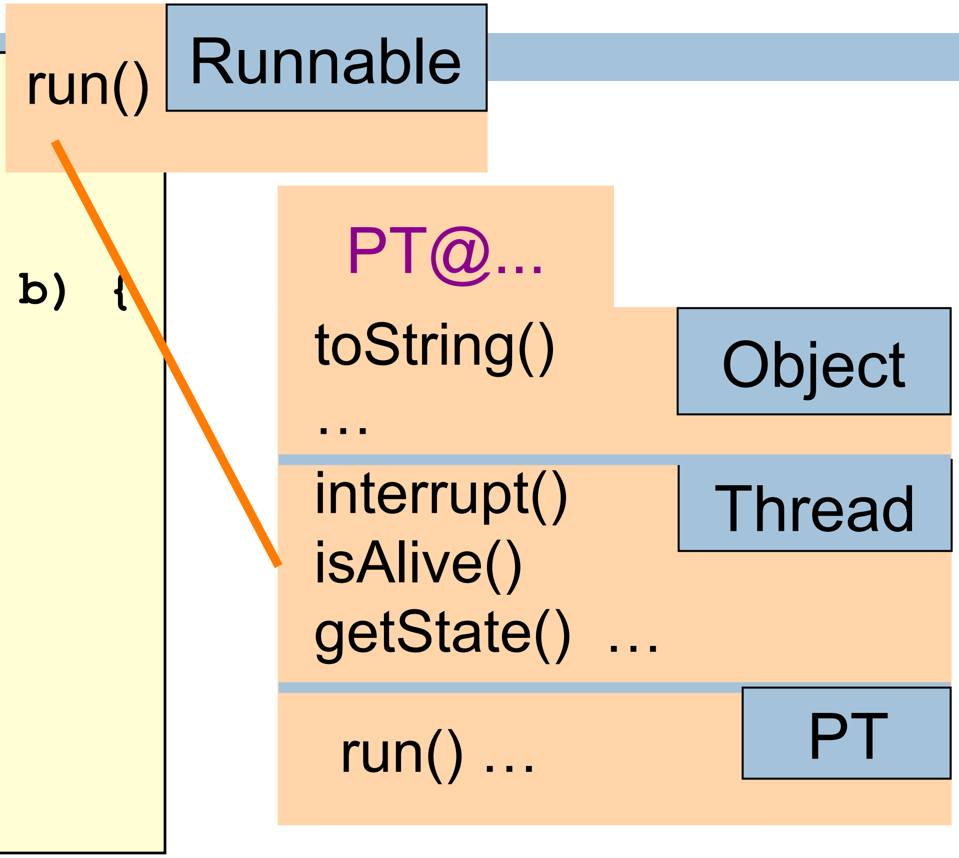
Creating a new Thread (Method 2)

```
class PrimeThread
       extends Thread {
   long a, b;

   PrimeThread(long a, long b) {
      this.a= a; this.b= b;
   }

   public void run() {
      // compute primes
      // a..b
      ...
   }
}
```

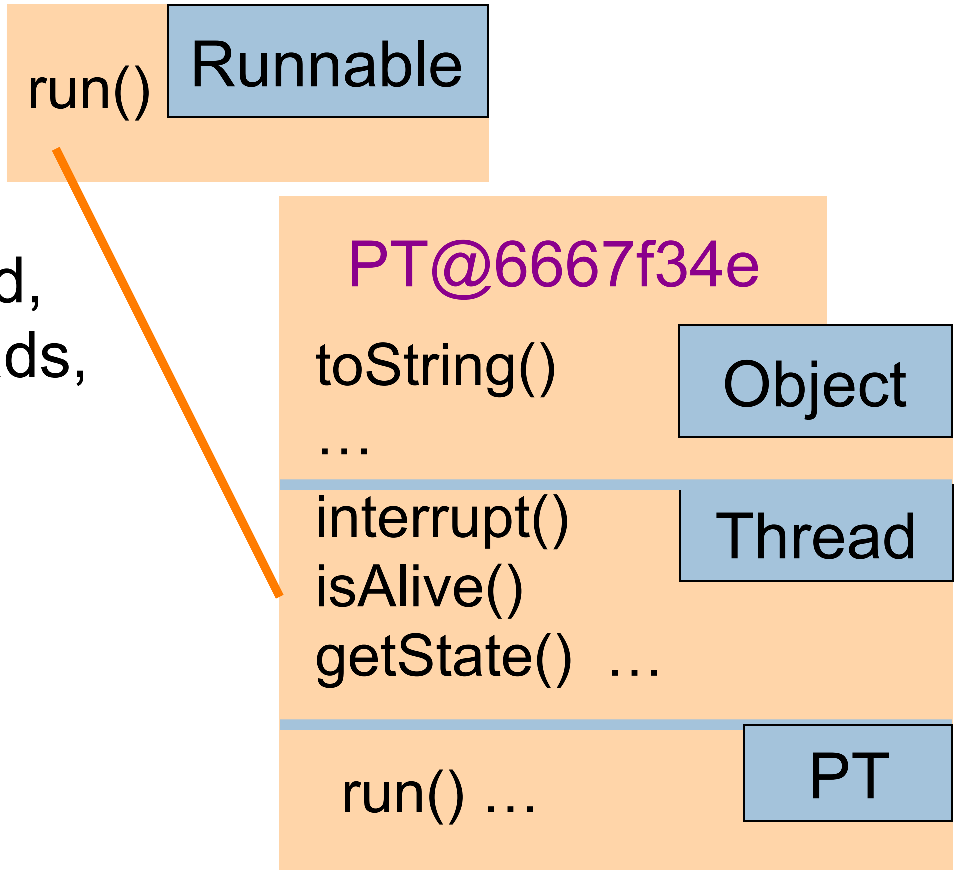
```
PrimeThread p=
  new PrimeThread(143, 195);
p.start();
```



Class Thread has methods to allow more control over threads

Class Thread has methods for handling threads

You can interrupt a thread,
maintain a group of threads,
set/change its priority,
sleep it for a while,
etc.



Class PT extends Thread, which implements Runnable

Now to Assignment A6B: Shipping Game

In a nut shell:

- Bunch of cities with roads between them (a graph)
- Parcels sitting at cities, have to be trucked to other cities
- Trucks, all at a city called Truck Depot, have to be used to move each parcel from its start city to its destination city. Then return Trucks to the Home Depot
- YOU have to write the program that tells the Trucks what to do!
- Efficiency is important! Use shortest paths where possible.

We DEMO A6B

Assignment A6B: Shipping Game

Assignment A6 is developed Michael (Shnik) Patashnik Undergrad TA

A&S, studying Economics and CS

Other CS2110 staff involved: Eric Chahin, Alex Fusco, Aaron Nelson, Alexandra Anderson.

Which one of *you* will be the next one to help us develop our assignments?

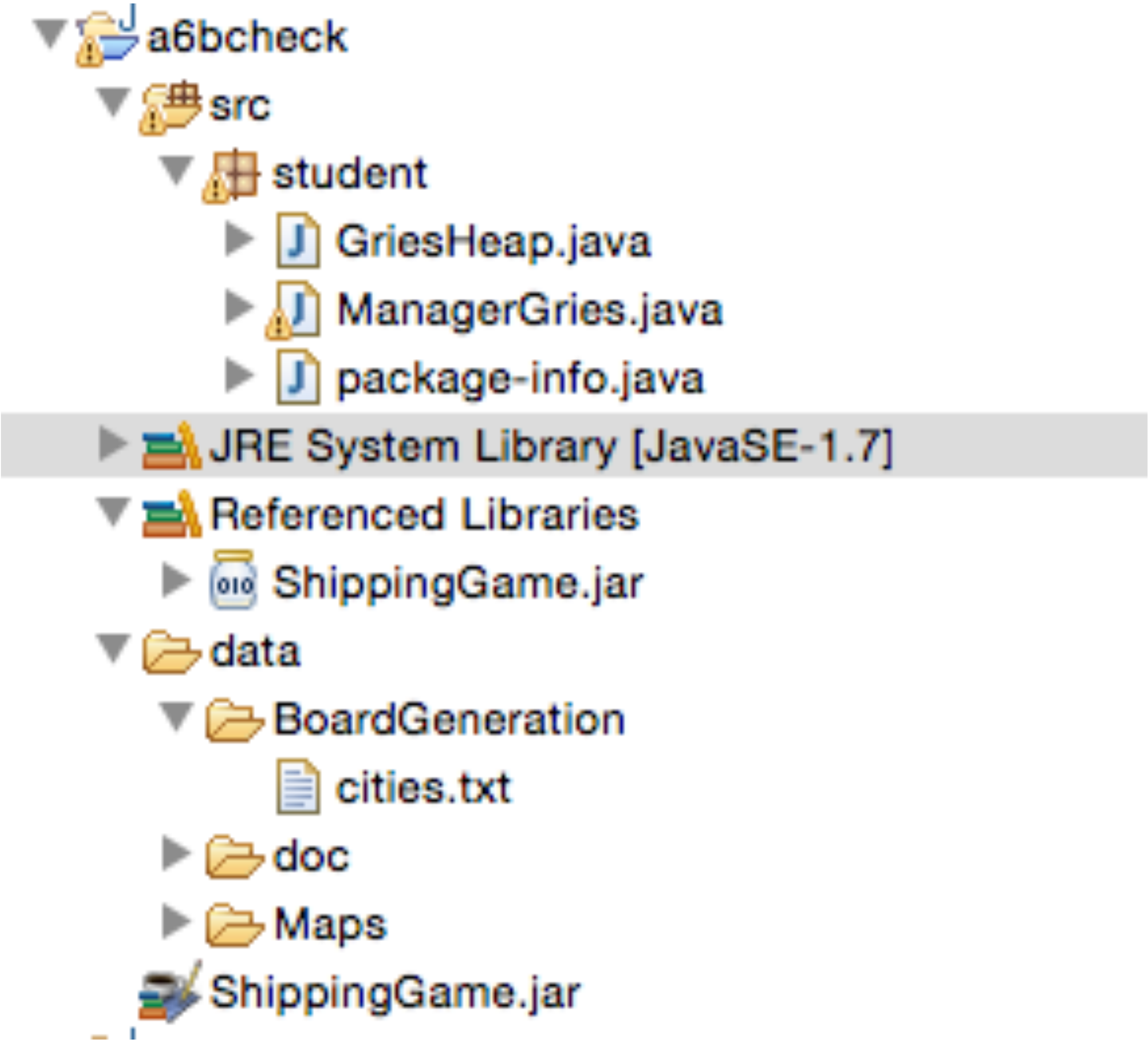
Ideas for A6b

- Spend a lot of time *thinking* about the design, looking at specs of Truck, Parcel, manager, etc. Look at class diagram on page 7 of the handout.
- Given a truck that has to pickup a Parcel, need to find a quickest/shortest path to where Parcel is. Dfs and bfs won't do. Probably need a version of shortest-path algorithm from a start node to another.
- Each Truck has a field UserData in which you can store anything you wish. E.g. a path from current location to destination of the Parcel it is carrying.
- Each Parcel also has a UserData field

You class MyManager extends game.Manager

We don't give you Java source files. We give you only the .class files and good specs of the classes. Specs are in Data/doc

We demo looking at API specs



```
▼ a6bcheck
  ▼ src
    ▼ student
      ▶ GriesHeap.java
      ▶ ManagerGries.java
      ▶ package-info.java
  ▶ JRE System Library [JavaSE-1.7]
  ▼ Referenced Libraries
    ▶ ShippingGame.jar
  ▼ data
    ▼ BoardGeneration
        cities.txt
    ▶ doc
    ▶ Maps
    ShippingGame.jar
```

Your main task

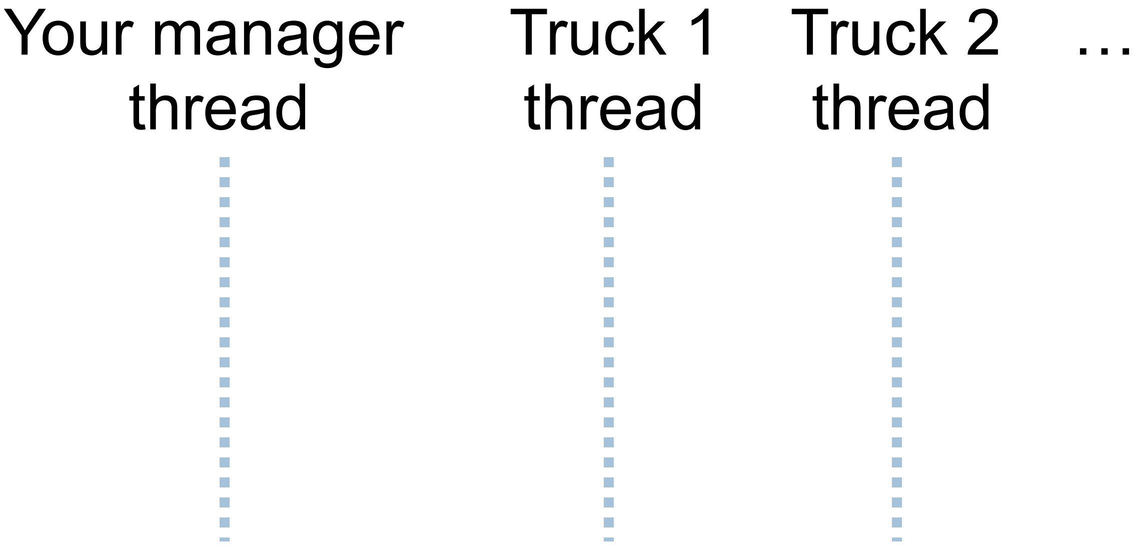
```
public class YourManager extends Manager {

    public @Override void run() {
        Look at map, parcels truck, do preprocessing
        and give Trucks their initial instructions
    }

    public  @Override void truckNotification(Truck t,
                                              Notification message) {
        Called when event happens with Truck t —it
        waited to long and is prodding, it arrived at a city,
        there’s a parcel at the city, etc. This method should
        give the truck directions on how to proceed.
    }
```

Manager and trucks run in their own thread

```
public class YourManager extends Manager {
    public @Override void run() {… }
    public @Override void
          truckNotification(Truck t, Notification mess) { ... }
}
```



Make sure shared variables don’t cause problems.

E.g. Two Trucks try to take the same Parcel

A6 Efficiency

You want to get the best score possible! How much you do, what you do, depends your time constraints, your abilities, whether your find this assignment fun. Here are things to consider.

It costs for a Truck to wait
It costs for a Truck to travel
It costs for a Truck to pick up and drop a Parcel
A LOT is gained by dropping a Parcel at its destination
Parcel Payoff is a LOT more if the truck that delivers it has the same color as the Parcel.

Big problem with shared data: a small example

Suppose x is initially 5

Thread t1	Thread t2
`x= x + 1;`	`x= x + 1;`

... after finishing, x = 6! Why?

Sequence:

t1 evaluates x+1 to get 6
t2 evaluates x+1 to get 6
t2 stores its value 6 in x
t1 stores its value 6 in x

We need ways to prevent this from happening. There are several. Here, we explain only Java's synchronization mechanism

Getting concurrent programs right is much much harder!

The synchronized block

```
Stack<String> s= new Stack<String>();
```

```
if (… ) {
    This is a block of code
}
```

```
synchronized(s) {
    This is a synchronized
    block of code
}
```

A block synchronized on an object prohibits any other thread from accessing the object while the block is being executed.

The synchronized block is a primary tool for eliminating shared data problems. (There are others)

Solution – with synchronization

```
private Stack<String> s= new Stack<String>();

public void doSomething() {
    String st;
    synchronized (s) {
        if (s.isEmpty()) return;
        st= s.pop();
    }
    code to do something with st
}
```

synchronized block

- Put critical operations in a **synchronized** block
- The **stack** object acts as a lock
- Only one thread can own the lock at a time
- Make synchronized blocks as small as possible

Solution – Locking

You can lock on any object, including **this**

```
public void doSomething(){
    synchronized (this) {
        body
    }
}
```

Note: the whole body is synchronized on **this**. There's a shorthand for this in Java

is equivalent to

```
public synchronized void doSomething(){
    body
}
```

Solving the shared x= x+1 problem

```
public class Ctr {
    int x= 0;
    public synchronized void inc {
        x= x + 1;
    }
}
```

```
public class T extends Thread {
    Ctr ctr;
    public T(Ctr c) { ctr= c;}
    public void run() {
        … ctr.inc(); …
    }
}
```

```
Ctr c= new Ctr();
Thread t1= new T(c);
Thread t2= new T(c);

T1.start();
T2.start();
```

T1 and T2 share a counter object. They can try to increment x at the same time (by calling inc), but one must wait.

Threads and synchronization in A6

A *lot* of synchronization happens behind the scenes in A6:

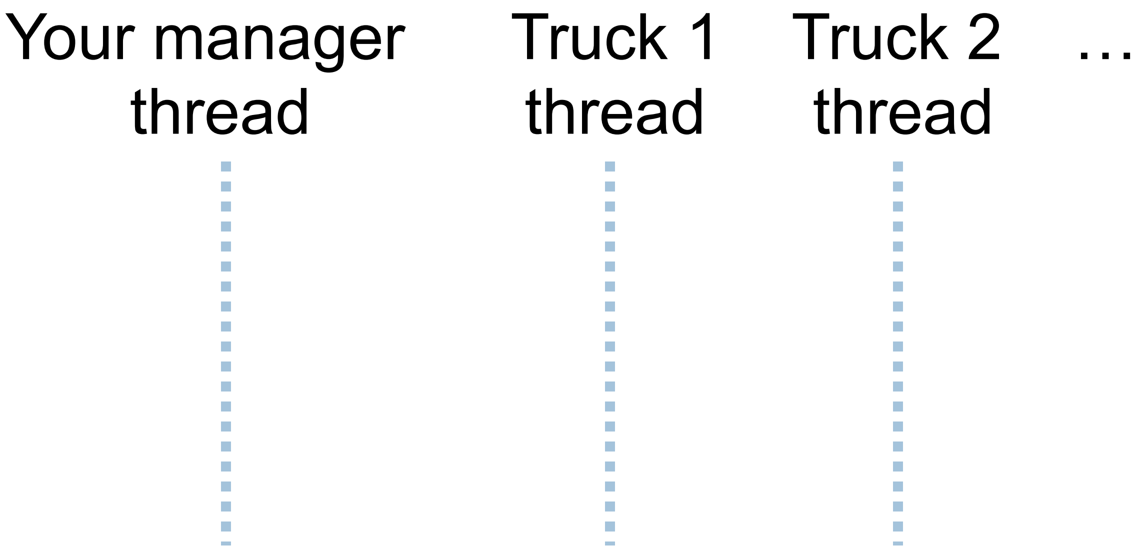
- The manager that you write is a Thread.
- Each Truck is a Thread.

Depending on your implementation, you may or may not have to use synchronization primitives in your part.
Most of you will not use synchronized blocks at all.

Just be careful and ask yourself whether concurrency can cause problems. E.g. can two trucks try to pick up the same Parcel at the same time?

Manager and trucks run in their own thread

```
public class YourManager extends Manager {
    public @Override void run() {… }
    public @Override void
          truckNotification(Truck t, Notification mess) { ... }
}
```



Make sure shared variables don’t cause problems.

E.g. Two Trucks try to take the same Parcel

Your method run(): Preprocessing

```
for Parcel p do
    Choose a truck t to deliver p.
    Store p in a data structure in t's user data.
end
```

How to chose? It’s up to you.
How to store data? It’s up to you.

Your truckNotification(Truck t, Notification mess)

```
// Always start with first if
if preprocessing not done then return;

if there are no Undelivered Parcels
    then Route t home and return;

if t holding a parcel then
        Route t to parcel's destination,
        drop it off if there
else  Find next parcel assigned to t,
      route to that parcel
```

Remember: several threads (Trucks) may be executing this at the same time. If shared data structures used, must make sure concurrency doesn't create problems

Truck t calls this method to say that it has done something or it is waiting for further instructions.

Synchronized collections

Study class Collections and the methods before working on the assignment:

synchronizedCollection

synchronizedSet

synchronizedSortedSet

synchronizedList

synchronizedMap

synchronizedSortedMap