

```

class MyThread extends Thread {
    public MyThread (int x) {
        a = x;
    }
    public void run () {
        b = a * a;
    }
    public int getb () {
        return b;
    }
    private int a;
    private int b;
}

MyThread t1 = new MyThread(4);
t2 = (6);
t3 = (2);

t1.start();
t2.start();
t3.start();

t1.join();
t2.join();
t3.join();

int sum = t1.getb() + t2.getb() + t3.getb();
    
```

always void
no arg

order doesn't matter
in this case

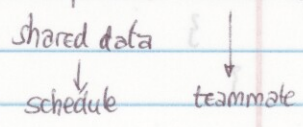
- ① class extends Thread
- ② implement "run" function
- ③ create thread objects
- ④ objects.start(); //not run
- ⑤ (usually) .join(); //wait until run() ends;

SCHEDULING
//no guarantee on
when thread starts,
finishes, and their
orders

```

class TeamMate {
    public TeamMate (Schedule s) {
        sch = s;
    }
    public void run () {
        //find available slot in sch, set meeting time
    }
    private schedule sch;
}

m1 = new TeamMate (mySchedule);
m2 = new TeamMate (mySchedule);
//problem: may choose same timeslot
//independent threads competing for
shared data
    
```



possible solution:

```

no concurrency {
    m1.start();
    m1.join();
    m2.start();
    m2.join();
}
    
```