



# Core Java

Trainer: Nilesh Ghule



# Thread creation

extends Thread

```
class MyThread extends Thread {
```

```
    ⋮  
    @Override  
    void run() {  
        // code  
    }
```

}

in main(),

```
MyThread t1 = new MyThread();  
t1.start();
```

implements Runnable

```
class MyRunnable implements Runnable {
```

```
    @Override  
    void run() {  
        // code  
    }
```

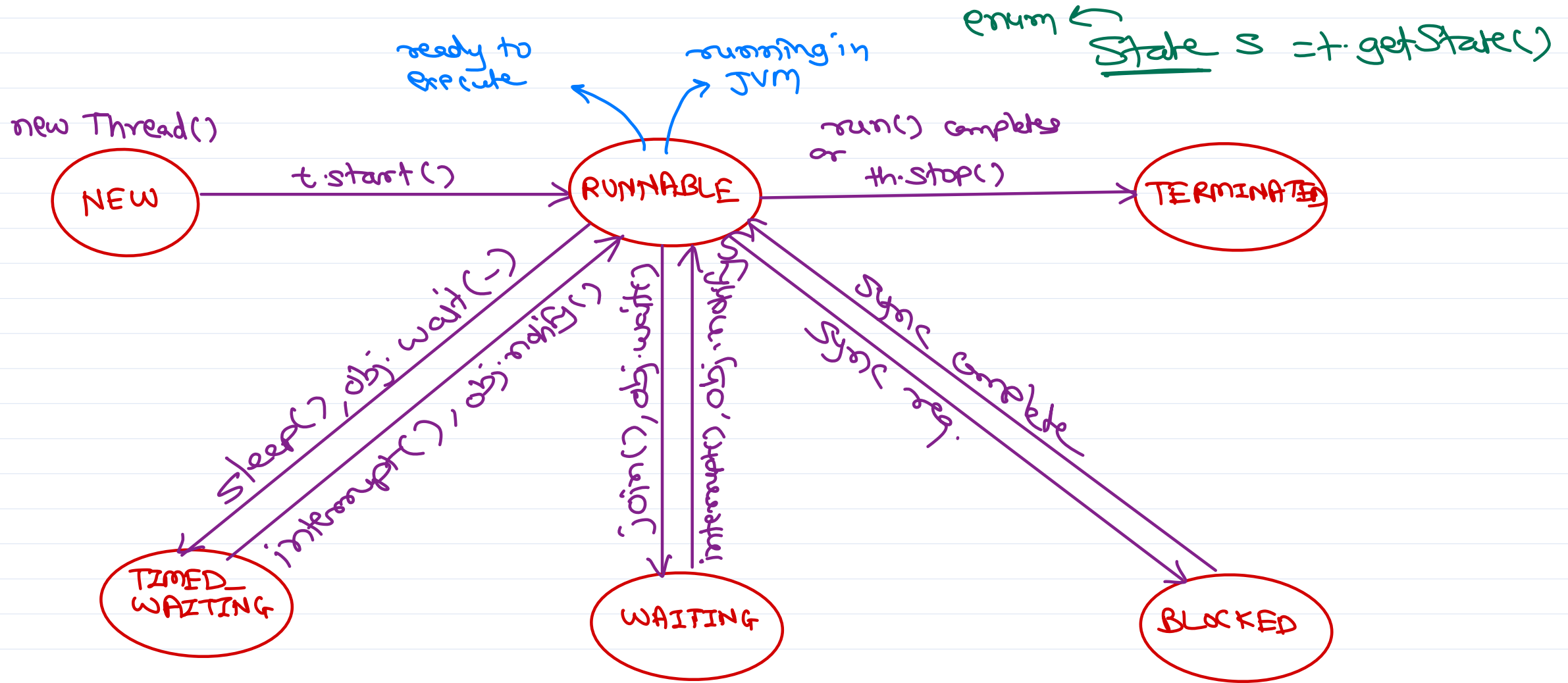
}

in main(),

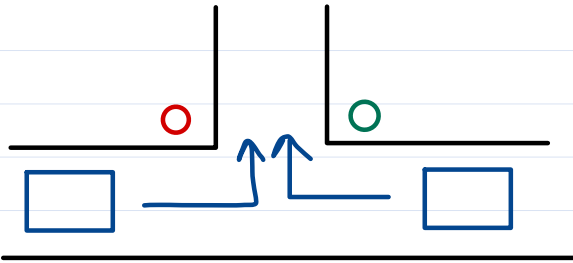
```
MyRunnable mr = new MyRunnable();  
Thread t2 = new Thread(mr);  
t2.start();
```



# Thread life cycle



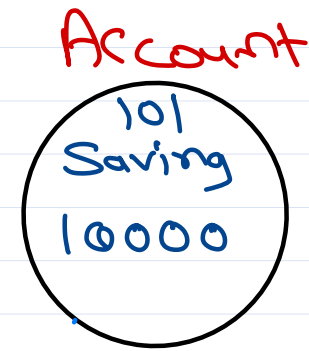
# Race condition



Race condition: Multiple threads/processes try to access same resource at same time.

Synchronization: Allowing only one thread to access the resource while blocking others, so that resource can be accessed safely.

OS/JVM provides specialized objects to sync threads e.g. Monitor, ...



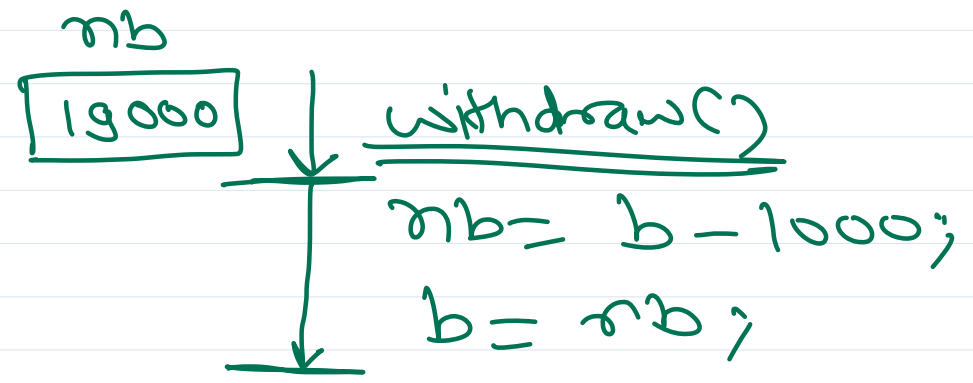
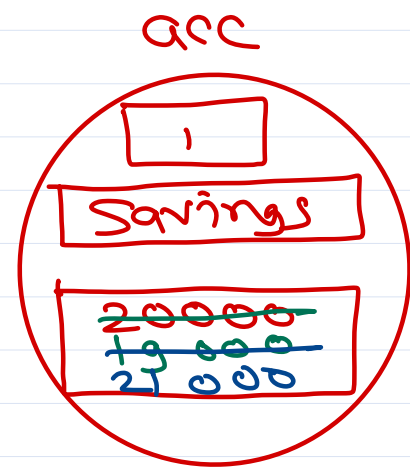
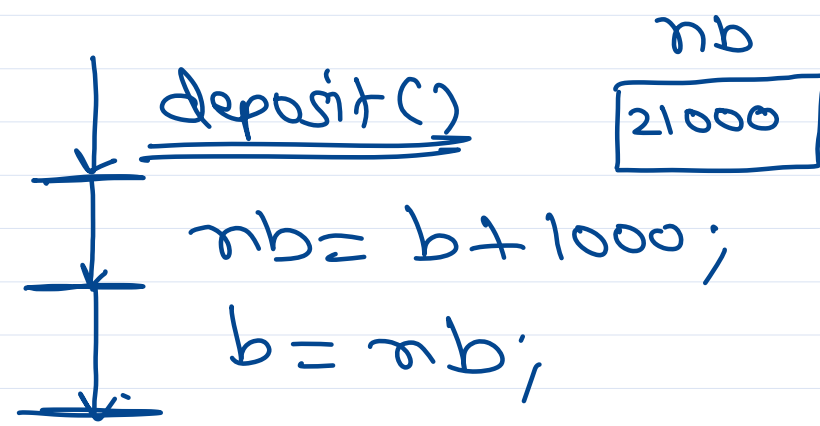
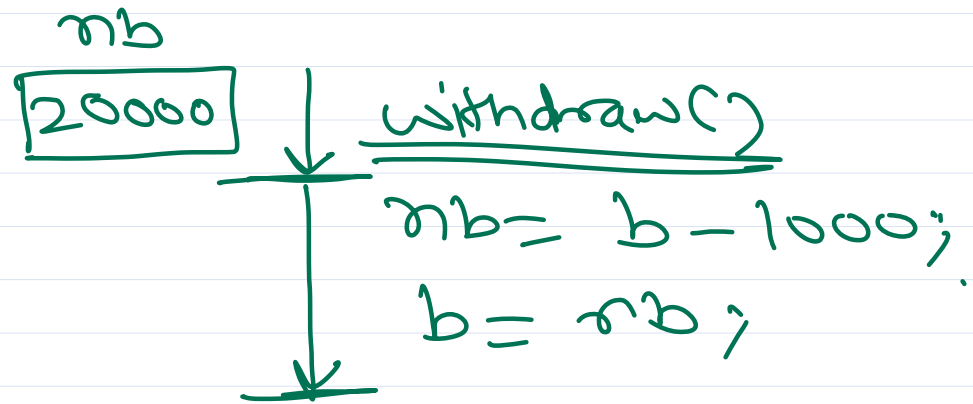
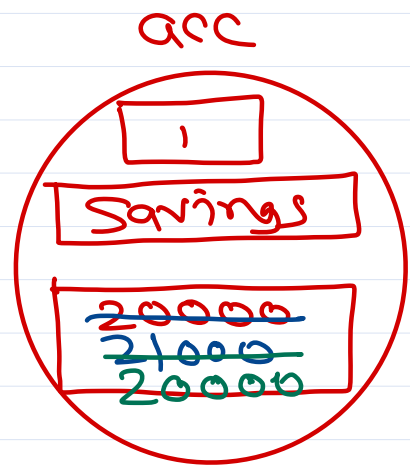
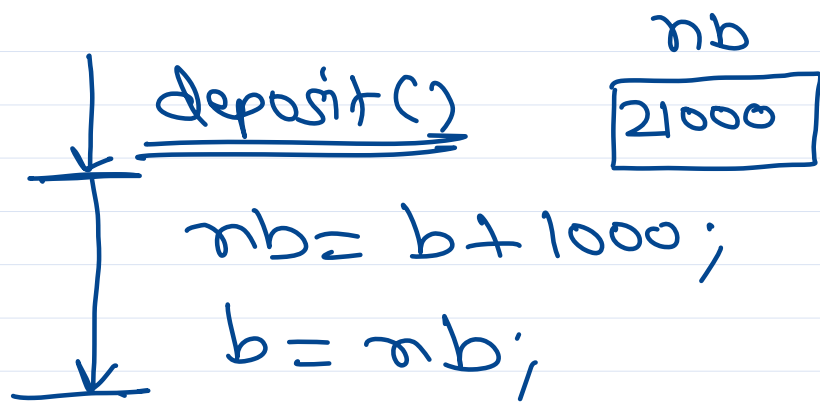
a.deposit(3000);

↓  
new bal = bal + 3000;  
bal = new bal;

a.withdraw(2000);

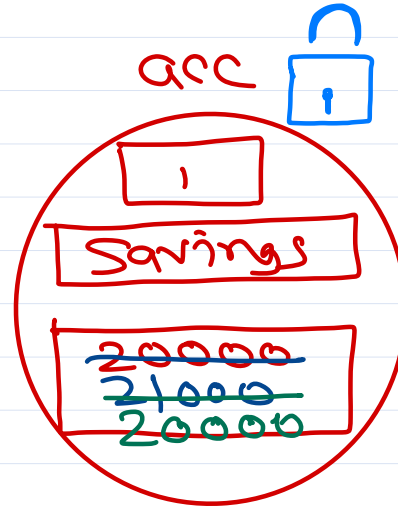
↓  
new bal = bal - 2000;  
bal = new bal;

# Race condition



# Synchronisation

$\downarrow$  deposit() nb  
21000  
 $\downarrow$   $nb = b + 1000;$   
 $\downarrow$   $b = nb;$



nb  
20000  $\times$  withdraw()  
 $\downarrow$   $nb = b - 1000;$   
 $\downarrow$   $b = nb;$

synchronized block

```
synchronized (acc) {
    acc.deposit(-);
}
```

synchronized method

```
class Account {
    ...
    public synchronized void deposit(-) {
    }
}
```

synchronized block

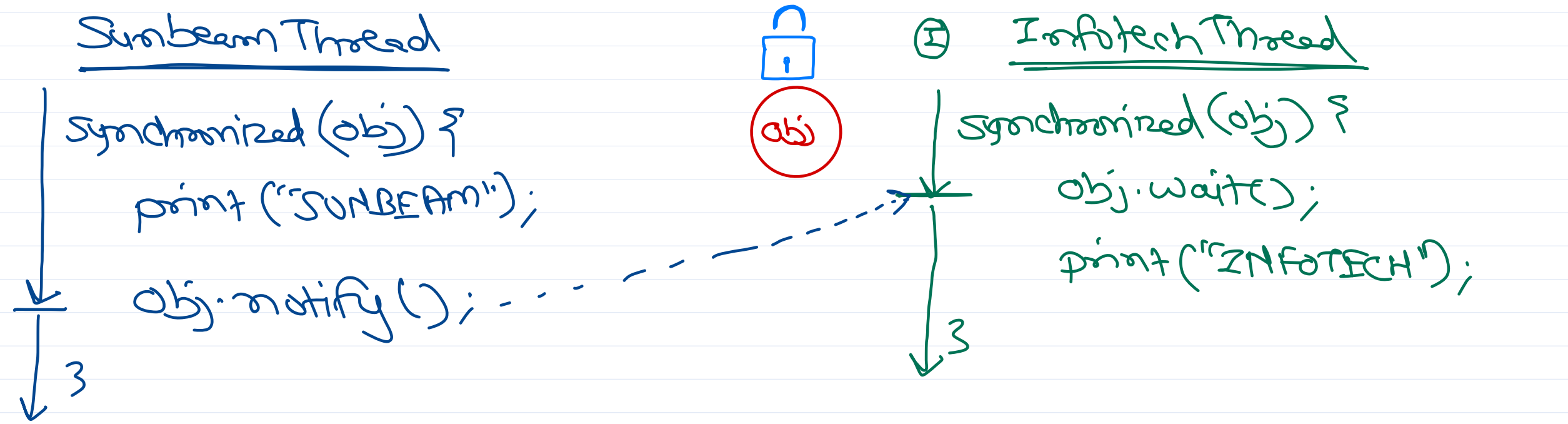
```
synchronized (acc) {
    acc.withdraw(-);
}
```

synchronized method

```
class Account {
    ...
    public synchronized void withdraw(-) {
    }
}
```



# Inter- thread communication





Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

