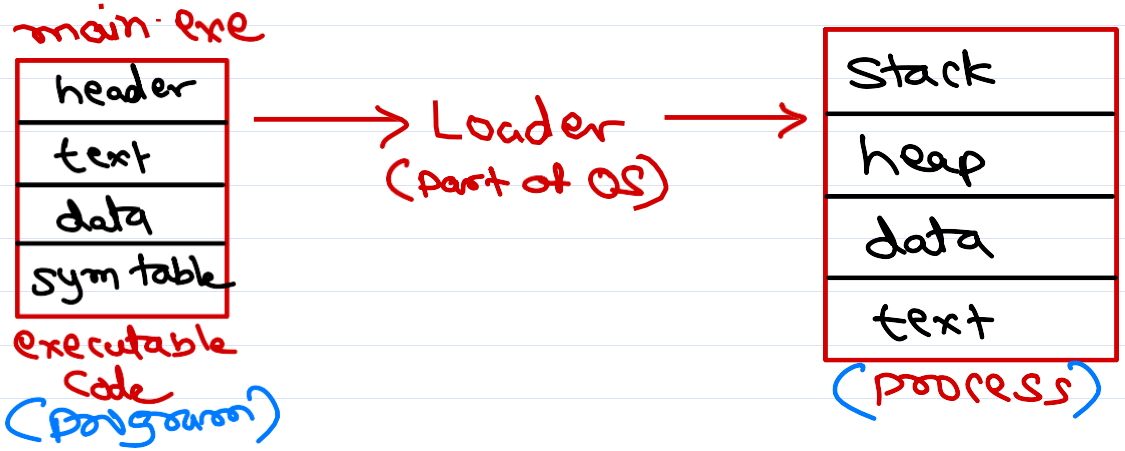
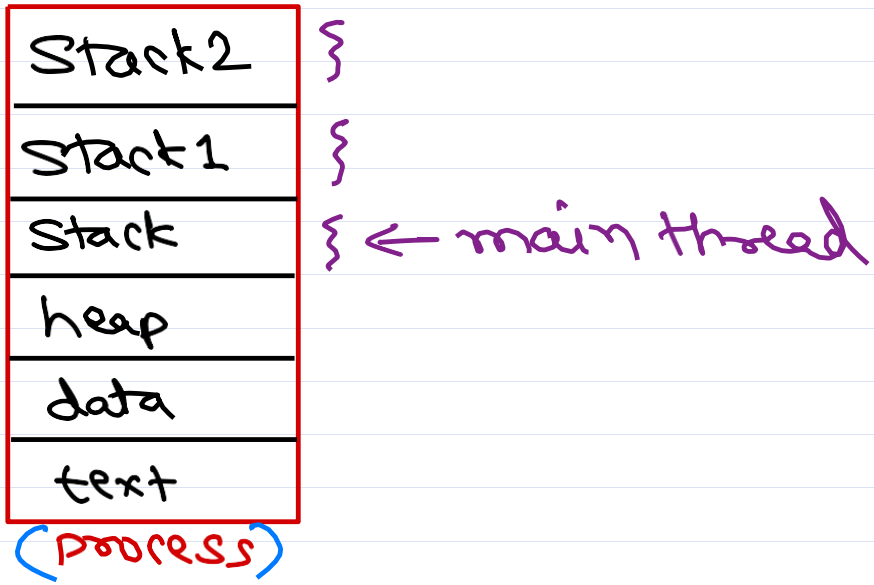
Process vs Thread



OS keeping info about process execution. → PCB



PCB

TCB TCB

process is program under execution.

Thread is a light-weight process.
Threads are used to execute multiple tasks concurrently within a process.

Process is like a container that holds resources required for execution;
while thread is unit of execution/scheduling.

Thread creation

extends Thread

```
class MyThread extends Thread {
    :
    @Override
    void run() {
        //code
    }
}

in main(),
    MyThread t1 = new MyThread();
    t1.start();
```

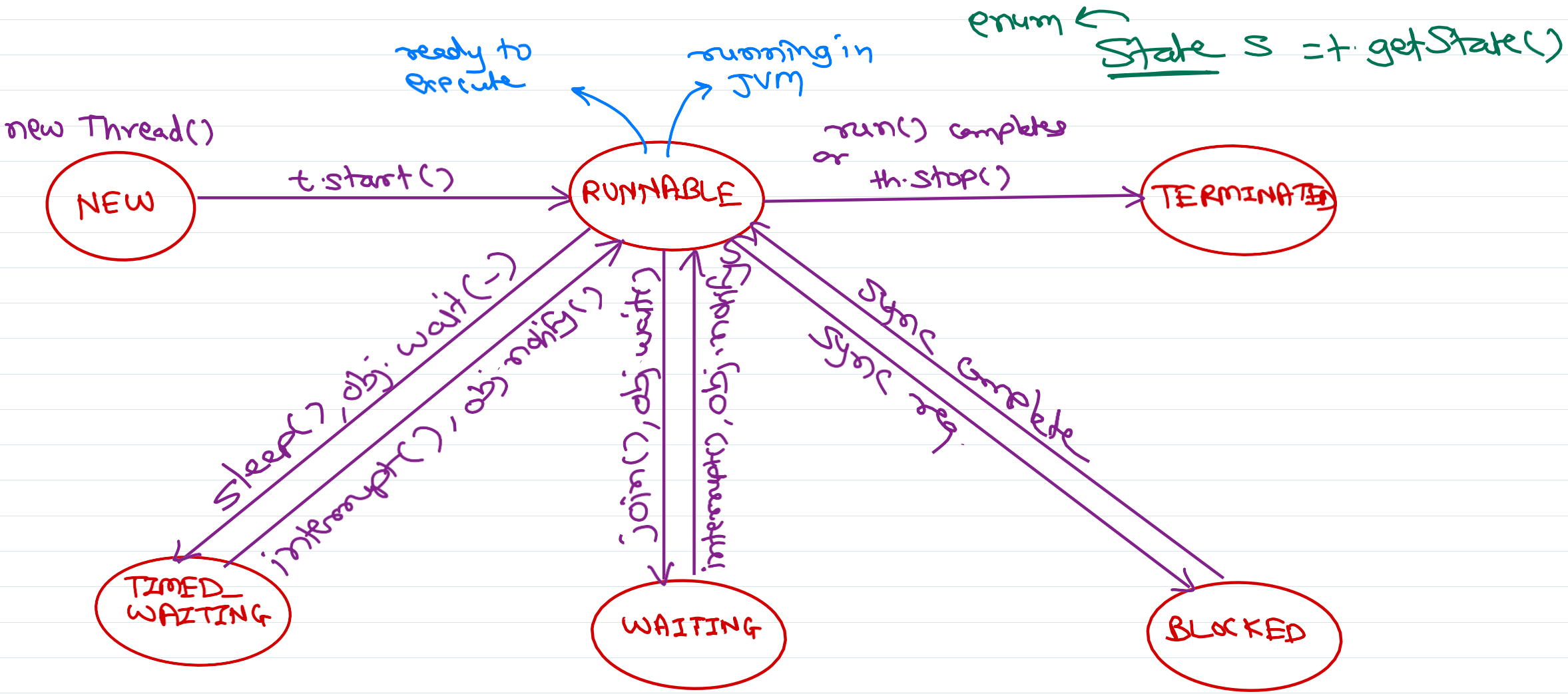
If a java class is already inherited from a class, it cannot be inherited from Thread class (multiple class inherit not allowed)

implements Runnable

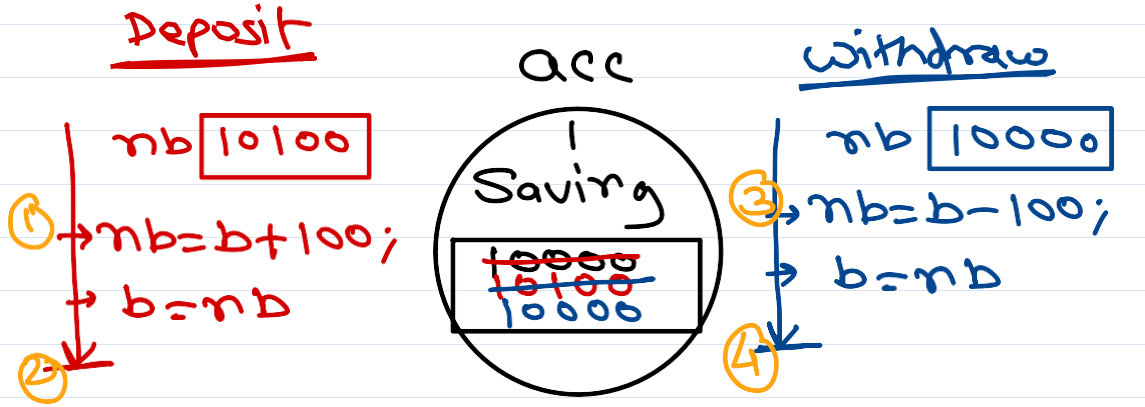
```
class MyRunnable implements Runnable {
    @Override
    void run() {
        //code
    }
}

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

Even if class is inherited from other class, implementing Runnable is possible. Since Runnable is functional interface, we can use lambda expression.

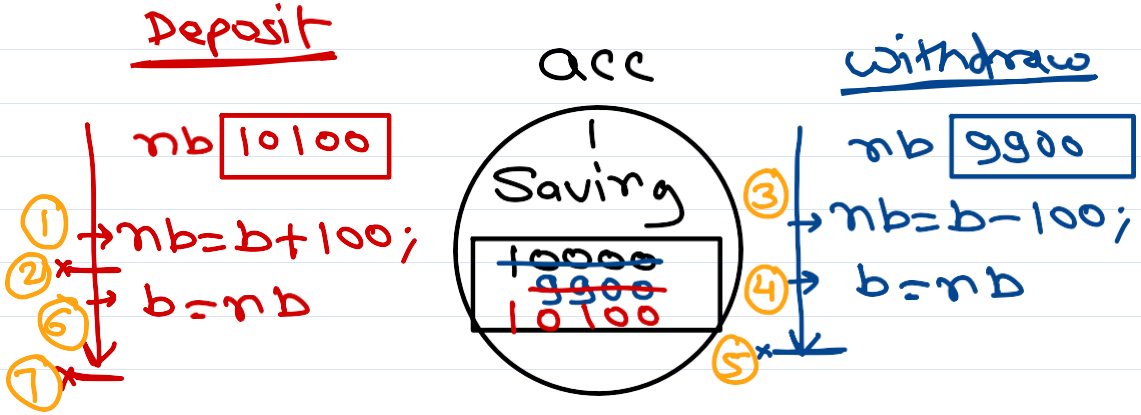
Thread life cycle



Race condition



When multiple threads access same resource at the same time, then race condn occurs → may lead to unexpected/undesired results.



Synchronization

We need to implement synchronization to avoid race condition.

In Java, each object is associated with a sync object called as "monitor". Monitor has two states i.e. locked (busy) and unlocked (available).

Only one thread can lock the monitor and only that thread can unlock it later.

Java has synchronized keyword to lock/unlock the monitor - ① synchronized block ② synchronized method.