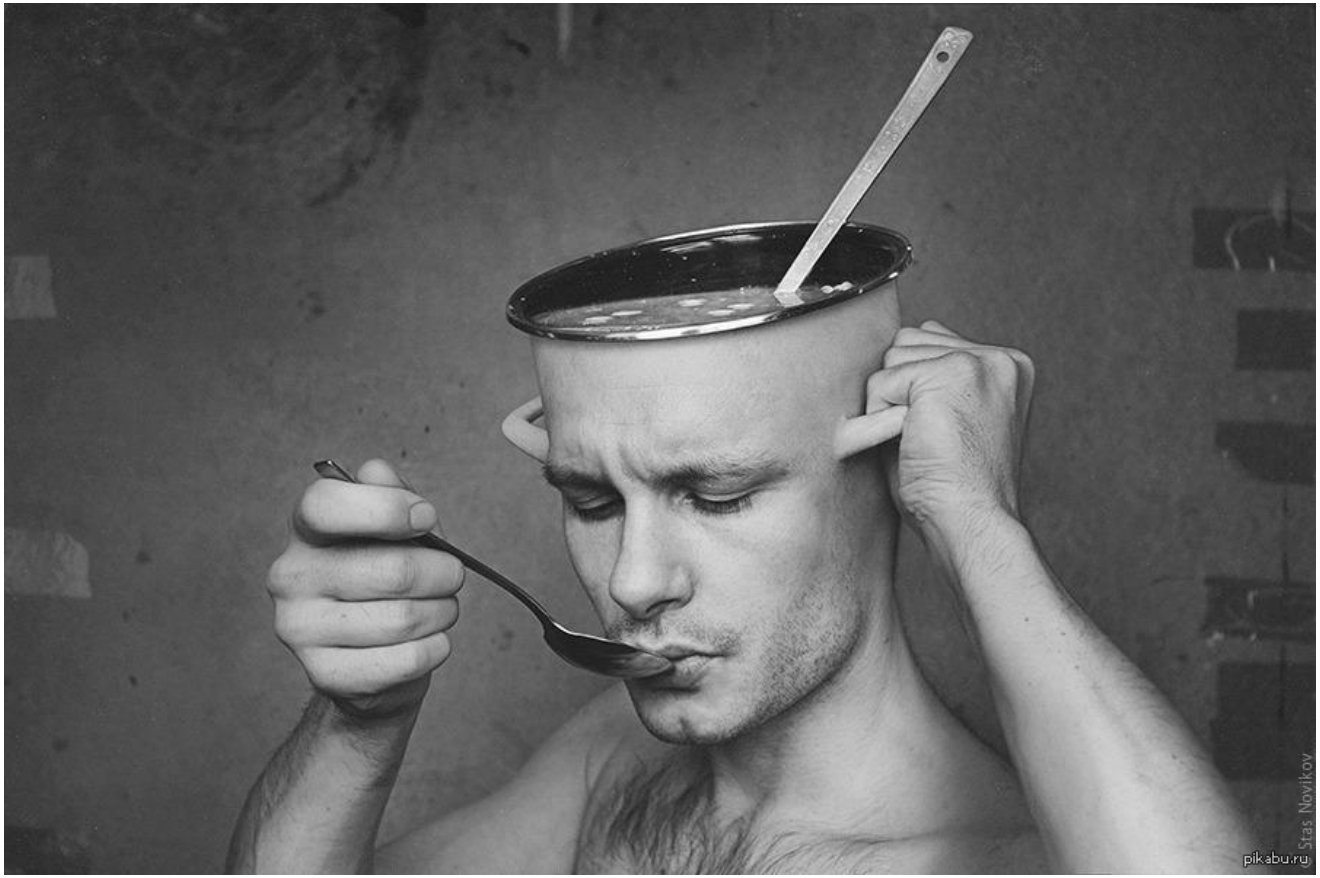
Понимание базовой многопоточности



Обо мне

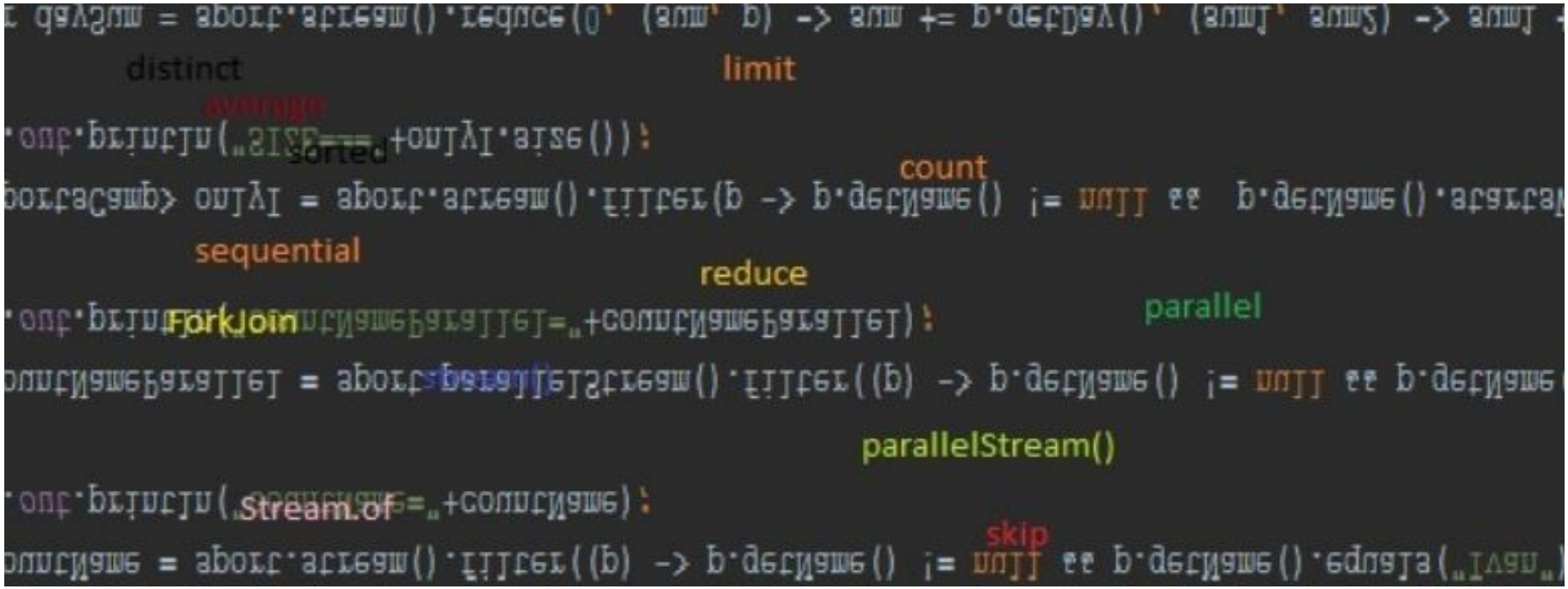
Artem Larin

Senior Java Developer at

Когда я читаю многопоточный код…

```
public synchronized int getSyncA() {
…
Thread t1 = new Thread() {
   public void run() {
      synchronized (one) {
         try {
            Thread.sleep(1000);
         } catch (InterruptedException e) {
…
t1.start();
Thread.sleep(200);
t2.start();
```

Я вижу это…


distinct
limit
sorted
count
sequential
reduce
parallel
ForkJoin
parallelStream()
Stream.of
skip

Когда я думаю о многопоточном коде…

Где же «секретное оружие»?

Бесконечный список книг?

Сакральное знание?

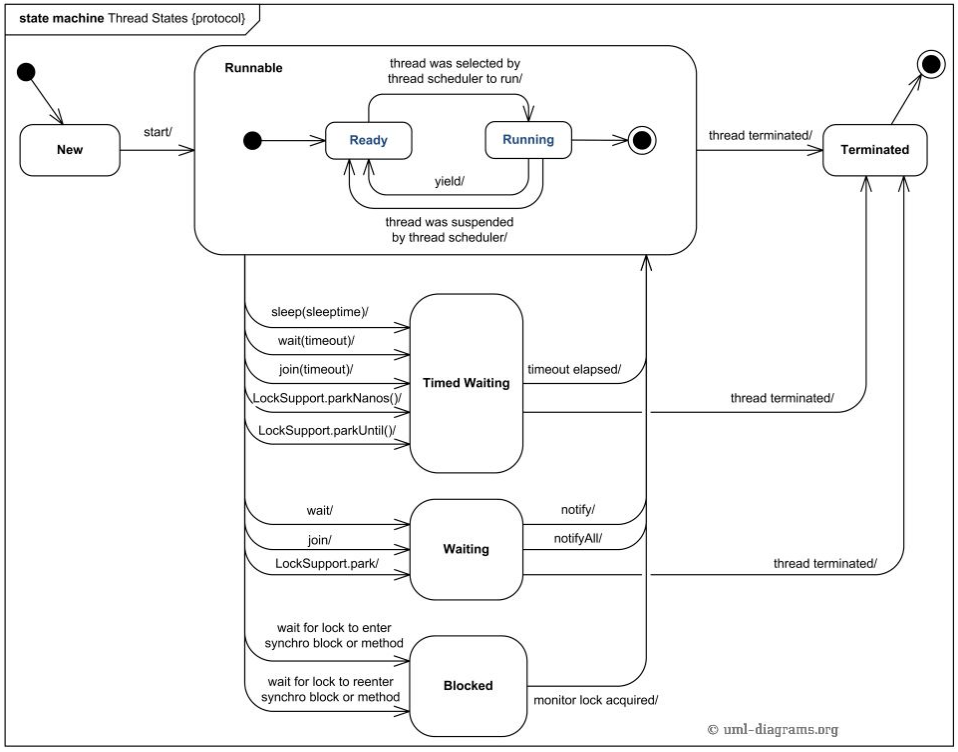
Ключи к пониманию базовой многопоточности

Секретное оружие №1

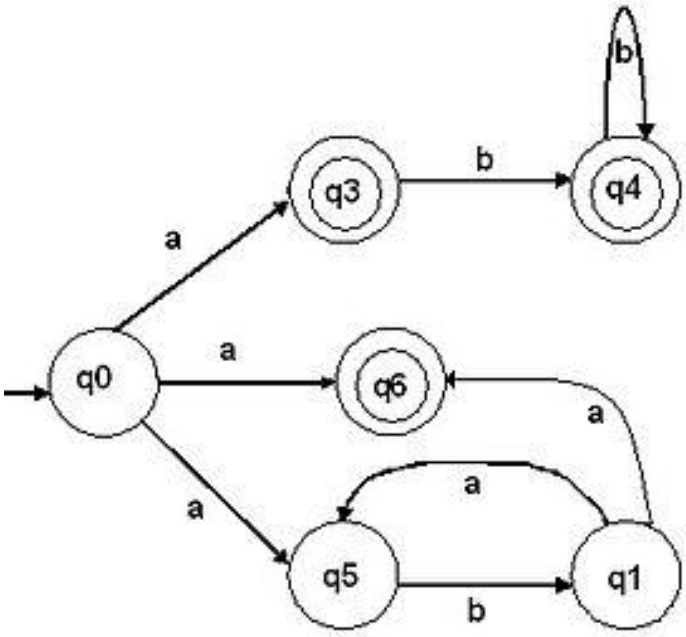
Секретное оружие №2

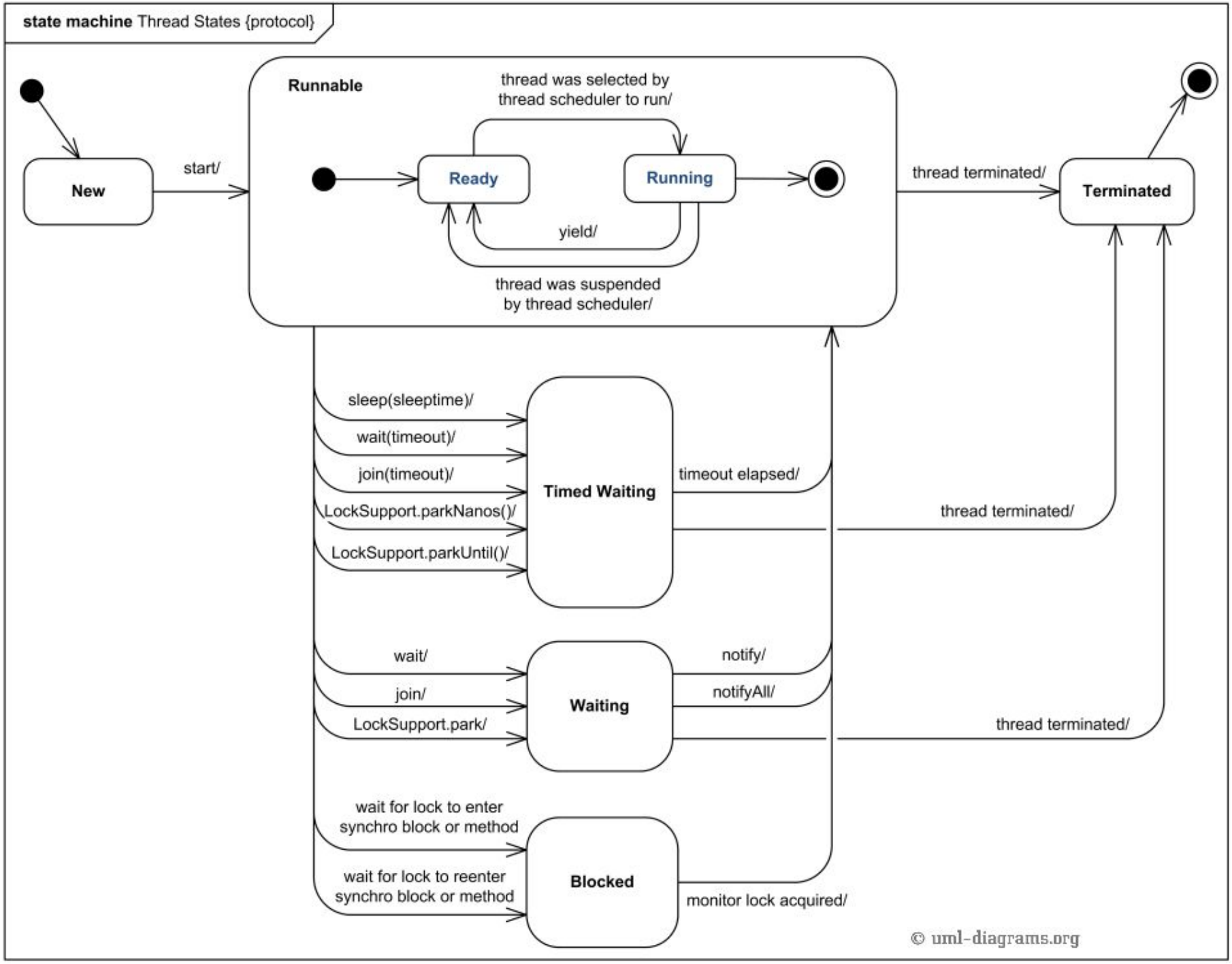
Секретное оружие №1

- Это знание машины состояний потока



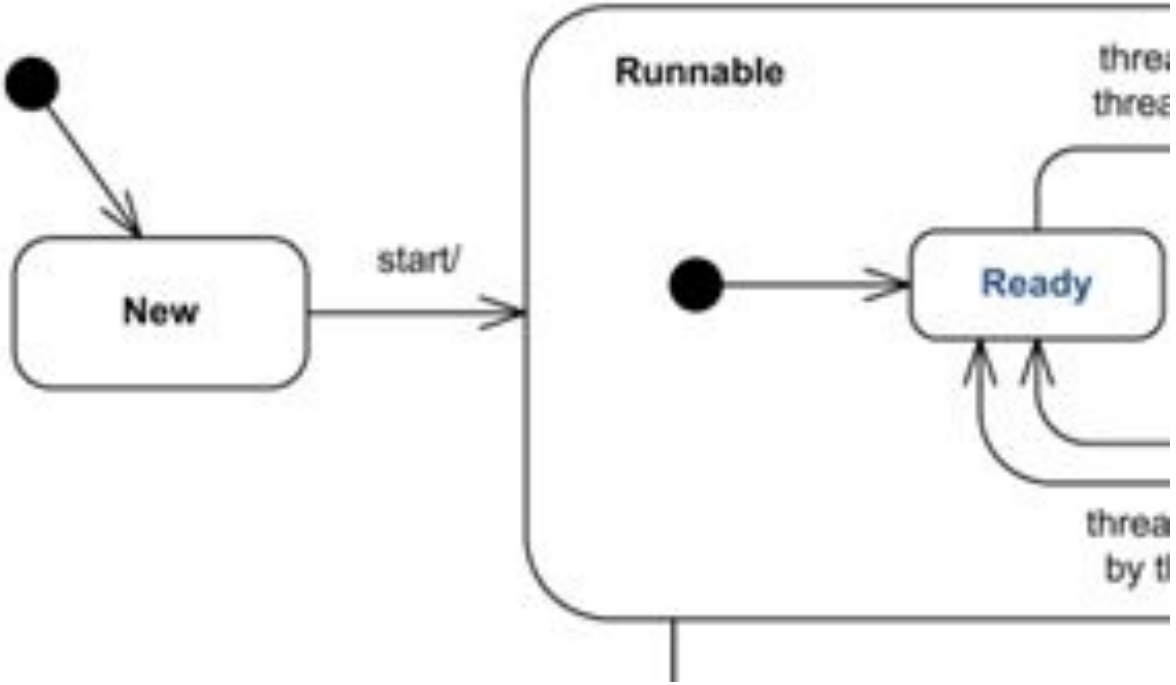
- Коне́чный автома́т — абстрактный автомат, число возможных внутренних состояний которого **конечно (!)**.




state machine Thread States {protocol}
Runnable
thread was selected by
thread scheduler to run/
New
start/
Ready
Running
yield/
thread was suspended
by thread scheduler/
thread terminated/
Terminated
sleep(sleeptime)/
wait(timeout)/
join(timeout)/
LockSupport.parkNanos()/
LockSupport.parkUntil()/
Timed Waiting
timeout elapsed/
thread terminated/
wait/
join/
LockSupport.park/
Waiting
notify/
notifyAll/
thread terminated/
wait for lock to enter
synchro block or method
wait for lock to reenter
synchro block or method
Blocked
monitor lock acquired/
© uml-diagrams.org

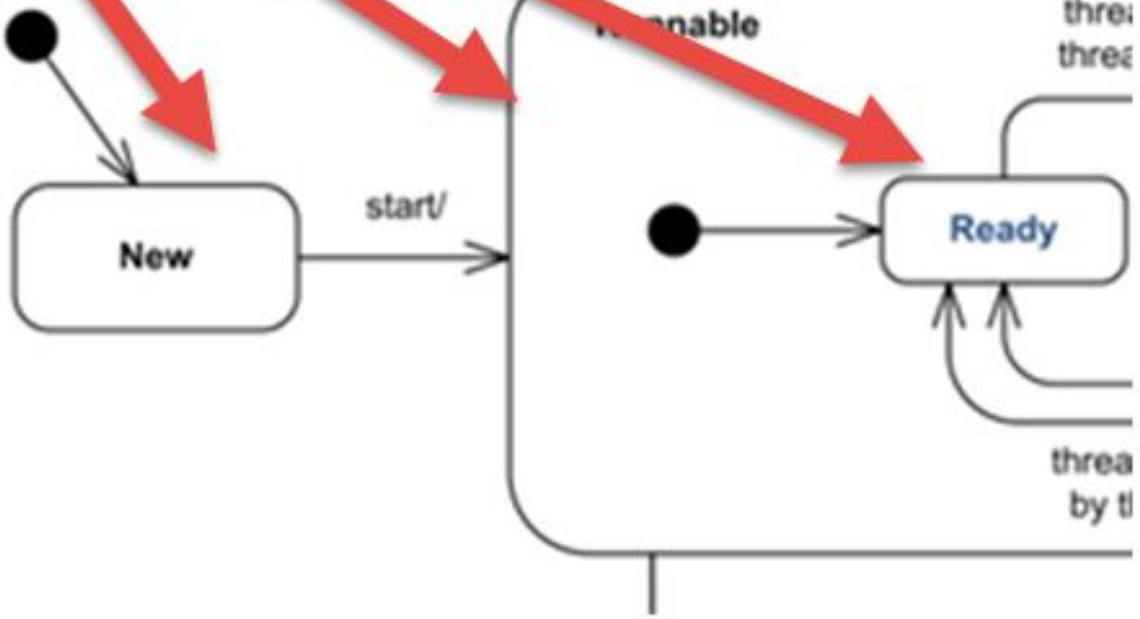
Машина состояний потока

- Состояния
- Переходы
- События



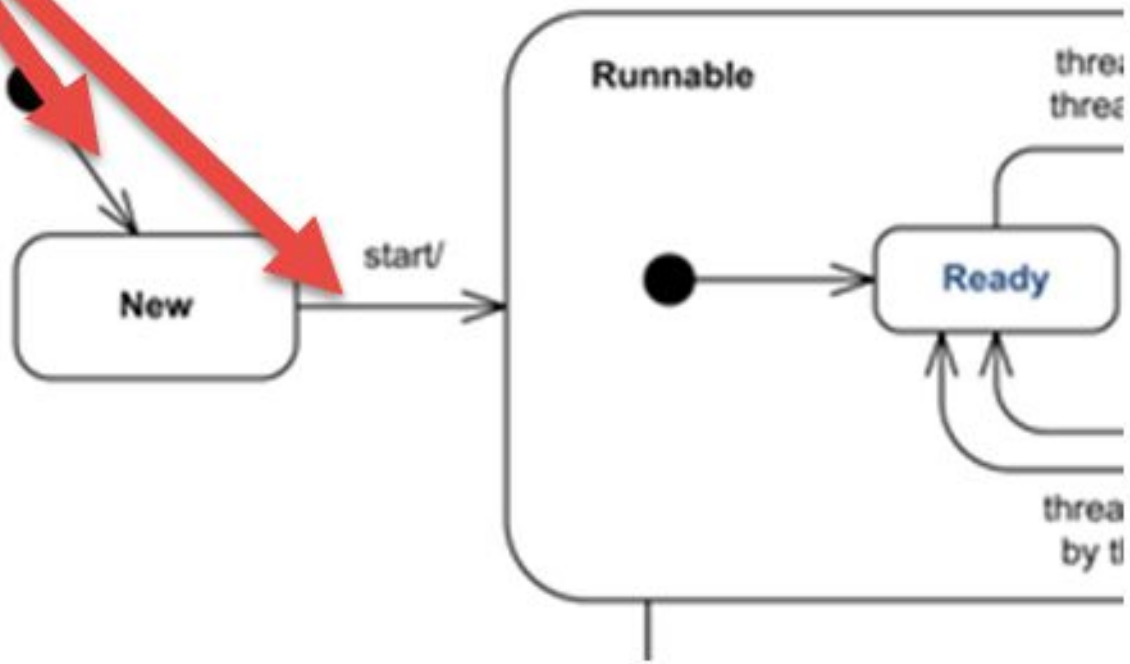
Машина состояний потока

- Состояния
- Переходы
- События



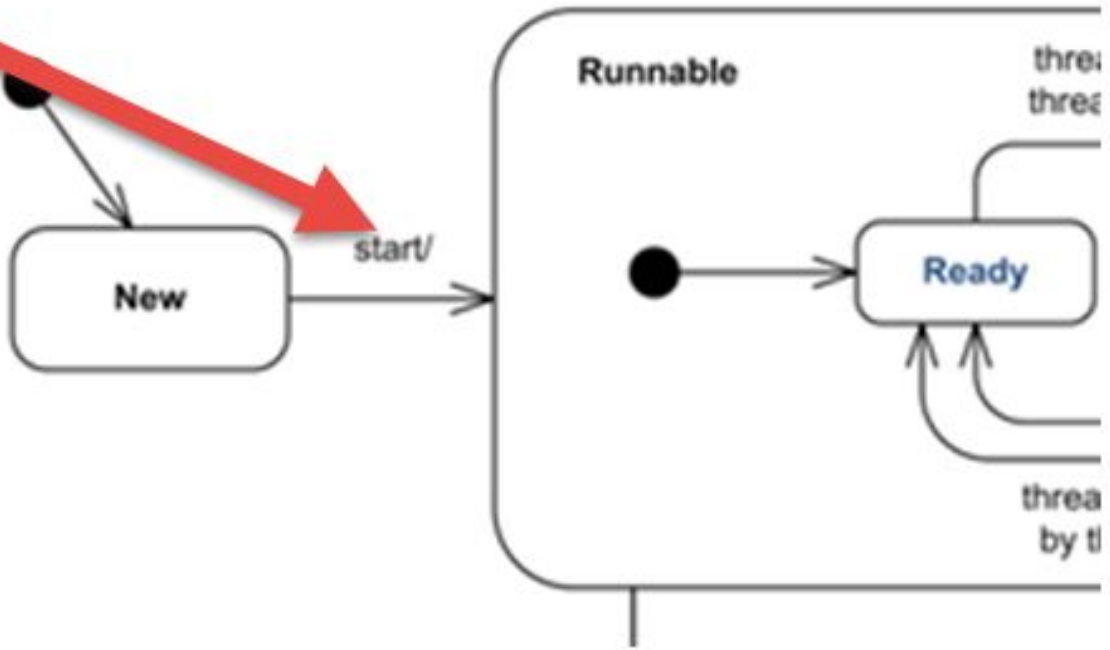
Машина состояний потока

- Состояния
- Переходы
- События

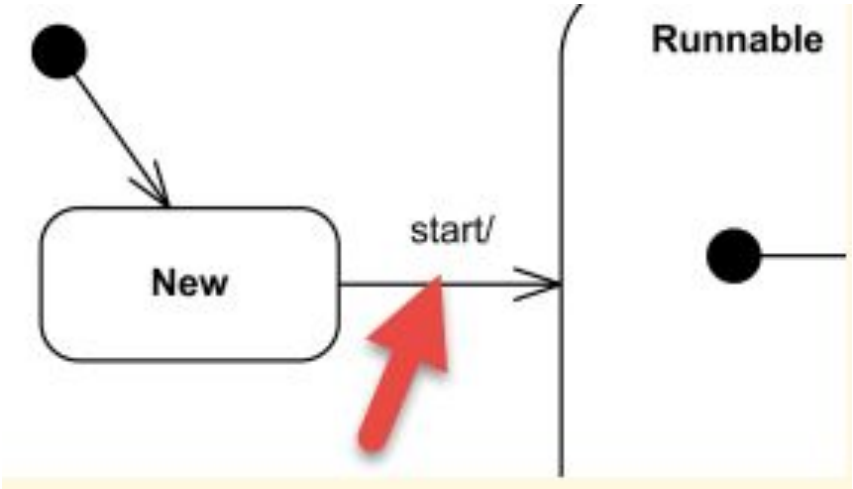


Машина состояний потока

- Состояния
- Переходы
- События



Как читать диаграмму?
Правило №1

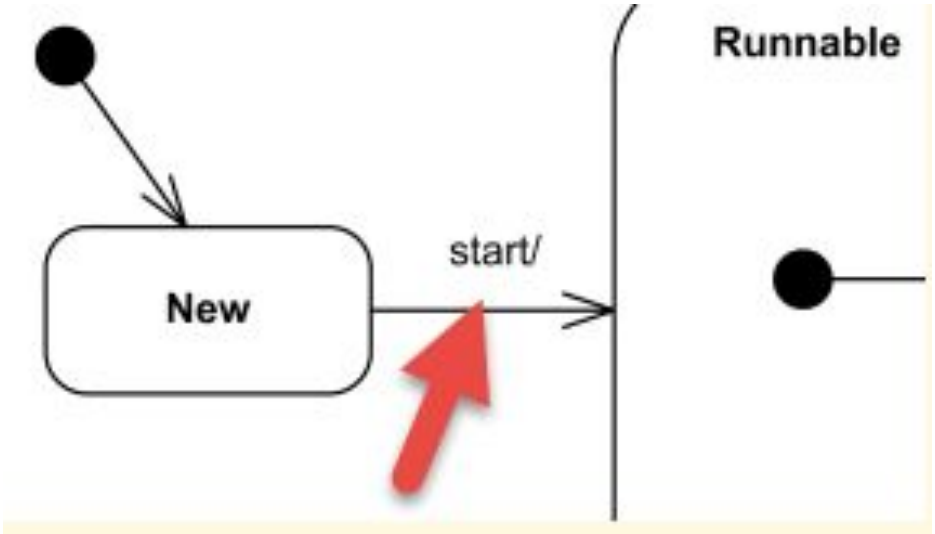


Надпись над стрелкой –
это вызов метода на объекте потока

Например

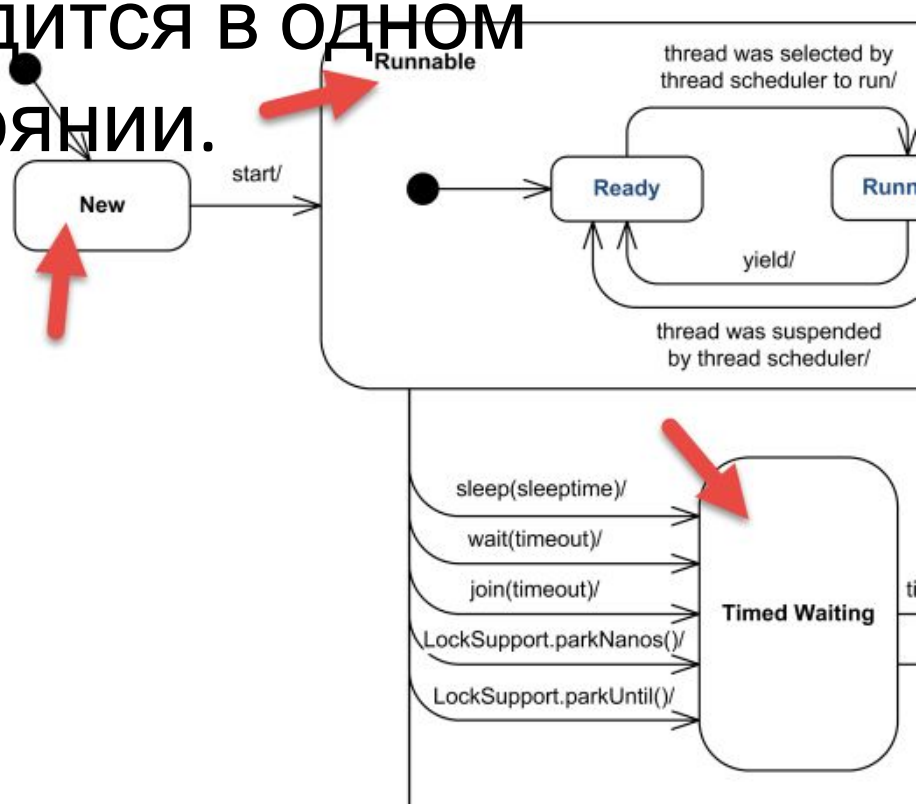
Thread t1 = new Thread() {…}

t1.start();



Как читать диаграмму?
Правило №2

В один момент времени поток находится в одном состоянии.



Как читать диаграмму?
Правило №3

Поток не имеет других состояний и переходов.

Секретное оружие №2

- Это переход в **<u>другое измерение времени</u>**

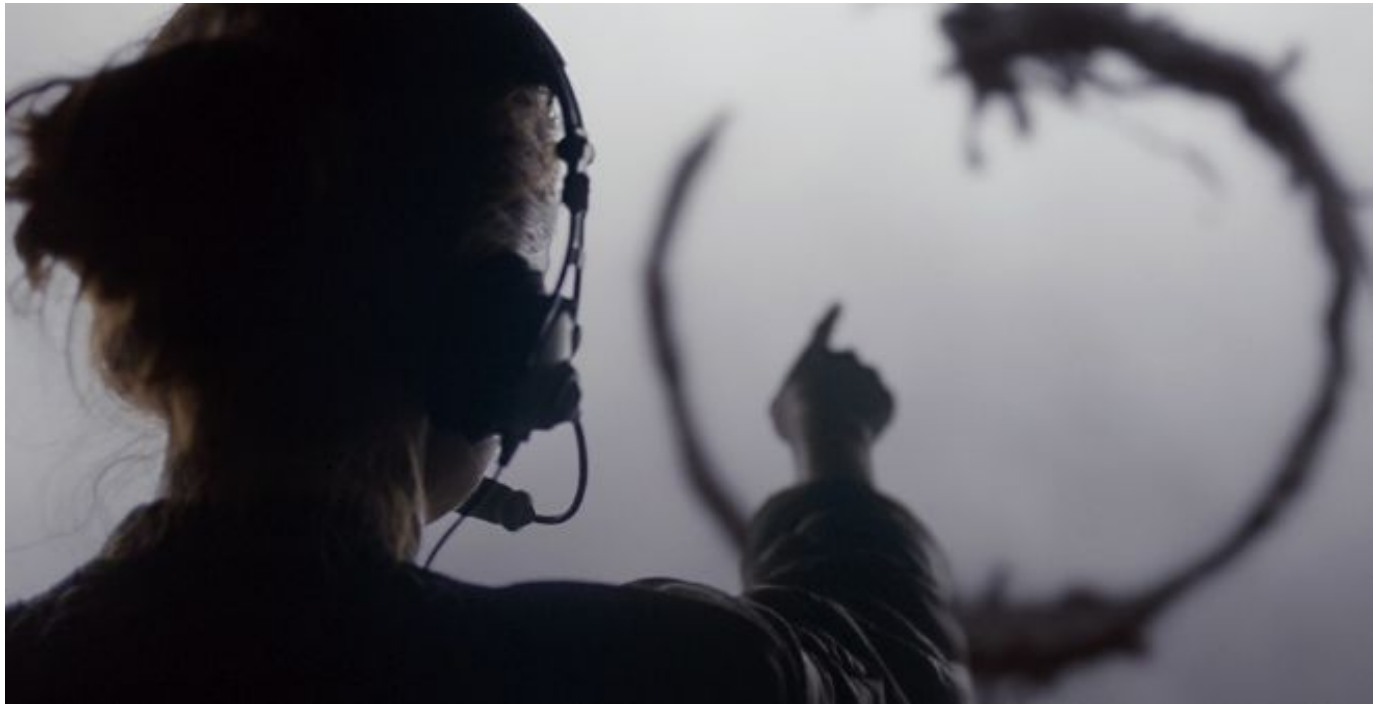
(как в фильме «Прибытие»)

Мы пытаемся понять код в **чужом** временном измерении

```
t.start();
try {
    Thread.sleep(100);
} catch (InterruptedException e) {
    System.out.println("Interrupted");
}
```

А нужно расписать переходы потоков
на псевдоязыке
<u>последовательно во времени</u>,
как видят время обычные люди

- **-> 37 m:R**
- **38 -> t:N**
- **40 -> t:R**



Что за «псевдоязык»?

- **-> 37 m:R** (эта запись означает, что при переходе управления на строку 37 главный поток main (сокращенно m), переходит в состояние **R**unnable)
  38 -> t:N (при передаче управления с 38 строки поток t переходит в состояние **N**ew)
  40 -> t:R (при передаче управления с 40 строки поток t находится в состоянии **R**unnable, потому что был вызван метод t.start())

Что за «псевдоязык»?

- **-> 37 m get(obj1)** (при переходе управления на строку 37 главный поток main захватил монитор объекта obj1)

```
37: synchronized (obj1) {
....
}
```

Что за «псевдоязык»?

- **39 -> m rel(obj1)** (при переходе управления со строки 39 главный поток main отпустил монитор объекта obj1)

synchronized (obj1) {

….

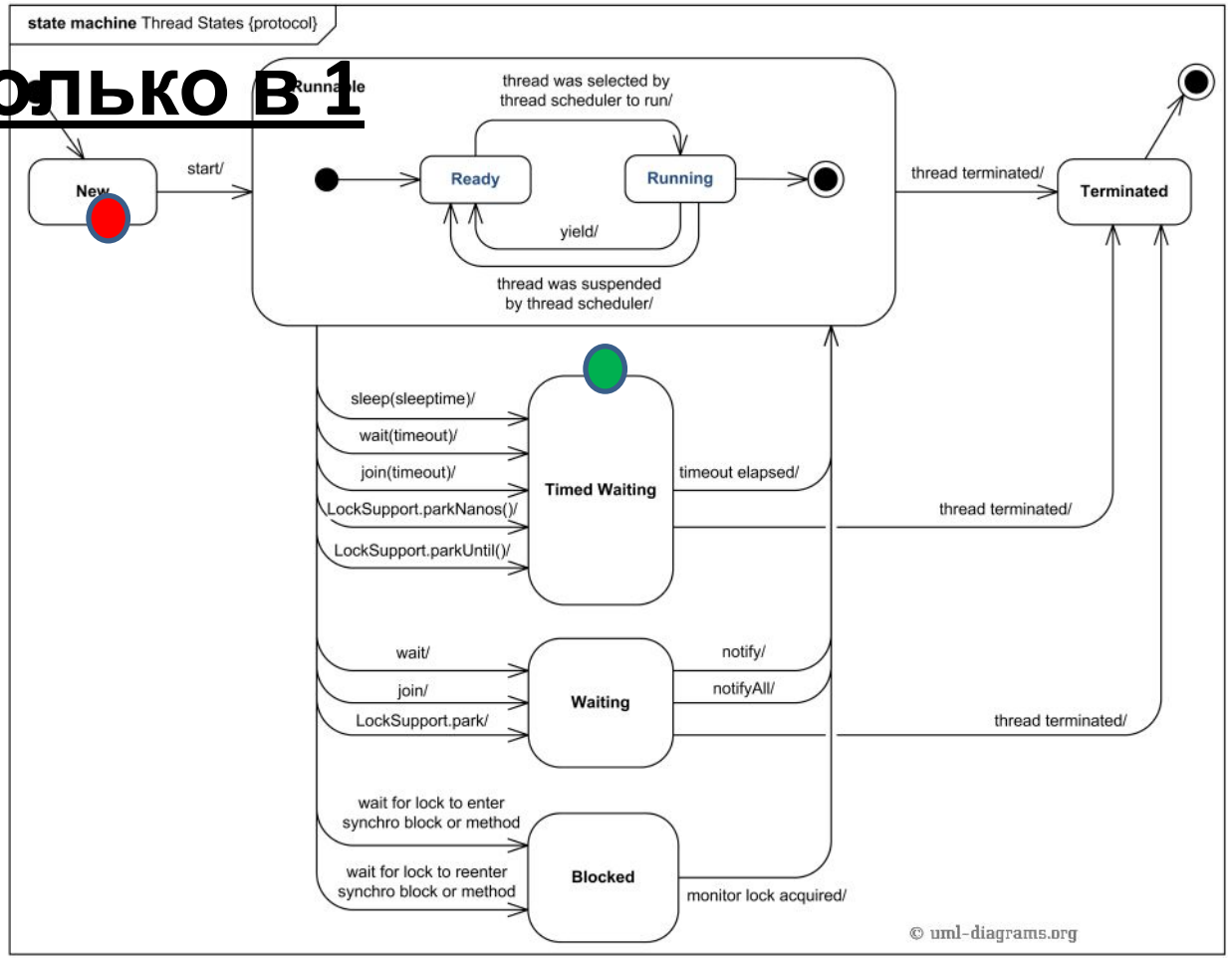
39: }

При этом не забываем правила!

<u>В один момент времени поток находится только в 1 состоянии!</u>

«Фишка» потока 1

«Фишка» потока 2



- Не нужно описывать каждую строчку кода. Указывайте только те, в которых происходят **изменения состояния** какого-либо потока.

Закрепим навыки

```
2   class SharedObj {
3       public int a = 10;
4       public synchronized int getSyncA() { return a; }
5   }
6   public class Demo1 {
7       public final static SharedObj one = new SharedObj();
8       public final static SharedObj two = new SharedObj();
9       public static void main(String s[]) throws InterruptedException {
10          Thread t1 = new Thread() {
11              public void run() {
12                  synchronized (one) {
13                      try { Thread.sleep(1000); }
14                      catch (InterruptedException e) { }
15                      synchronized (two) {
16                          System.out.println("Success!");
17                      } } } };
18          Thread t2 = new Thread() {
19              public void run() {
20                  synchronized (two) {
21                      try { Thread.sleep(1000); }
22                      catch (InterruptedException e) { }
23                      System.out.println(one.getSyncA());
24                  } } };
25          t1.start();
26          Thread.sleep(200);
27          t2.start(); }
28  }
```

Другие способы определения дедлоков

- JDK: jps/jstack
- OpenJDK: JCStress (Алексей Шипилев)

Другие кейсы

- join/sleep
- wait/notifyAll