Concurrency

thread synchronization

Problem

Problem code

```
public class CommonResource {
    public int x = 0;
}
```

Problem code

```java
public class CountThread implements Runnable {
    private final CommonResource res;

    public CountThread(CommonResource res) {
        this.res = res;
    }

    public void run() {
        res.x = 1;
        for (int i = 1; i <= 4; i++) {
            System.out.printf("%s %d \n", Thread.currentThread().get
            res.x++;
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Problem code

```
public class Program {
    public static void main(String[] args) {
        CommonResource commonResource = new CommonResource();
        for (int i = 1; i <= 5; i++) {
            Thread t = new Thread(new CountThread(commonResource));
            t.setName("Thread " + i);
            t.start();
        }
    }
}
```

Thread Synchronization

Types of thread synchronization

- Mutual Exclusive (взаимное исключение)
  - synchronized method
  - synchronized block
  - static synchronization
- Cooperation (Inter-thread communication in java) (кооперация)

Operator `synchronized`

synchronized block

```java
class CountThread implements Runnable {
    private final CommonResource res;

    public CountThread(CommonResource res) {
        this.res = res;
    }

    public void run() {
        synchronized (res) {
            res.x = 1;
            for (int i = 1; i <= 4; i++) {
                System.out.printf("%s %d \n", Thread.currentThread()
                res.x++;
                try {
                    Thread.sleep(100);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
```

synchronized method

```java
class CommonResource {
    private int x;

    synchronized void increment() {
        x = 1;
        for (int i = 1; i <= 4; i++) {
            System.out.printf("%s %d \n", Thread.currentThread().getNa
            x++;
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

synchronized method

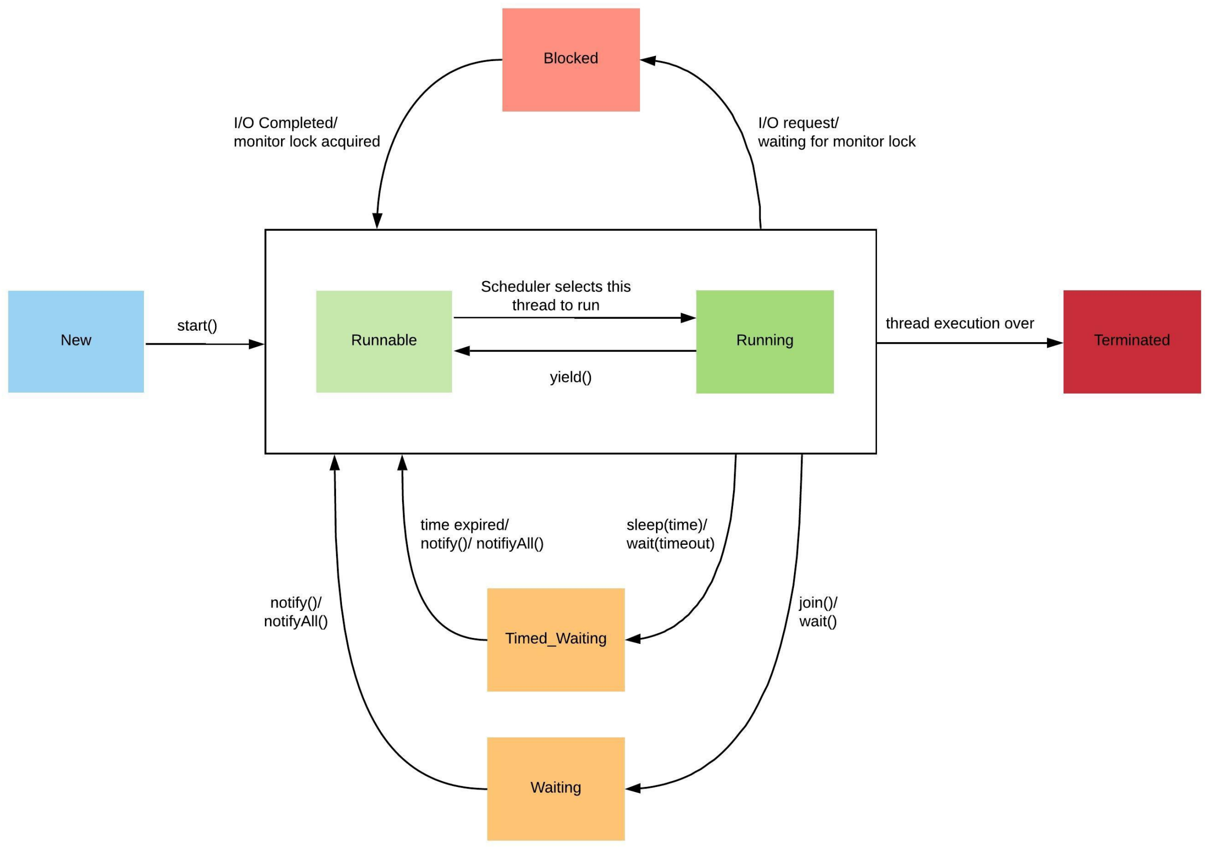
```java
class CountThread implements Runnable {
    private final CommonResource res;

    public CountThread(CommonResource res) {
        this.res = res;
    }

    public void run() {
        res.increment();
    }
}
```

Cooperation

Methods

- `wait()`
- `notify()`
- `notifyAll()`

Thread Lifecycle



Blocked
I/O Completed/ monitor lock acquired
I/O request/ waiting for monitor lock
New
start()
Runnable
Scheduler selects this thread to run
yield()
Running
thread execution over
Terminated
time expired/ notify()/ notifiyAll()
sleep(time)/ wait(timeout)
notify()/ notifyAll()
join()/ wait()
Timed_Waiting
Waiting

Example

```java
// Класс Магазин, хранящий произведенные товары
public class Store {
    private int product = 0;

    public synchronized void get() {
        while (product < 1) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        product--;
        System.out.println("Покупатель купил 1 товар");
        System.out.println("Товаров на складе: " + product);
        notify();
    }

    public synchronized void put() {
        while (product >= 3) {
```

Example

```java
class Producer implements Runnable {
    private Store store;

    public Producer(Store store) {
       this.store = store;
    }

    public void run() {
        for (int i = 1; i <= 5; i++) {
            store.put();
        }
    }
}
```

Example

```java
class Consumer implements Runnable {
    private Store store;

    public Consumer(Store store) {
       this.store = store;
    }

    public void run() {
        for (int i = 1; i <= 5; i++) {
            store.get();
        }
    }
}
```

Example

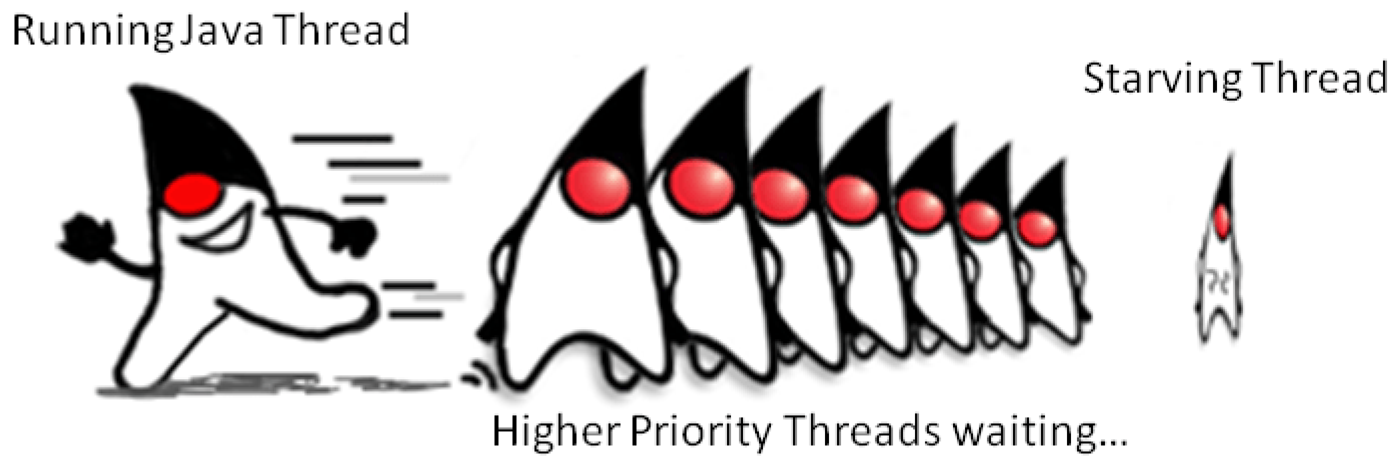
```java
public class Program {
    public static void main(String[] args) {
        Store store=new Store();
        Producer producer = new Producer(store);
        Consumer consumer = new Consumer(store);
        new Thread(producer).start();
        new Thread(consumer).start();
    }
}
```

Typical problems in Java concurrency

- Deadlock (взаимная блокировка)
- Starvation (голодание)
- Nested Monitor Lockout (блокировка вложенного монитора)
- Slipped Conditions (изменчивое условие)

Deadlock

Starvation


Running Java Thread
Starving Thread
Higher Priority Threads waiting...