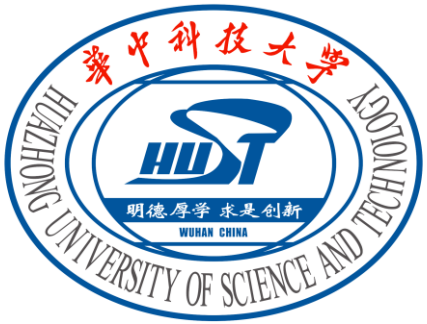
基于Java的面向对象程序设计

陈维亚

weiya_chen@hust.edu.cn

华中科技大学软件学院

第24讲：Java线程简介



目录

1. 线程

❑ 进程与线程

进程 - 运行中的程序

拥有独立的内存空间

进程间可以通信（pipe，socket）

线程 - “轻量级”的进程

线程存在于进程之中

共享进程的资源

每个进程至少拥有一个线程，称为main thread，可以创建出其它线程。

1. 线程

❑ 线程的生命周期

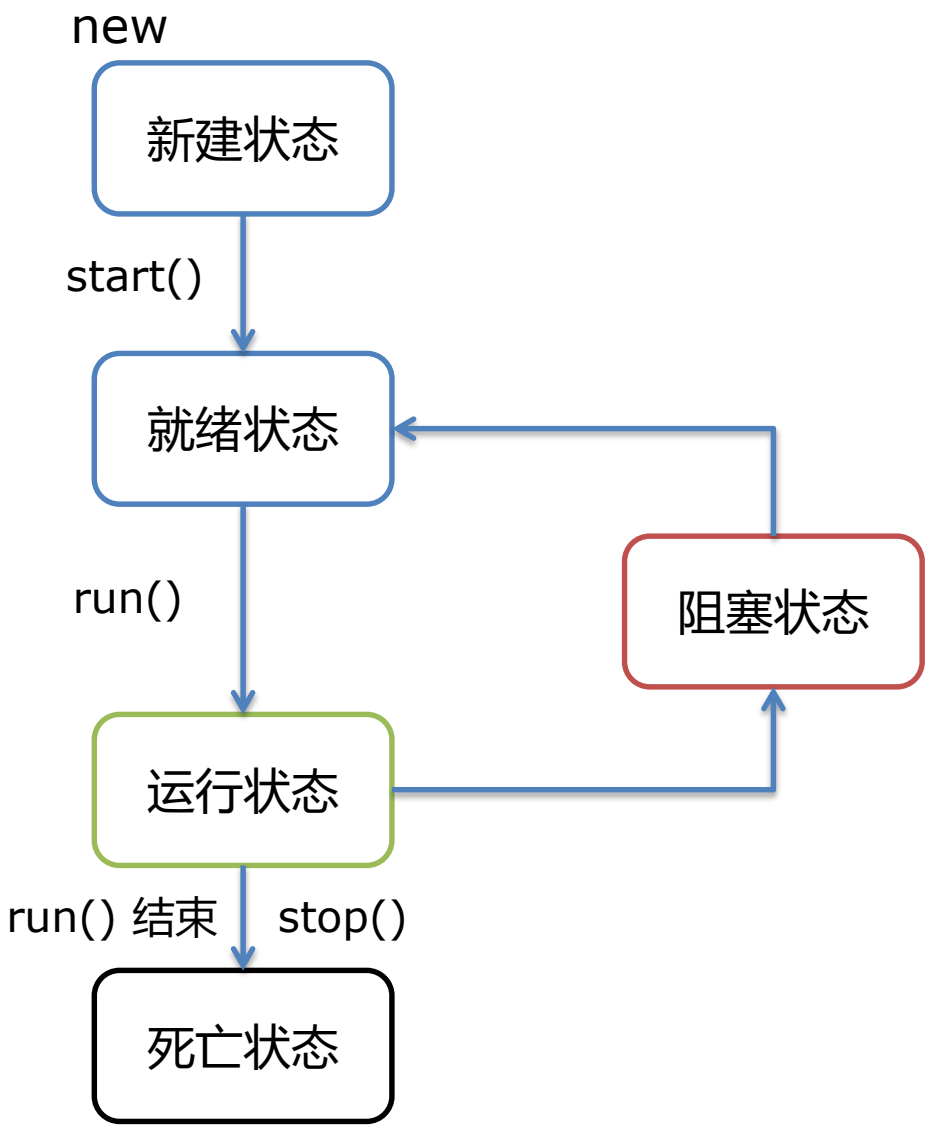
阻塞状态:

一个线程执行了sleep、suspend等方法，失去所占用资源之后，该线程就从运行状态进入阻塞状态：

等待阻塞：运行状态中的线程执行 wait() 方法，使线程进入到等待阻塞状态。

同步阻塞：线程在获取 synchronized 同步锁失败。

其他阻塞：通过调用线程的 sleep() 或 join() ，线程就会进入到阻塞状态。



2. 使用线程

❑ 创建线程

方法一：创建一个实现了Runnable接口的对象，该对象被用作Thread对象的参数。

```
public class HelloRunnable implements Runnable {

    public void run() {
        System.out.println("Hello from a thread!");
    }

    public static void main(String args[]) {
        (new Thread(new HelloRunnable())).start();
    }

}
```

2. 使用线程

❑ 创建线程

方法二： Thread类本身也实现了Runnable接口，我们可以创建Thread类的子类，重写其run方法。

```
public class HelloThread extends Thread {

    public void run() {
        System.out.println("Hello from a thread!");
    }

    public static void main(String args[]) {
        (new HelloThread()).start();
    }

}
```

2. 使用线程

❑ 创建线程

方法一：创建一个实现了Runnable接口的对象，该对象被用作Thread对象的参数。

方法二： Thread类本身也实现了Runnable接口，我们可以创建Thread类的子类，重写其run方法。

2. 使用线程

❑ 阻塞线程

Thread.sleep 可用于暂停执行当前线程一段时间

目的：

- 让出资源给其它线程
- 让当前线程按一定的节奏执行

暂停时间以毫秒或纳秒计，但并不精确

```
public class SleepMessages {
    public static void main(String args[])
        throws InterruptedException {
        String info[] = {
            "Mares eat oats",
            "Does eat oats",
            "Little lambs eat ivy",
            "A kid will eat ivy too"
        };

        for (int i = 0;
             i < importantInfo.length;
             i++) {
            //Pause for 4 seconds
            Thread.sleep(4000);
            //Print a message
            System.out.println(info[i]);
        }
    }
}
```

2. 使用线程

❑ 中断线程

干预线程的执行流程，比如可以让线程提前结束。

```
for (int i = 0; i < importantInfo.length;
i++) {
    // Pause for 4 seconds
    try {
        Thread.sleep(4000);
    } catch (InterruptedException e) {
        // We've been interrupted: no more
messages.
        return;
    }
    // Print a message
    System.out.println(importantInfo[i]);
}
```

❑ 等待线程

join方法使一个线程等待另一个线程执行完成，比如：

t.join()；

表示当前线程必须等 t 执行完毕后才能继续执行。

练习 1

写一个main函数运行如下程序，并将其进行改写为实现Runnable接口。

```
public class GuessANumber extends Thread {
    private int number;
    public GuessANumber(int number) {
        this.number = number;
    }

    public void run() {
        int counter = 0;
        int guess = 0;
        do {
            guess = (int) (Math.random() * 100 + 1);
            System.out.println(this.getName() + " guesses "
+ guess);
            counter++;
        } while(guess != number);
        System.out.println("** Correct!" + this.getName() +
"in" + counter + "guesses.**");
    }
}
```

3. 线程的同步

不同的线程可以靠共享对象的引用实现交流，这种交流方式高效，但是会带来同步的问题。

```
public class SynchronizedCounter {
    private int c = 0;

    public synchronized void increment() {
        c++;
    }

    public synchronized void decrement() {
        c--;
    }
}
```

```
public void addName(String name) {
    synchronized(this) {
        lastName = name;
        nameCount++;
    }
    nameList.add(name);
}
```

总结

线程的定义

线程的生命周期

简单线程的使用

下节预告

Java 图形编程