学习任务二 开发小游戏

子任务 2.4 实现线程

1、 任务引入：

打字母游戏中，26 个字母从天而降，各自都是一个独立的线程，所以我们要学会定义线程

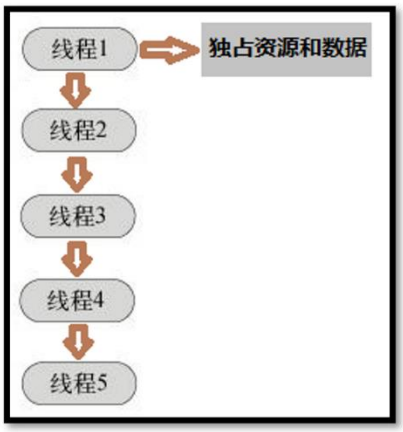
2、 相关知识链接-理解线程的概念

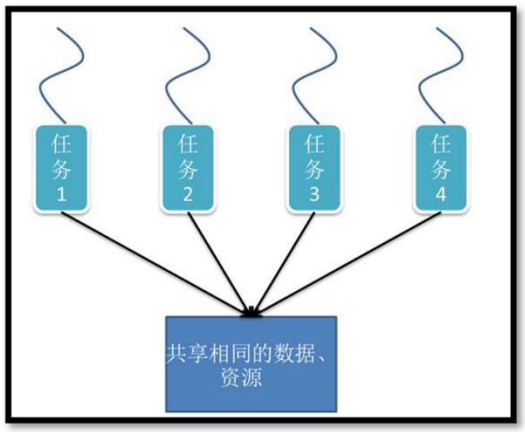
1.1 理解多线程的工作方式

Java 给多线程编程提供了内置的支持。 一条线程指的是进程中一个单一顺序的控制流，一个进程中可以并发多个线程，每条线程并行执行不同的任务。

多线程是多任务的一种特别的形式，但多线程使用了更小的资源开销。这里定义和线程相关的另一个术语 - 进程：一个进程包括由操作系统分配的内存空间，包含一个或多个线程。一个线程不能独立的存在，它必须是进程的一部分。一个进程一直运行，直到所有的非守护线程都结束运行后才能结束。

多线程能满足程序员编写高效率的程序来达到充分利用 CPU 的目的。例如，我们看到下图所示的程序执行两种方式，显然多线程可以提高工作的效率。

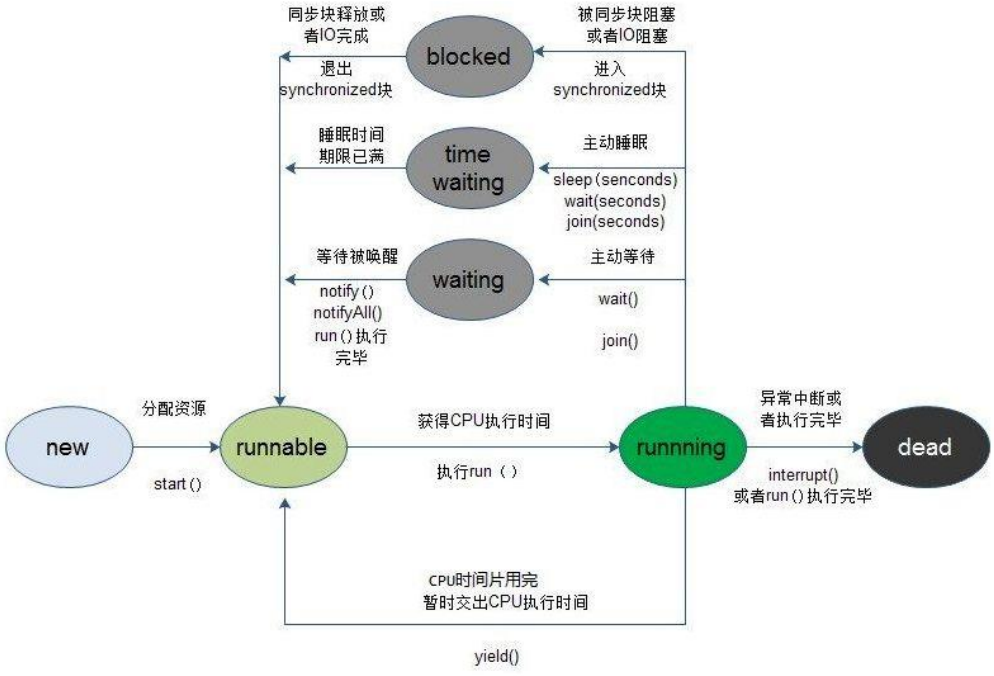




单线程处理事物的方式　　多线程处理事物的方式

1.2 理解线程的生命周期

线程是一个动态执行的过程，它也有一个从产生到死亡的过程。下图显示了一个线程完整的生命周期。



通过这个图形，我们可以看出，线程如果没有竞争对手，会经历从创建，到准备执行，到执行，到死亡的全过程，但是如果遇到如下情况，可能会被暂停执行：

1）如果被要求睡眠一段时间或者等待一段时间，那么这个时候 cpu 会去执行其他准备好执行的线程，等到时间到，当前线程又会处于准备好的状态，等待 cpu 来执行

2）如果遇到堵塞，当前线程也会推出对资源的占用，等待堵塞消除再进入准备运行状态

3）如果需要当前线程等待，当前线程也会推出对资源的占用，等待 notify 方法唤醒他，再进入准备运行状态

看完这个线程的生命周期图，我需要大家回答下面两个问题

1） 线程里面要做的事情应该写在哪个方法里面？

2） 启动线程调用哪个方法？

3、相关知识链接-创建线程（微课学习内容）

1）学会创建线程

创建线程两种方法：继承一个 Thread 父类和实现 Runnable，大家可以试着完成下面两个线程的创建和测试：

```
package thread;
public class myThread1 extends Thread{
    public void run(){
        try {
            for(int i=0;i<6;i++){
                System.out.println("听音乐中"+(i+1));
                Thread.sleep(50);
            }
            System.out.println("听完音乐");
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

```java
package thread;
public class myThread2 implements Runnable{
    public void run(){
        try {
            for(int i=0;i<3;i++){
                System.out.println("写论文中"+(i+1));
                Thread.sleep(100);
            }
            System.out.println("完成论文");
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

```java
package thread;
import java.io.*;
public class myThread {

    public static void main(String args[]){

        Thread t1=new myThread1();
        Thread t2=new Thread(new myThread2());
        t1.start();
        t2.start();

    }
}
```

2）试一试

请大家模仿上面的案例完成一个倒数的线程设计，要求是启动程序，出现一个倒数的初始数字，然后每隔一秒，数字开始逐一减少，直到 0，线程结束。

4、技能训练：

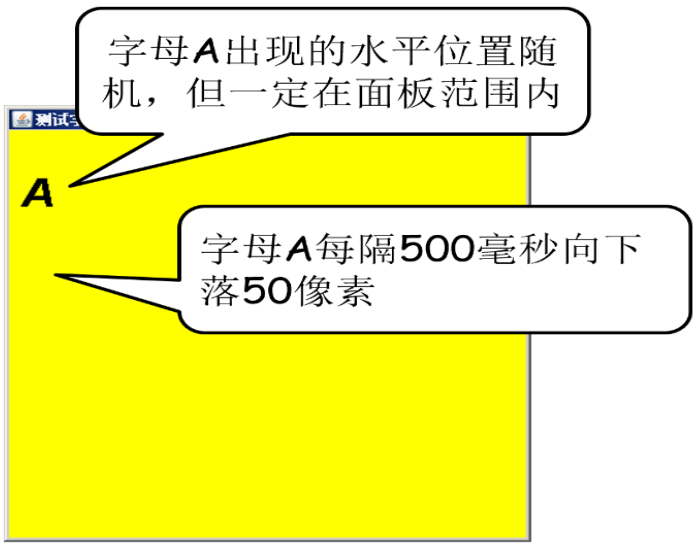
根据上面所学的内容，完成如下训练任务。

1）完成一个图形计时器：综合使用图形绘制和线程设计，完成如下图形计时器。

【任务分析】

- 面面板里画扇形，扇形的角度是一个可以变化的变量
- 板里设计一个线程
- 线程的内容是每隔一段时间修改角度，（角度从 0 开始 变化到 360），然后重画

2）完成游戏界面的一个字母下落



【任务分析】

- 做一个字母面板
- 在一个位置绘制一个字母 A（x=0 到面板的宽度之间一个随机值；y=0）
- 在面板中做一个线程，每隔一段时间，修改字母 A 的 y 坐标，增加 y 坐标的值

5、其他知识链接

1）线程的同步

在一般情况下，创建一个线程是不能提高程序的执行效率的，所以要创建多个线程。但是多个线程同时运行的时候可能调用线程函数，在多个线程同时对同一个内存地址进行写入，由于 CPU 时间调度上的问题，写入数据会被多次的覆盖，所以就要使线程同步。

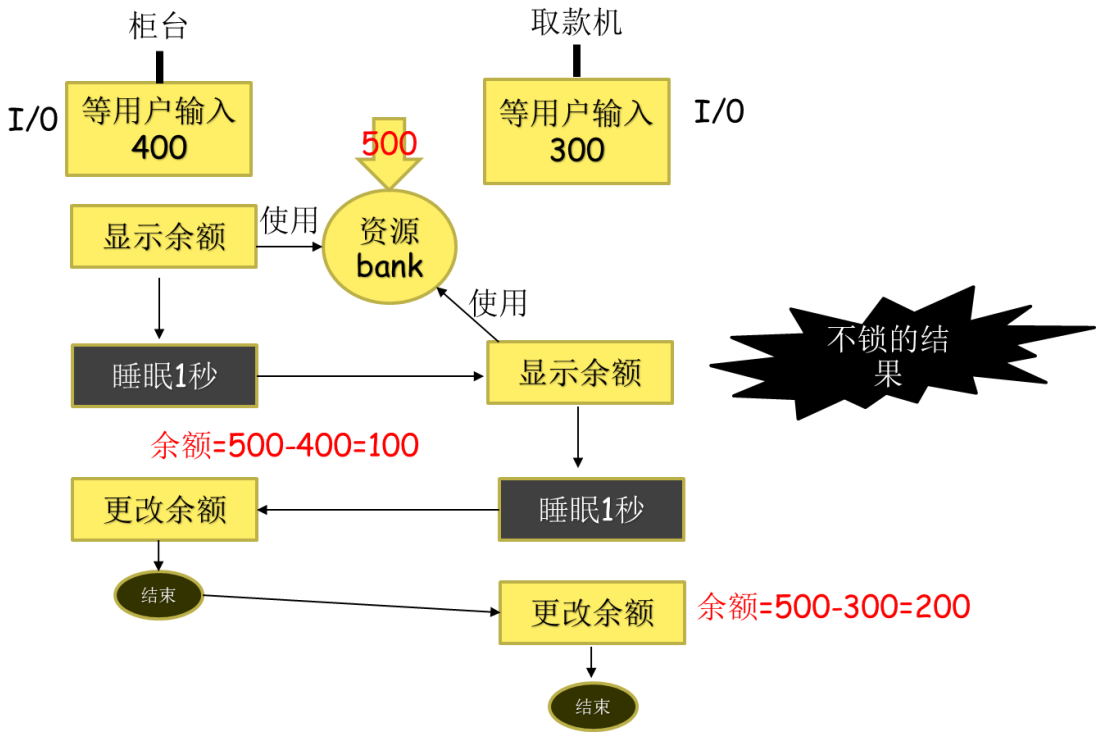
当有一个线程在对内存进行操作时，其他线程都不可以对这个内存地址进行操作，直到该线程完成操作， 其他线程才能对该内存地址进行操作，而其他线程又处于等待状态，实现线程同步的方法有很多。

在 Java 里面，通过 synchronized 进行同步的保证。例如：

```
class MyTest{

private static final Object lock=new Object();

public static synchronized void test(){
//同步的方法
}

public void test2(){
synchronized(lock){
//方法级同步，也可以使用 this 实现对象级同步
}
}

}
```

2）理解 Java 中的“锁”

如果不加锁，我们看到线程对于数据资源的访问会出现错误，如下：



添加“锁”之后,公共资源的数据得到了保护：