**命名方式：**

类名首字母大写。

方法名首字母小写。

驼峰法：myCry 下划线法：my\_cry

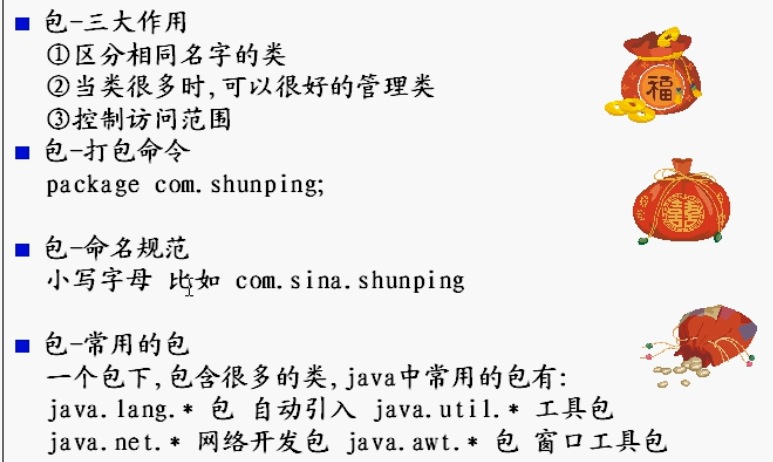
**关于static：**

静态变量，可以被任何一个对象使用。类中的静态变量是可以用 类名.静态变量名 来访问的，也可以通过对象实例访问，但是规定静态变量用 **类名.静态变量名** 来访问.

静态方法，所有的对象共有的方法。**静态方法中不能访问非静态变量。也不能访问非静态方法。**例如在static main方法中不能调用其他非static 方法。

Static {} 静态区域块，在定义类的时候自动执行，不需要事例化类时才创建。

**关于 package:**



**关于继承：**

class 子类 extends 父类，不允许多重继承。

父类的 public 和 protected 可以被继承。

**关于重载(overloading)**

1: 方法名必须相同

**2:方法的参数类型，个数，顺序至少一个不同**

3:返回类型，修饰符可以不同，但仅仅该两项不同不能构成重载。

**关于覆盖(override)**

**子类方法名，参数类型，返回类型必须和父类一样**，**子类方法不能缩小方法的访问权限(但可以扩大)，**如父类为public,子类不能是protected, private

**关于多态：一个引用在不同情况下的不同状态**

例如父类是animal 类，定义了cry 方法，猫类和狗类继承与animal类，override 了cry 方法，那么animal **ani = new cat(),**可以通过多态访问ani.cry方法。**（父类引用可以指向子类实例）**

**关于抽象类：abstract 修饰**

抽象方法必须在**抽象类，**解决父类方法的不确定性。

当子类继承的父类是抽象类时，**需要通过接口实现该抽象类（父类）的所有抽象方法**。(隐形要求时继承抽象类必须复写抽象方法)。

**抽象类不能实例化。**

**关于接口：**

**定义接口：**interface Usb{

public void start();

public void stop();

}

**class 类 implements 接口**:

类实现一个接口，需要实现该类的所有方法。类似于抽象类。

1:接口不能实例化

2:接口中**所有方法**不能有主体。但抽象类中可以拥有实现的方法。

3:接口中变量不能用private 和protected修饰,且**本质上为static**变量（不管与没有static 修饰），而且是**final**。

4:一个类可以实现多个接口：对单继承的一种补充

**class 类 implements 接口1，接口2**：

5: 经常将全局变量定义在接口内，通过 **接口名.变量名** 访问接口内变量。

6: 接口不能继承其他类，但可以继承另一个接口。

例子：

interface Usb{

public void start();

public void stop();

}

class camera implements Usb{

public void start(){ “start”}

public void stop(){“stop”}

}

class computer {

public void useUsb(**Usb usb**){usb.start();usb.stop()} **=>类似于多态时的调用，是后期绑定，只有在运行是才知道变量类型**

}

调用的时候即computer().useUsb(camera k1)

**关于final:**

1:当不希望父类的某个方法被子类override

2:当不希望某个变量不被修改

3:当不希望某个类被继承时

**关于数组:**

数据类型 数组名称［］＝ new 数据类型［大小］

数组名称.length 返回长度

对象数组：

Dog dogs[] = new Dog[6];

**dogs[0] = new Dog[]; =>必须有,需要new 对象**

index 不能out of bounds.

**关于移位:**

算数右移>>：低位溢出，符号位不变，用**符号位补溢出的高位**

算数左移<<：符号不变，低位补0(相当于乘以2^k,k位移动位数，负数同样适合)

逻辑右移>>>:低位溢出，高位补0

**关于集合类：**

**List:** ArrayList, LinkedList, Vector, Stack

**Map:** HashMap, Hashtable

**Set:** HashSet, TreeSet

**Queue:**

Arraylist:保存的类型是object 类，如果 Dog tmp=**(Dog)**a.get(0)**(不行，因为object类不能赋予dog类，需要强制转换**；

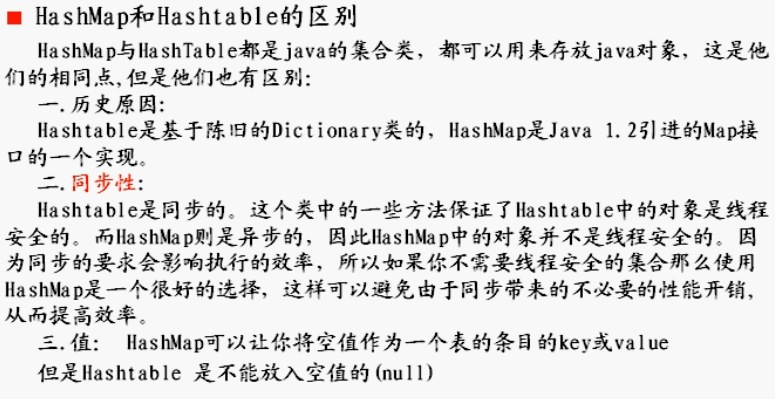
如果使用泛型，即ArrayList<Dog> = new ArrayList<Dog>();

遍历一个hashmap h:

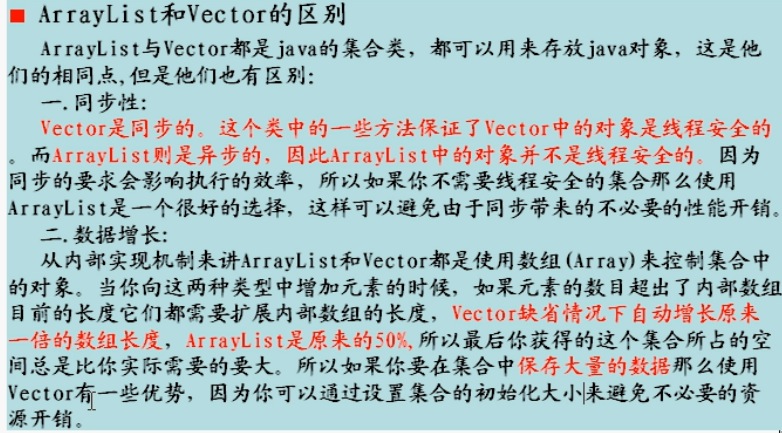
Iterator it=h.keySet().iterator(); =>取出的key set是无序的

While(it.hasNext()){String key=it.next().toString();}

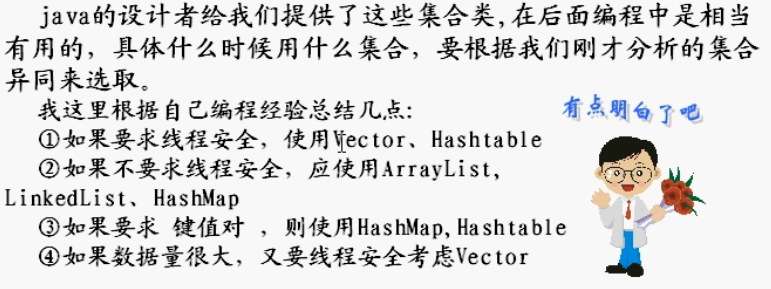
HashMap 和 HashTable 区别：



ArrayList 和 Vector :



**总结：**

****

**关于泛型：**

类似于c++的 模版，

可以免除调用ArrayList 返回对象的强制转换，ArrayList<Dog> = new ArrayList<Dog>()。

可以适用于建立一个适合多类型的类，如：

class A<T>{

privte T o;

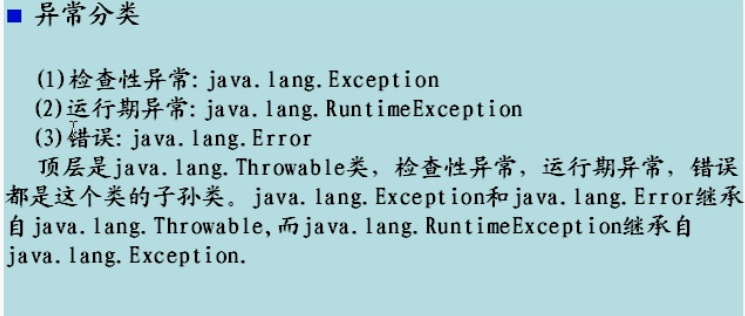
public A(T a){

o = a;

}

}

**关于异常：**

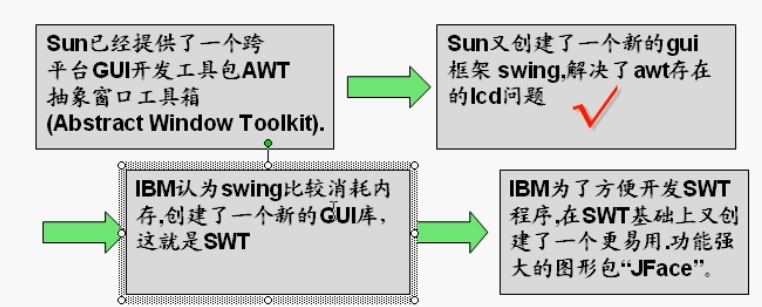
****

**String:**

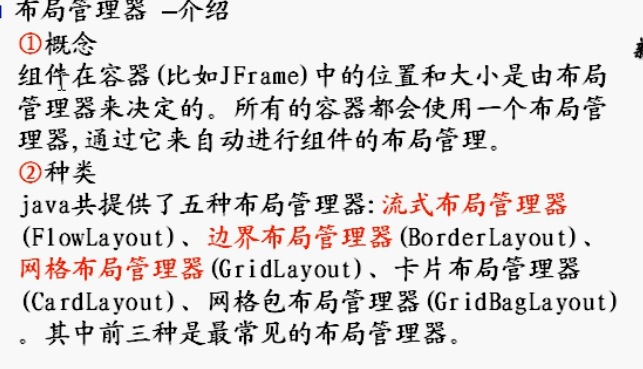
String a = “abc”;String s=”abc”; => a 和 s 指向的地址是一样的。(jvm中有string pool)

但是如果new 的化，不会有次问题，String a = **new** String(“abc”)

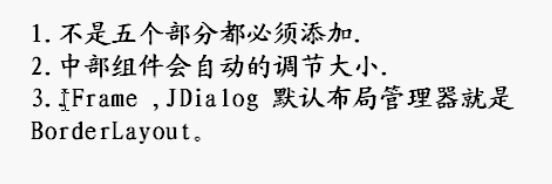
**关于GUI:**



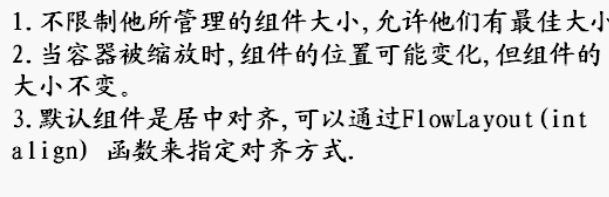
关于布局管理器：



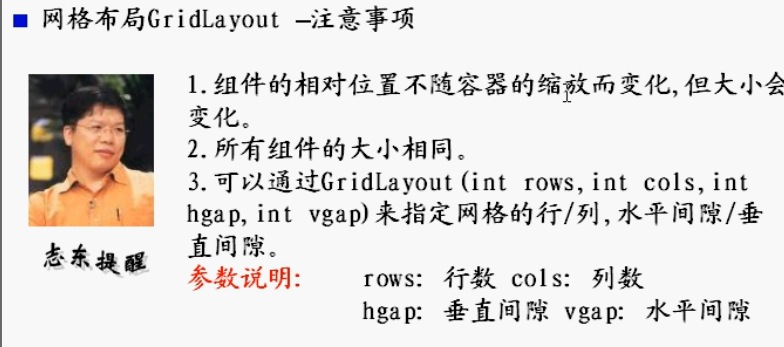
BorderLayout:



FlowLayout:



GridLayout:





JPanel:

一个图形界面有一个JFrame, JFrame可以里面有多个JPanel.

JTextField: 文本框

JPasswordField: 密码框

JLabel: 标签,可以放入一个图片（JLable(new ImageIcon(“图片的路径”))）

JCheckBox: 复选框组建

JRadioButton: 单选框组建 （要把此对象放入到ButtonGroup里面处理）

JComboBox:下拉框组建

JList:列表组建

JScrollPane:滚动窗格组建。=>输入是JList。

记事本: JTextArea, JMenuBar, JMenu,JMenuItem,JFileChooser(文件选择组建)

**java绘图：**

继承JPanel:

覆写 paint(Graphics g)方法：

注意在panel里面画其他物品时，要传入相同的画笔，即g.

Paint()方法将会被调用：所以其占有CPU 比较大

1:窗口最大化，最小化

2:窗口大小发生改变

3:repaint函数调用

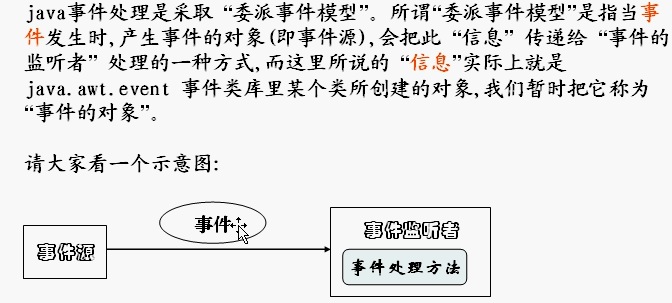
4:第一次显示窗口的时候自动调用。

**Graphic 类：**



ImageIO 可以读入图片。

**java 事件处理：**



**注册监听**：事件源.addActionListener(事件监听者),一个事件源可以被多个监听者监听。

怎样区分不同的事件源：事件源.setActionCommand(有关信息，如”黑色”)

事件监听者实现接口ActionLisener，要覆写actionPerformed(ActionEvent e) 方法：

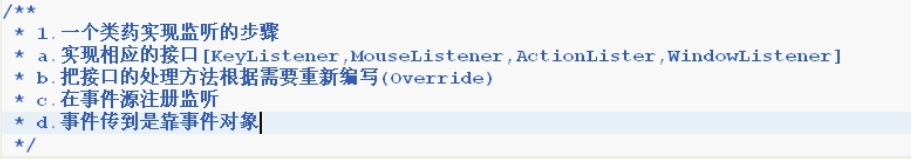


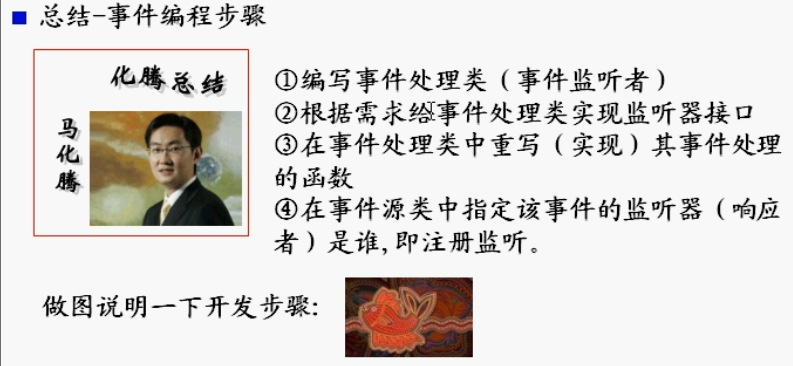
键盘（KeyListener接口，实现keyPressed(),keyReleased(),keyTyped()方法），按钮鼠标等都是不同的事件，要区别对待。

任何一个类只要实现了相应的接口，就可以成为事件监听者。常用的接口有：

KeyListener，MouthListener, ActionListener, KeyListener,WindowListener.

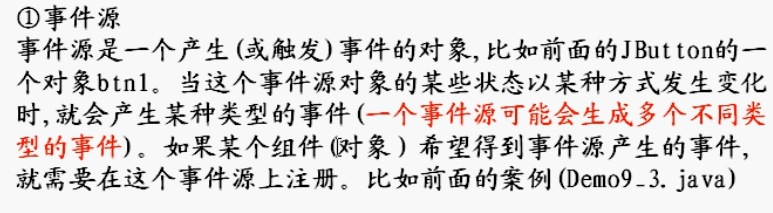
**一般的步骤：**

****

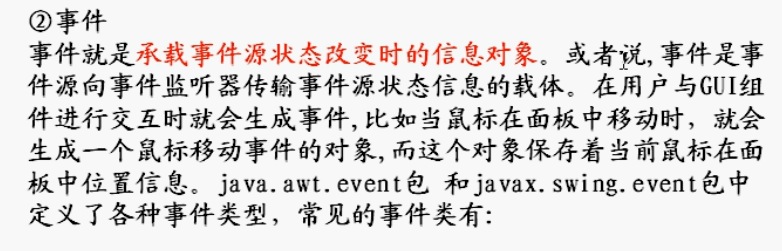
****

**深入事件处理：**

**事件源；**

****

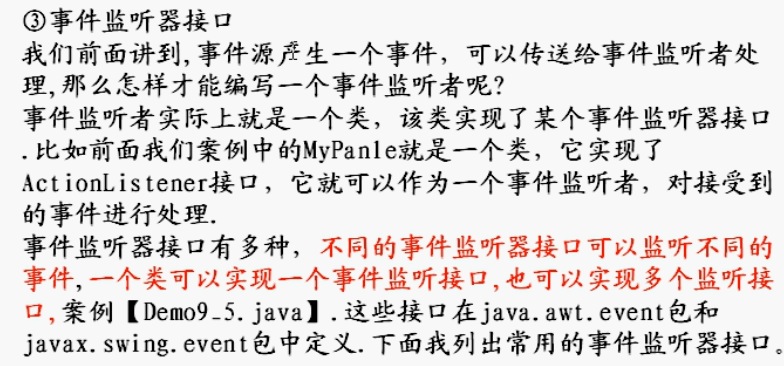
**事件：**

****

****

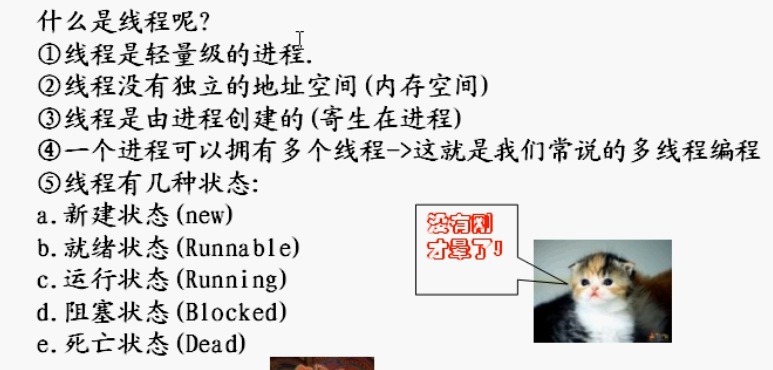
还有一种接口，可以监听鼠标在pannel上拖拽的信息，MouseMotionListener

**事件监听器：**

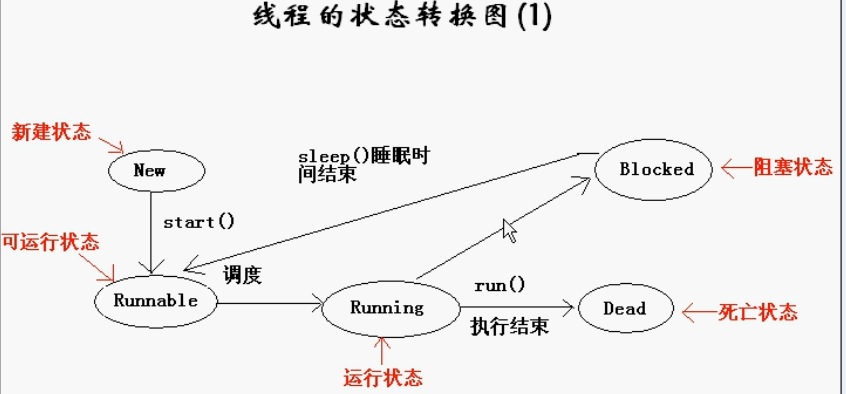
****

**关于线程:**

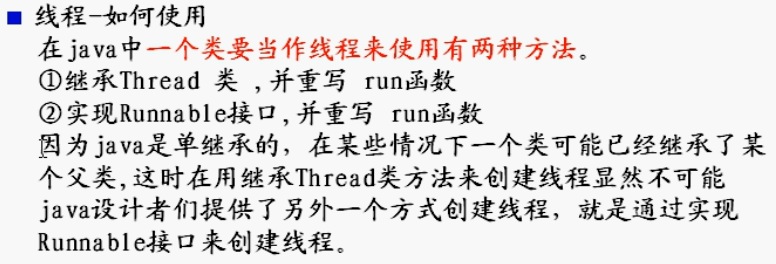
**定义：**

****

**状态图：**

****

**任何类都能当作一个线程，两种方法,在常用包里面：**

****

**继承Thread 类：**Cat extends Thread

Cat cat = new Cat()

Cat.start()

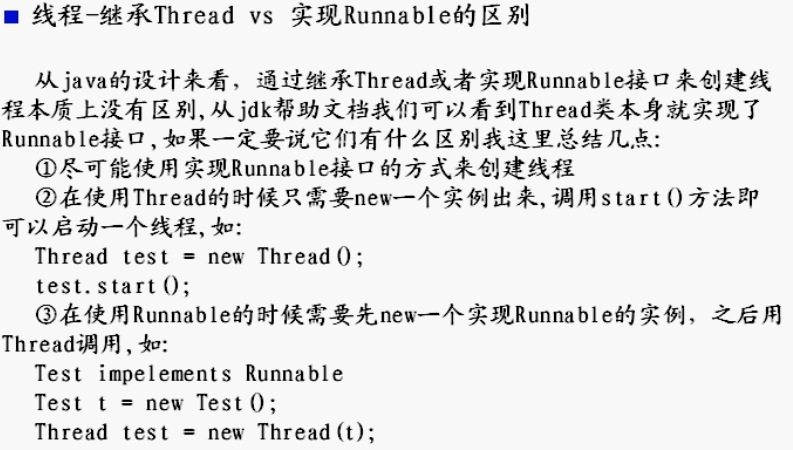
**实现Runnable接口：**Dog implements Runnable

Dog dog = new Dog() //首先对象

Thread t = new thread(dog) //将此对象加入到一个新的Thread对象

t.start()

**两种方法的区别：**

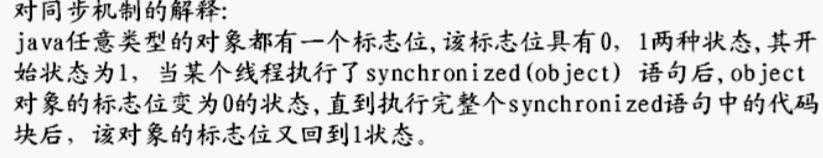
****

**线程深入理解：**

1: 线程对象只能启动一个线程

2:线程对象只能启动一次，即start()一次。否则有异常抛出。因此最好线程启动时最好try … catch …

3: 线程得同步问题，原子性。synchronized(Object){你要同步得代码},这里object类似于一个门，即对象锁。任意一个对象都可以作为对象锁，一般用this：



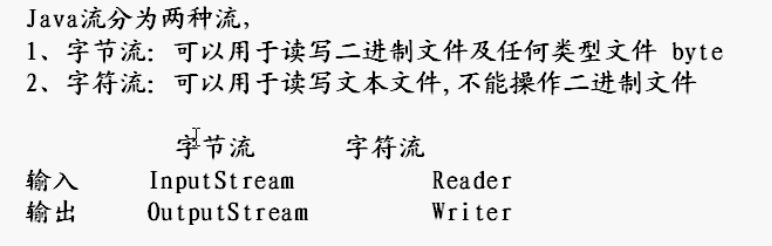
**Aside:**

休息1s: Thread.sleep(1000);

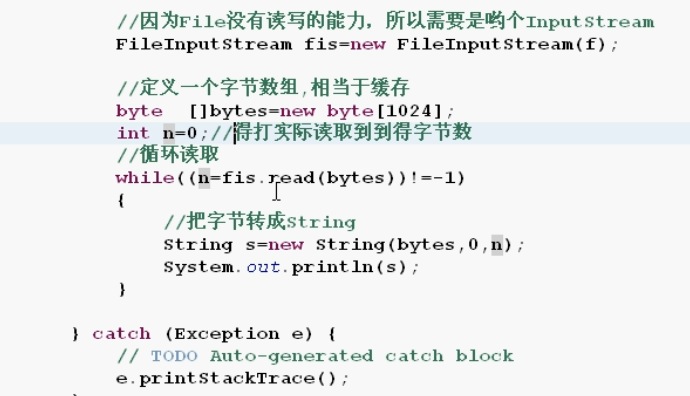
得到当前线程得名字：Thread.currentThread().getName()

**关于IO:**

分类：

****

FileInputStream 类，



最后**必须在finally**里面关闭文件。

FileOutputStream 类:



写入两行字符串，注意有’\r\n’

FileReader 类：

FileWritter类：字符流，操作char

BufferedReader 类：

BufferedWriter类：操作string.效率高。