1，==比较的是俩个值的地址值是否相等

2，string重写object里面的equals方法

3，字符串的常量是在字符串常量池区域内，堆空间存放的是地址值

4，instanceof用来判断一个是不是某个对象的实例，如果是一个实例，返回true，否则返回false

注意：：以Java Application的方式运行java类的时候，有时会提示Could not find the main class.Program will exit出现这个问题的解决办法：

①在该程序中写入一个主函数

②在项目上右击--->Properties--->Java Compiler,

将运行环境改成与编译环境一致

、、、、、、、重写equals方法：

//比较俩个test对象的属性是否相同，相同则返回true，否则返回false

```
public boolean equals(Object obj){
        if (this==obj) {
                return true;
        }
        else if (obj instanceof test) {
                test te=(test)obj;
                return this.name.equals(te.name)&&this.age==te.age;
        }
        else {
                return false;
        }
}
```

注意：当一个类的对象需要存放到集合里面的时候，存在set中，hashset（只能存不相同的元素），这个类需要重写equals方法。

5，tostring方法

在没有重写tostring方法的类中打印的是当前对象所在的类以及该对象所在堆空间的对象实体的信息

ctrl+shift+t在eclipse中查看相应的源码

①java.lang.Object类的toString方法的定义如下：

```
public String toString() {
        return getClass().getName() + "@" + Integer.toHexString(hashCode());
    }
```

②当我们打印一个对象的引用时，实际上就是调用这个对象的toString方法。

③当我们打印的对象所在的类没有重写Object中的toString方法时，调用的是Object中的toString方法，返回此对象所在的堆空间对象的实体信息。

④当一个类重写了toString方法的时候，打印的时候调用的就是重写的toString方法，常常这样重写，将对象的属性的信息返回

自动调用：source-->toString(选中属性)

注意：outline显示该类的变量和方法

6，像在string类，包装类，file类，date类中已经重写了toString方法

任何基本数据类型的东西+“”，最终都会转化为一个字符串

byte-->Byte

long-->Long

7，包装类（封装类wrapper）

int-->Integer

boolean-->Boolean

char-->Character

float-->Float

double-->Double

在一个java项目中想要那段代码执行就执行那段代码的方法：

项目-->build path-->add libraries-->junit-->jnit4加入一个类库

Junit单元测试类：

①当前工程下-右键build path-->add libraries-->Junit4

②在主类中，创建一个空参的无返回值的方法，如：public void test（）用于代码的测试，方法上声明@test

③导入import org.junit.Test

④在test（）方法中，进行代码的编写和测试

⑤测试：双击方法名，右键run-->as junit Test即可

8包装类：8中基本数据类型对应着一个类，此类即为包装类

基本数据类型包装类以及String之间的相互转换

①基本数据类型与包装类之间的转换

a， 基本数据类型-->包装类

调用相应的构造器

如：int i=1;

Integer i1=new Integer(i);

Integer中的参数可以是整型也可以是字符串类型，但是字符串类型的东西需要输一个数字型的字符串

会抛出java.lang.NumberFormatException的异常

注意：boolean b=true;

Boolean b2=new Boolean(b);

Boolean中的参数只有是true的时候才会返回一个true，其他任何情况返回的都是false，它不会抛异常

包装类类型的初始化值是null

b，包装类-->基本数据类型

调用包装类的相应的XxxValue（）方法。

如： int i2=i1.intValue();

JDK5.0之后，自动装箱和拆箱

如：Integer i3=12;---自动装箱

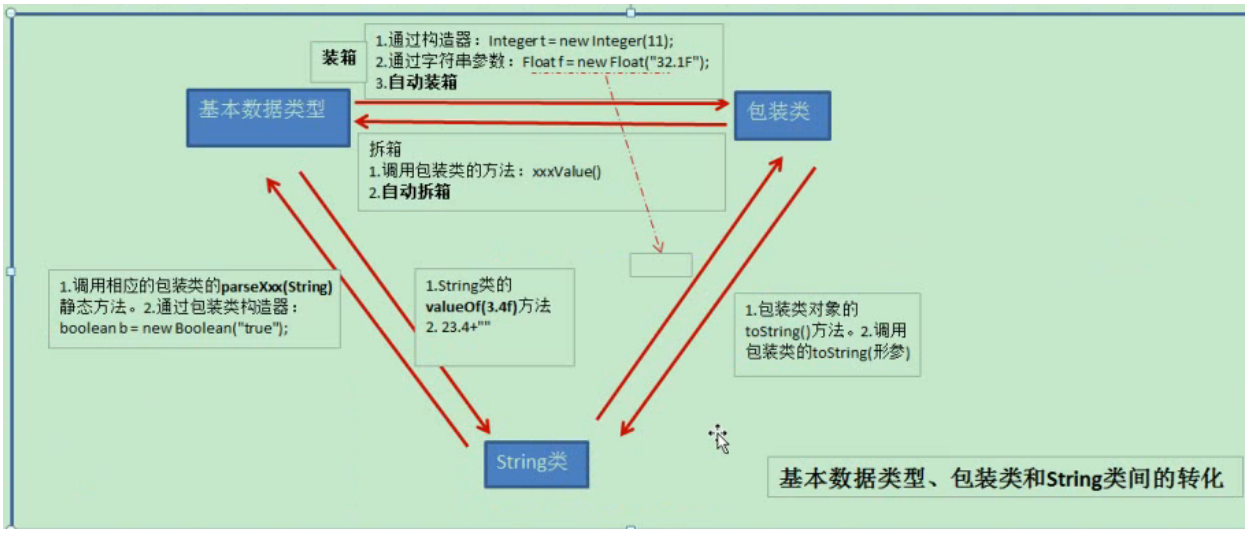
int i=i3; ---自动拆箱

其他数据类型也一样

②基本数据类型，包装类与包装类String类之间的转换

基本数据类型、包装类-->String类：调用String类重载的valueOf(Xxx x)方法

String类-->基本数据类型、包装类：调用包装类重载的parseXxx（数据类型）(String str)方法



scanner是在java.util包中

sort默认从小到大排序。

①Collections.sort(v1,Collections.reverseOrder());---->对向量v1进行排序，通过Collection.reverseOrder实现从大到小的排序。

②Collections.sort(v1);---->对向量v1进行排序，默认是从小到大的排序

Integer i=Collections.max(v1);取出向量v1中的最大值

java中的vector：（下标从0开始）

①vector可实现自动增长的对象数组，java.util.vector提供了向量类以实现类似动态数组的功能；

②创建一个向量类的对象之后，可以往其中随意的插入不同类的对象，

③对于预先不知道或者不愿预先定义数组大小，并且需要频繁地进行查找，插入，删除工作的情况，可以考虑使用向量类；

java中vector的方法：

①插入：

a，addElement（Object obj）；将obj插入向量的尾部，obj可以是任何类型的对象，如插入整数1

的时候，不能直接在obj处写1，而需要用Integer进行装箱

b，setElementAt（Object obj，int index）；将index处的对象设置为obj，原来的对象将被覆盖，

index不能超出v1中的已有的size（）-1；

c，insertElementAt（Object obj，int index）；在index指定的位置插入obj，原来的对象以及此后

的对象依次顺序往后顺延；

②删除：

a，removeElement（Object obj）；从向量中删除obj，若有多个存在，则从头开始测试，删除找

到的第一个与obj一样的对象；

b，removeAllElement（）；删除向量中的所有对象；

c，removeElementAt（int index）；删除index所指定的地方的对象；

③查询：

a，indexOf（Object obj）；从向量的头部开始搜索obj，返回所遇到的第一个obj所对应的下标，

若不存在，则返回-1；

b，indexOf（Object obj，int index）；从index所表示的下标处开始搜索obj

c，lastindexOf（Object obj）；从向量尾部开始逆向搜索obj；

d，lastindexOf（Object obj，int index）；从index所表示的下标处由尾至头逆向搜索obj

e，firstElement（）；获取向量对象中的首个obj

f ，lastElement（）；获取向量对象的最后一个obj

v1.size（）返回v1的长度