**Using Objects as Parameters**

So far, using simple types as parameters to methods.

However, common to pass objects to methods.

For example, consider the following short program:



This program generates the following output:

ob1 == ob2: true

ob1 == ob3: false

As you can see, the **equalTo( )** method inside **Test** compares two objects for

equality and returns the result. That is, it compares the invoking object with the

one that it is passed. If they contain the same values, then the method returns

**true**. Otherwise, it returns **false**. Notice that the parameter **o** in **equalTo( )**

specifies **Test** as its type. Although **Test** is a class type created by the program,

it is used in just the same way as Java’s built-in types.

One of the most common uses of object parameters involves constructors.

Frequently, you will want to construct a new object so that it is initially the

same as some existing object. To do this, you must define a constructor that

takes an object of its class as a parameter. For example, the following version

of **Box** allows one object to initialize another:



**Returning Objects**

A method can return any type of data, including class types that you create. For

example, in the following program, the **incrByTen( )** method returns an object

in which the value of **a** is ten greater than it is in the invoking object.

