Bozo *pb = new Bozo; // use default

Just as a program invokes a constructor when an object is created, it invokes a destructor when an object is destroyed. You can have only one destructor per class. It has no return type, not even `void`; it has no arguments; and its name is the class name preceded by a tilde. The Bozo class destructor, for example, has the following prototype:

```
~Bozo(); // class destructor
```

Class destructors become necessary when class constructors use `new`.

**Knowing Your Objects: The this Pointer**

There's still more to be done with the Stock class. So far each class member function has dealt with but a single object, which has been the object that invokes it. Sometimes, however, a method might need to deal with two objects, and doing so may involve a curious C++ pointer called `this`. Let's see how this need can unfold.

Although the Stock class declaration displays data, it's deficient in analytic power. For example, by looking at the `show()` output you can tell which of your holdings has the greatest value, but the program can't tell because it can't access `total_val` directly. The most direct way of letting a program know about stored data is to provide methods to return values. Typically, you use inline code for this:

```cpp
class Stock
{
private:
    ...
    double total_val;
    ...
public:
    double total() const { return total_val; }
    ...
};
```

This definition, in effect, makes `total_val` read-only memory as far as a direct program access is concerned.