Q: I've made connections in InterfaceBuilder like I have a zillion times before. So, why are all of my outlets nil when I run my program?

A: In earlier releases, you had to provide a method to set each outlet that you declared in a new class. For example, if your subclass had an outlet named **myOutlet**, the source code for your subclass had to declare and implement this method:

```
- setMyOutlet:anObject
{
        myOutlet = anObject;
        return self;
}
```

In InterfaceBuilder, the Unparse command in the Classes window's pop-up list would create these "set" methods automatically.

These outlet initialization methods are no longer required. However, for backward compatibility, if an object responds to such a message, the runtime system uses them to initialize outlets. Basically, the runtime system does this:

```
if ([yourObject respondsTo:@selector(setMyOutlet:)])
        setMyOutlet:anObject
```

All is hunky dory, if you remember the days when these methods were required. Now that InterfaceBuilder does not generate them automatically, it is not obvious that these method names have special meaning. So, if you have a method name **setMyOutlet** you must set the outlet within that method. If you don't initialize the outlet,

then **myOutlet** is nil at runtime.

Overall it is best to avoid **setMyOutlet** style method names altogether, unless you are consciously taking advantage of this "feature," and are doing additional initialization within the "set" method. Care must be taken, as the order in which outlets and other objects are initialized is neither guaranteed nor predictable.

QA772

Valid for 1.0, 2.0, 3.0