[0640] Casting Between Related Classes

Ugh. Don't do this. Find another way. #endofstory

"What?" "I have to talk about it?" "Fine."

OK, I guess casting between classes might be needed in certain special circumstances, but seriously, all of the rules associated with this are not something a programmer should be expected to remember - nor will they. If you HAVE to cast one object into another, TEST like crazy to make sure what you thought would happen actually did happen and then, for the love of all that is good and holy, please add a comment explaining what you think is happening. That way, in 5 years when this code mysteriously breaks, we'll be able to track you down and beat you senseless. #youvebeenwarned

Upcasting and Downcasting

There are two types of quasi-legitimate object casts - upcasting and downcasting. Upcasting is easy. It means that you are converting an object of a subclass into its superclass. That means you just lose all of the features that your subclass brought to the table and keep all of the features that the superclass provided. Makes sense - kind of. And I will admit that upcasting is used occasionally.

Downcasting - yuck. Downcasting is where you take a superclass' object and convert it into a subclass' object on the fly. With one specific exception, downcasting is gross and dumb and you should never ever ever do it. #ever

The one exception is if you first check to see if the object you want to downcast already belongs to the subclass you want to downcast to. (Huh?)

For example, let's say you have a collection of Bird objects. (Again with the birds? Oy!) You can step through that collection with a foreach loop to see if any of them are Eagles (using the "is" keyword). If any of the Bird objects turn out to be Eagles, you can then downcast them to the Eagle subclass safely.

```
foreach (Bird bird in birds)
{
   if (bird is Eagle)
   {
      ((Eagle)bird).DoSomeOfThatEagleStuffMav();
   }
}
```

THAT. IS. THE. ONLY. TIME!