```
respondsTo: aSymbol
    "Answer whether the method dictionary of the receiver's class contains
    aSymbol as a message selector."

    ^self class canUnderstand: aSymbol
```

The implementation is trivial, delegating the check to selector `canUnderstand:` defined in `Behavior`:

```
canUnderstand: selector
    "Answer whether the receiver can respond to the message whose selector
    is the argument. The selector can be in the method dictionary of the
    receiver's class or any of its superclasses."

    (self includesSelector: selector) ifTrue: [^true].
    superclass == nil ifTrue: [^false].
    ^superclass canUnderstand: selector
```

Finally, `includesSelector:` is also defined in `Behavior`, where it boils down to checking the method dictionary of the class:

```
includesSelector: aSymbol
    "Answer whether the message whose selector is the argument is in the
    method dictionary of the receiver's class."

    ^ self methodDict includesKey: aSymbol
```

When a receiver gets a message it does not understand, its standard response is sending the `doesNotUnderstand:` message to the system. If we would rather try to remedy the situation ourselves, we only need to override the message, doing something like the following:

```
doesNotUnderstand: aMessage
    "Handles messages not being understood by attempting to
    proxy to a target"
    target perform: aMessage selector withArguments: aMessage arguments].
```

We assume that `target` refers to the proxy object that we hope will be able to handle the misdirected message.

Latent typing is not an excuse for sloppy programming. We duck types, not responsibilities.

Problems

Public inheritance means "is-a." That requires careful thinking from the part of the programmer to come up with class hierarchies that really fit this pattern. If you have a class with a method and a subclass where you realize that method makes no sense, it is beyond the scope of public inheritance, and it should be a sign of bad design. Languages, however, prove accommodating.

In C++, you can get away with it by making the nonsensical method either return an error or throw an exception. The prototypical example concerns birds (Meyers 2005, item 32):