# Object Oriented Languages

## Metaobject Protocols

**Structure**: how are values to be stored in an object?

# Object Oriented Languages

## Metaobject Protocols

**Structure**: how are values to be stored in an object?

Most objects are implemented as vectors with the object accessors implemented as simple vector references

# Object Oriented Languages

## Metaobject Protocols

**Structure**: how are values to be stored in an object?

Most objects are implemented as vectors with the object accessors implemented as simple vector references

The generic function `compute-and-ensure-slot-accessors` describes how slots are accessed

# Object Oriented Languages

## Metaobject Protocols

**Structure**: how are values to be stored in an object?

Most objects are implemented as vectors with the object accessors implemented as simple vector references

The generic function `compute-and-ensure-slot-accessors` describes how slots are accessed

The standard method does a simple vector access

# Object Oriented Languages

## Metaobject Protocols

**Structure**: how are values to be stored in an object?

Most objects are implemented as vectors with the object accessors implemented as simple vector references

The generic function `compute-and-ensure-slot-accessors` describes how slots are accessed

The standard method does a simple vector access

So if you want slots that live on disk rather than in memory, or count the number of times they are accessed, add a method here

# Object Oriented Languages

## Metaobject Protocols

**Classes**: The generic functions `allocate` and `initialize` describe how objects are created

# Object Oriented Languages

## Metaobject Protocols

**Classes**: The generic functions `allocate` and `initialize` describe how objects are created

So if you want to have an object that lives outside the normal class hierarchy, or has a different structure, add methods here

# Object Oriented Languages

## Metaobject Protocols

**Classes**: The generic functions `allocate` and `initialize` describe how objects are created

So if you want to have an object that lives outside the normal class hierarchy, or has a different structure, add methods here

And so on

# Object Oriented Languages

## Metaobject Protocols

There are standard methods everywhere, of course, that do the “usual” things, i.e., they implement the behaviour you would expect from a normal OO language

# Object Oriented Languages

## Metaobject Protocols

There are standard methods everywhere, of course, that do the "usual" things, i.e., they implement the behaviour you would expect from a normal OO language

For example, a method on `initialize` that fills in a newly allocated object

```
(defmethod initialize ((cl <class>) keywords)
  (call-next-method)
  (let ((direct-supers (class-direct-superclasses cl))
        (direct-slotds (find-keyword :direct-slots keywords ()))
        (direct-inits (find-keyword :direct-keywords keywords ())))
    (unless (compatible-superclasses-p cl direct-supers)
      (error "incompatible superclasses:~%~s can not be a subclass of ~%~s"
             cl direct-supers))
    (setf (class-precedence-list cl)
          (compute-class-precedence-list cl direct-supers))
    (setf (class-keywords cl)
          (compute-keywords cl direct-inits
                            (compute-inherited-keywords cl direct-supers)))
    (let* ((inherited-slotds (compute-inherited-slots cl direct-supers))
           (effective-slotds
            (compute-and-ensure-slot-accessors
             cl (compute-slots cl direct-slotds inherited-slotds)
             inherited-slotds)))
      (setf (class-slots cl) effective-slotds)
      (setf (class-instance-length cl) (length effective-slotds)))
    (mapcar #'(lambda (super)
                (add-subclass super cl)) direct-supers))
  cl)
```

# Object Oriented Languages

## Metaobject Protocols

Exercise. Investigate the `Metaobject` class in Java

Exercise. Investigate Joose, the JavaScript Metaobject system

Exercise. Investigate Moose, the Perl Metaobject system

# Object Oriented Languages

## Metaobject Protocols

Metaobject protocols are very powerful and so are easy to abuse

# Object Oriented Languages

## Metaobject Protocols

Metaobject protocols are very powerful and so are easy to abuse

And easy to misuse accidentally

# Object Oriented Languages

## Metaobject Protocols

Metaobject protocols are very powerful and so are easy to abuse

And easy to misuse accidentally

But they do allow you to do exactly what you want in an OO system

# Object Oriented Languages

## Metaobject Protocols

Metaobject protocols are very powerful and so are easy to abuse

And easy to misuse accidentally

But they do allow you to do exactly what you want in an OO system

You are not limited by the language, only your imagination

# Object Oriented Languages

## End of OO

Object Oriented languages and OO concepts are widely used

# Object Oriented Languages

## End of OO

Object Oriented languages and OO concepts are widely used

There are a large number of ways of doing OO

# Object Oriented Languages

## End of OO

Object Oriented languages and OO concepts are widely used

There are a large number of ways of doing OO

From no classes and no inheritance all the way to MOPs

# Object Oriented Languages

## End of OO

Object Oriented languages and OO concepts are widely used

There are a large number of ways of doing OO

From no classes and no inheritance all the way to MOPs

You need to able to make a choice!

# Object Oriented Languages

## End of OO

*Object-oriented programming is an exceptionally bad idea which could only have originated in California*
Edsger Dijkstra

# End of Languages

There are many languages out there, both general purpose and specialist

# End of Languages

There are many languages out there, both general purpose and specialist

It's not a case that one language is better than another, more that one is *better for the problem in hand*

# End of Languages

There are many languages out there, both general purpose and specialist

It's not a case that one language is better than another, more that one is *better for the problem in hand*

Be aware and learn the *concepts*, they are transferable between many languages

# End of Languages

There are many languages out there, both general purpose and specialist

It's not a case that one language is better than another, more that one is *better for the problem in hand*

Be aware and learn the *concepts*, they are transferable between many languages

**Pick the right tool for the job**