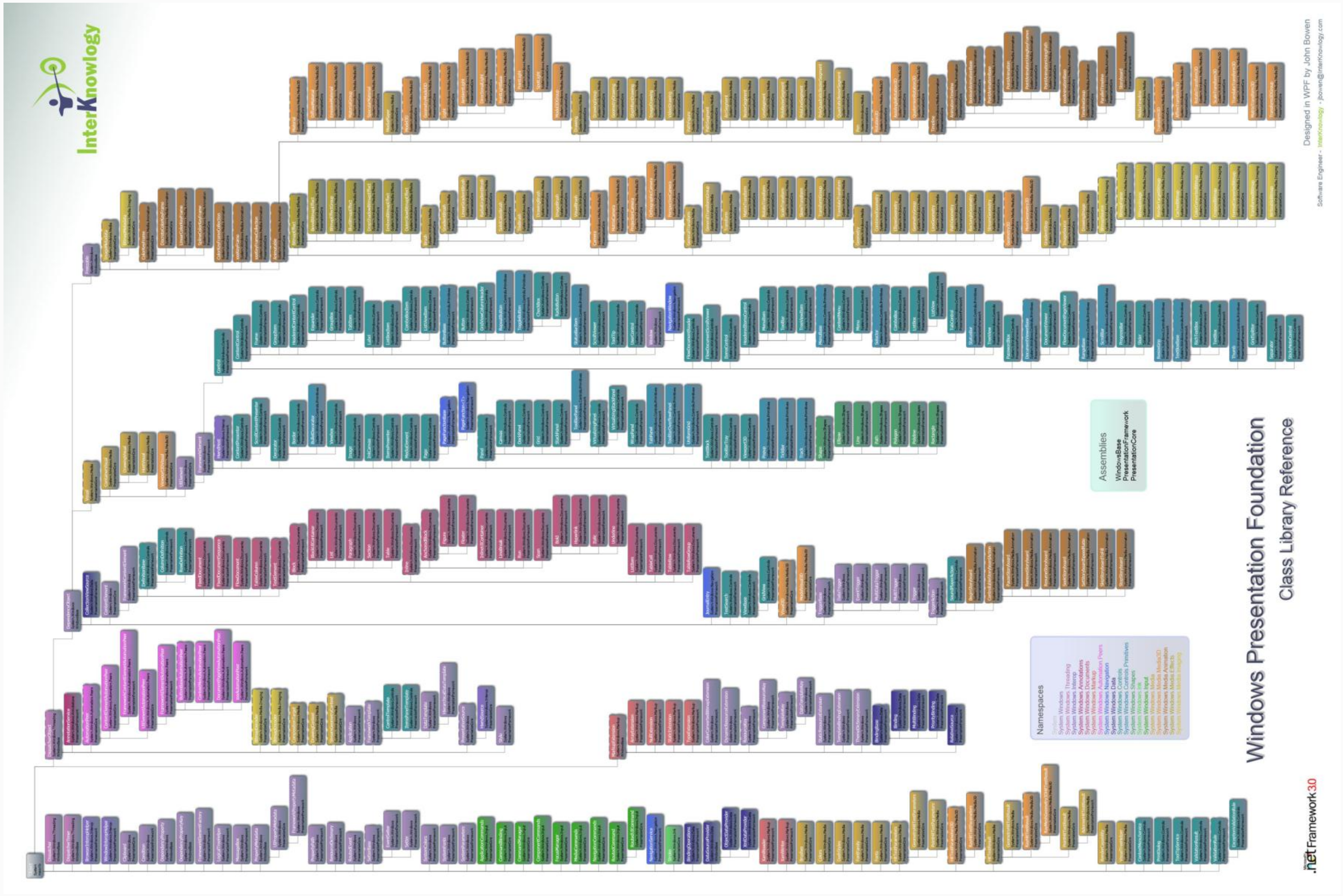
Escape from the Maze of Twisty Classes

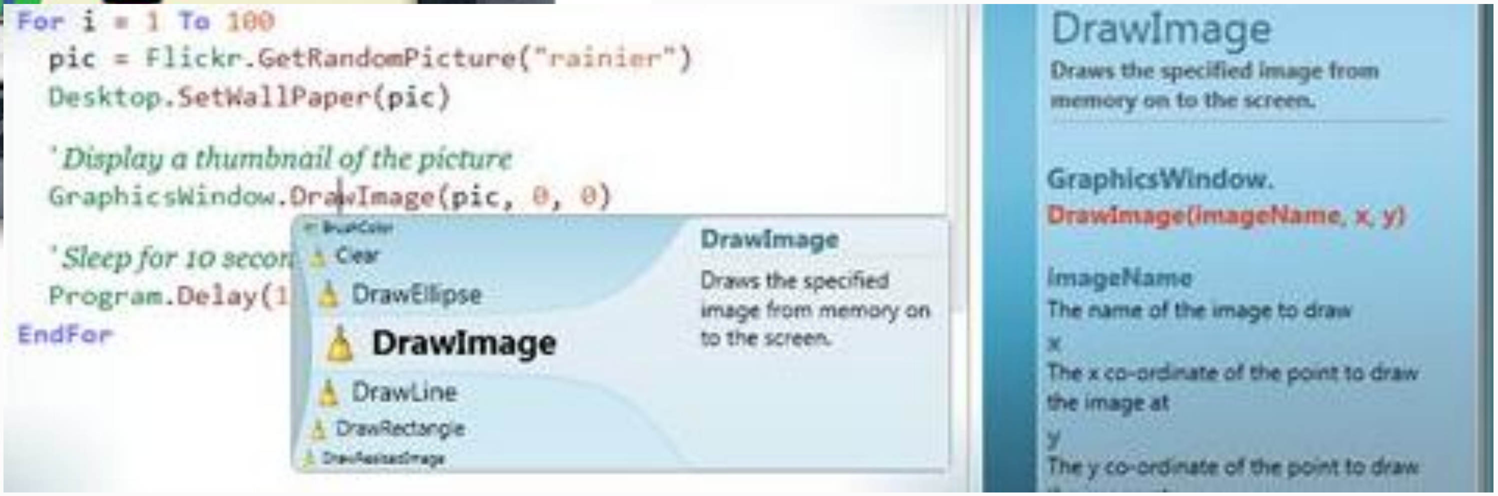
Class Library Overload



Batteries Included

Batteries Included

GPS & Intellisense



```
For i = 1 To 100
  pic = Flickr.GetRandomPicture("rainier")
  Desktop.SetWallPaper(pic)

  'Display a thumbnail of the picture
  GraphicsWindow.DrawImage(pic, 0, 0)

  'Sleep for 10 secon
  Program.Delay(1
EndFor
```

DrawImage

Draws the specified image from memory on to the screen.

DrawImage

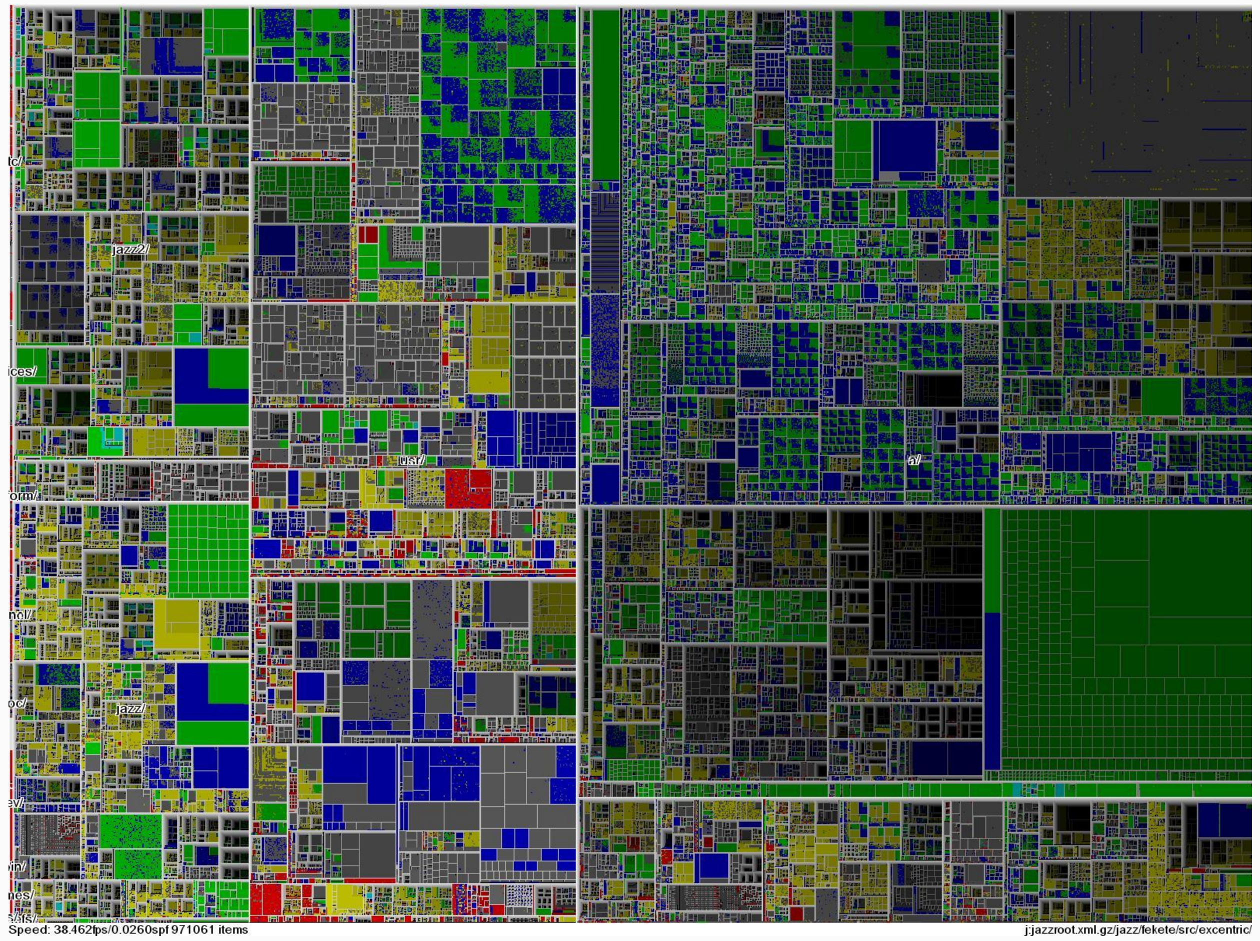
Draws the specified image from memory on to the screen.

GraphicsWindow.
DrawImage(imageName, x, y)

imageName
The name of the image to draw
x
The x co-ordinate of the point to draw the image at
y
The y co-ordinate of the point to draw

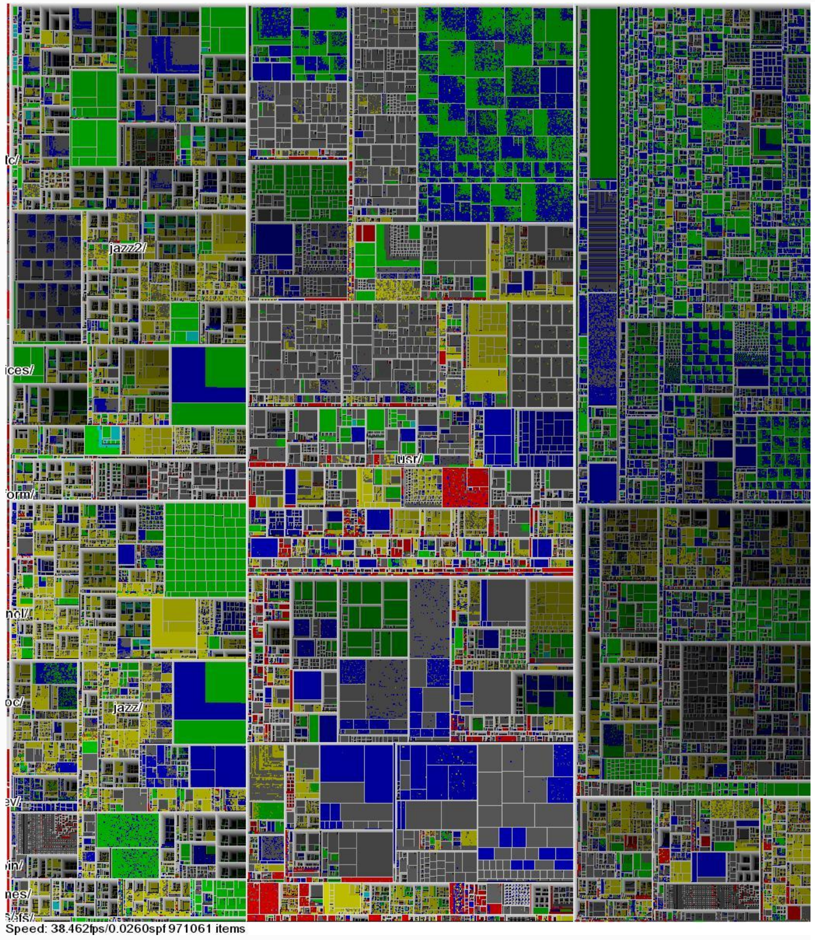
Breadth & Hick's Law

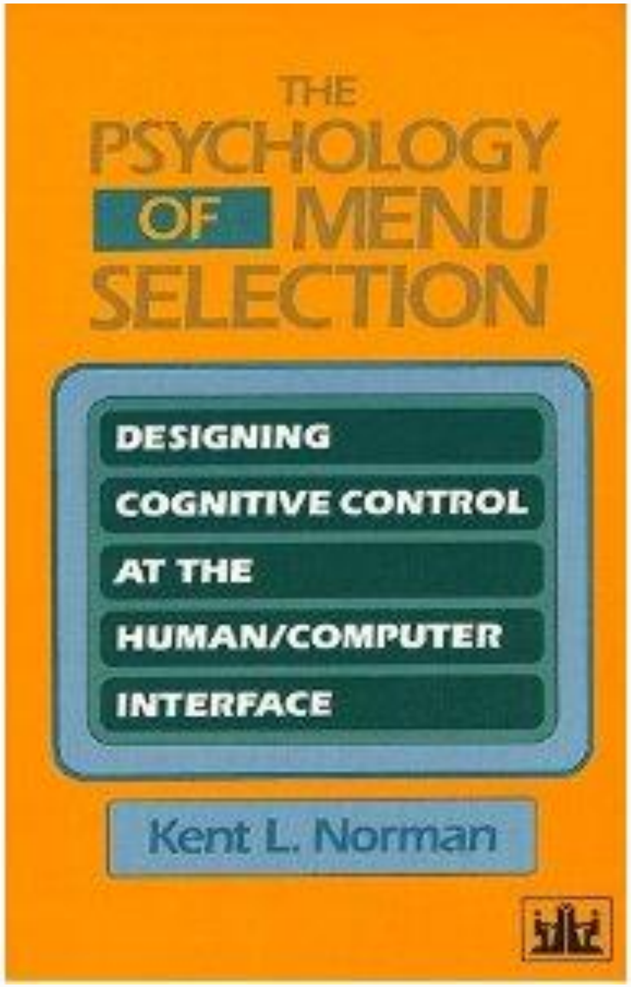
Depth > Breadth*


tc/
jazz2/
ices/
usr/
orm/
a/
nol/
oc/
jazz/
ev/
bin/
nes/
s-fs/
Speed: 38.462fps/0.0260spf 971061 items
j:jazzroot.xml.gz/jazz/fekete/src/excentric/

* > ➔ more evil

Depth > Breadth

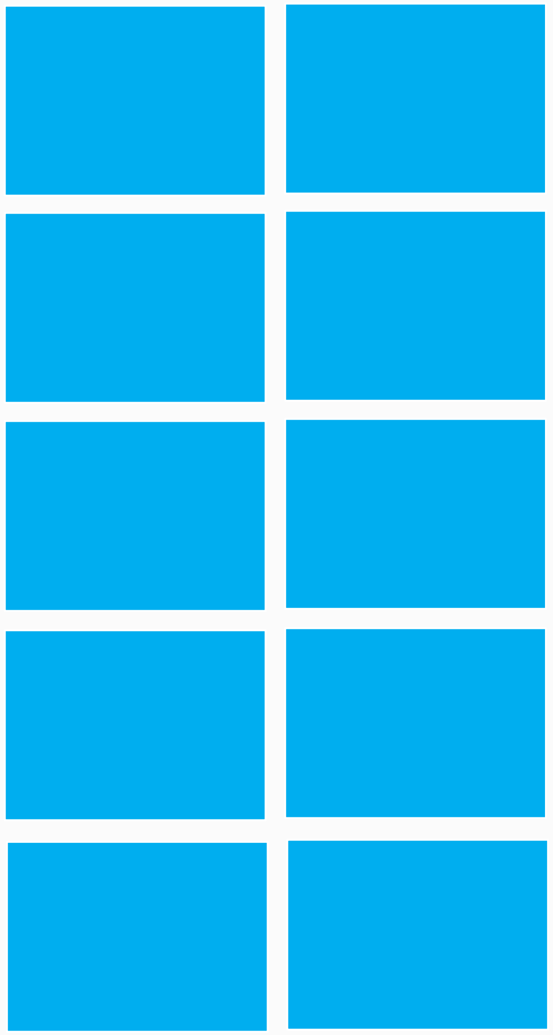




Fat classes → Skinny mixins (traits)

+Members/class decreases

+More reuse



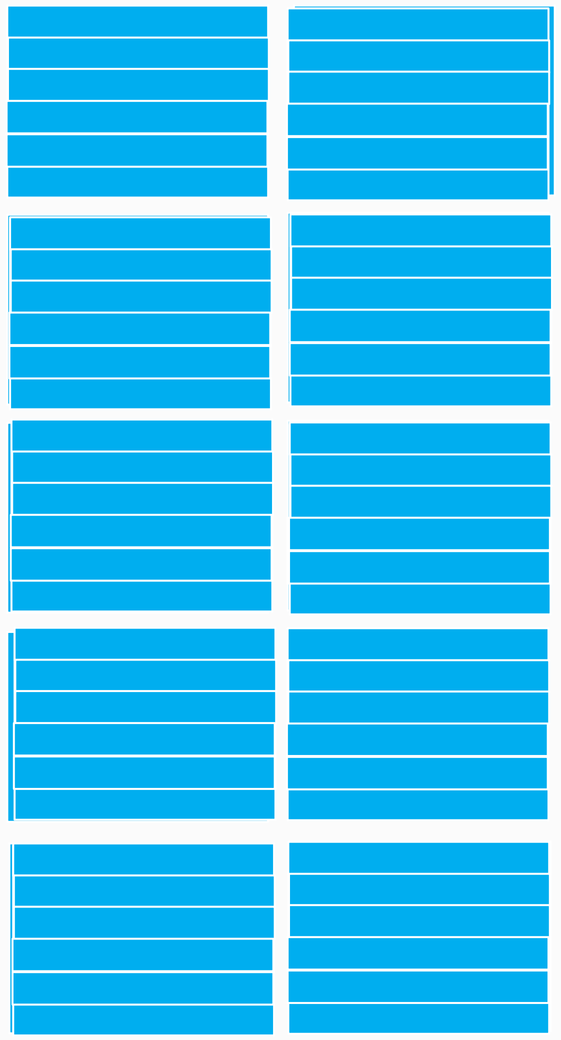
Fat classes → Skinny mixins (traits)

+Members/class decreases

+More reuse

- Maze of twisty traits!

- More depth



Hide Depth

List everything as a choice!

Only “one” namespace

Infer dependencies

Hide Depth

```
trait Panel {}

trait Widget {
  ref Parent : Panel;
}




val myPanel =      new  Panel();
val myWidget =     new Widget() { Parent = myPanel };
```

Hide Depth

```
trait Panel {}
trait Canvas :: Panel {}
trait Widget {
  ref Parent : Panel;
}
trait CanvasWidget : Widget {
  var Position : Point;
  Parent <: Canvas;
}

val myPanel =      new  Panel();
val myWidget =     new Widget() { Parent = myPanel };

myWidget.Position = new Point(100,100);
```

Hide Depth

```
trait Panel {}
trait Canvas :: Panel {}
trait Widget {
  ref Parent : Panel;
}
trait CanvasWidget : Widget {
  var Position : Point;
  Parent <: Canvas;
}

val myPanel =      new Canvas();
val myWidget =new CanvasWidget(){ Parent = myPanel };

myWidget.Position = new Point(100,100);
```

Dealing with Breadth

Alphabetical Lists

Semantic Categorization

Search & Ranking!

Dealing with Breadth

- Alphabetical Lists
- Semantic Categorization
- Search & Ranking!
  - Automatic
  - Explicit

Trait Rank

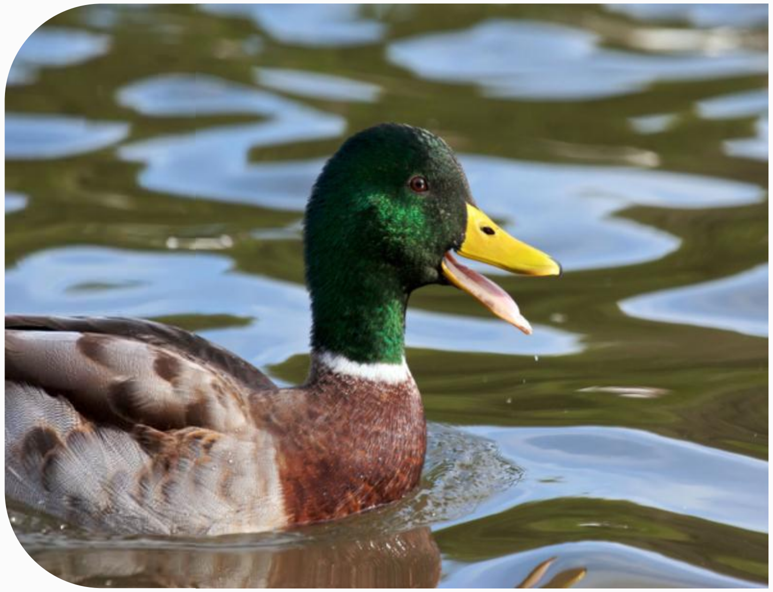
B : A $\phi(A) >= \phi(B)$

B + A $\phi(A) >= k^3 \times \phi(B)$

B +/: A $\phi(A) >= \phi(B)$
$\phi(B) >= k^3 \times \phi(A)$

sym	coef
:	1
++	$k_{< 1}$
+	k^3
*	k^6
-	k^9

Trait Rank



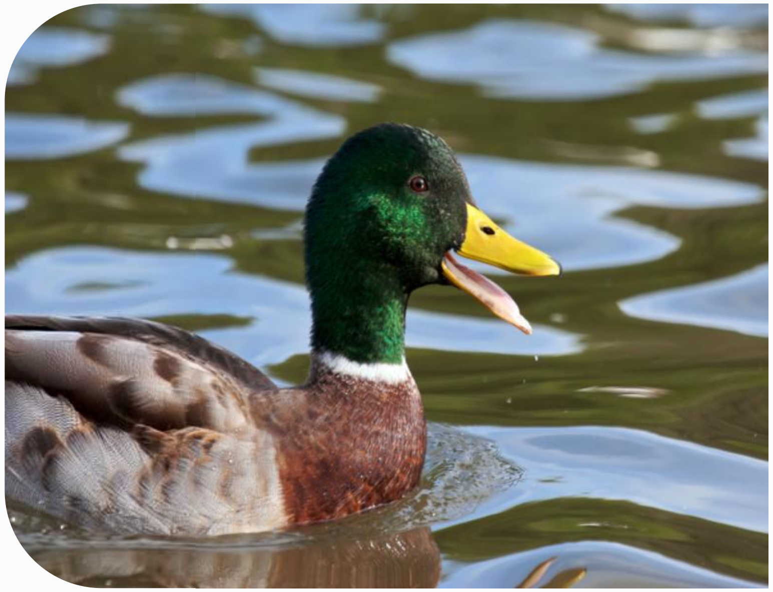
```
trait Duck */: Animal

def Quack -/: Animal +/++ Duck
```

Any animal **can** quack (*but not likely*).

Animals that quack are **very likely** to be ducks.

Ducks **like** to quack.

Trait Rank



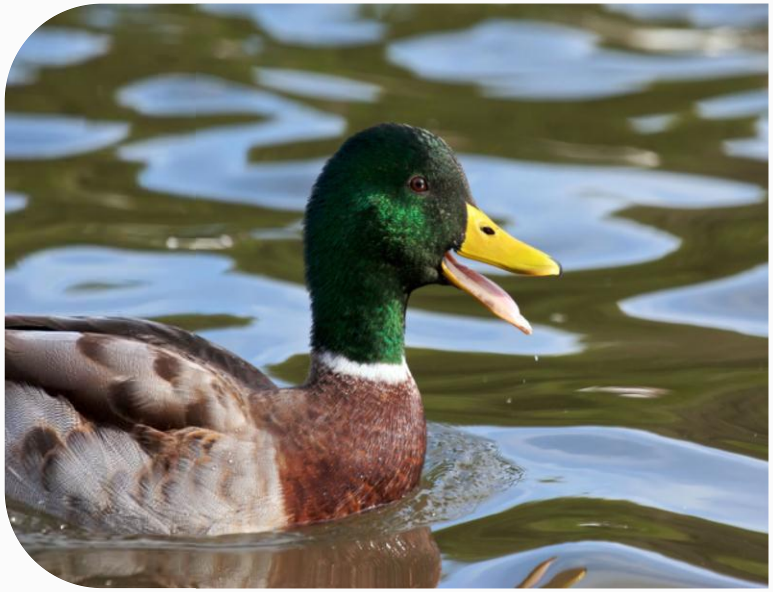
trait Duck */: Animal

def Quack -/: Animal +/++ Duck

$\phi(\text{Animal}) >= \phi(\text{Duck})$ [:]

$\phi(\text{Duck}) >= k^6 \times \phi(\text{Animal})$ [*]

Trait Rank



trait Duck */: Animal

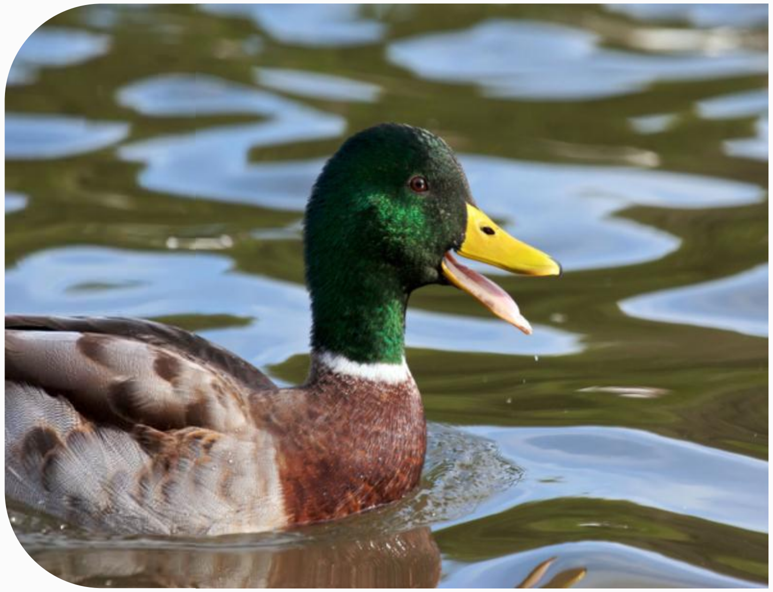
def Quack -/: Animal +/++ Duck

$\phi(\text{Quack}) >= k^9 \times \phi(\text{Animal})$ [-]

$\phi(\text{Quack}) >= k \times \phi(\text{Duck})$ [+]

$\phi(\text{Duck}) >= k^3 \times \phi(\text{Quack})$ [++]

Trait Rank



Chance of animal to quack: k^9

Duck: k

Animal already quacked once: k^4

$\phi(\text{Quack}) >= k^9 \times \phi(\text{Animal})$ [-]

$\phi(\text{Quack}) >= k^3 \times \phi(\text{Duck})$ [+]

$\phi(\text{Duck}) >= k \times \phi(\text{Quack})$ [++]

Trait Rank

Can work deeply; i.e., on members

```
Shape2d : Shape [Surface[Dim +/: 2D]]
```

Demo

YinYang (阴阳)
- Designed for tablets