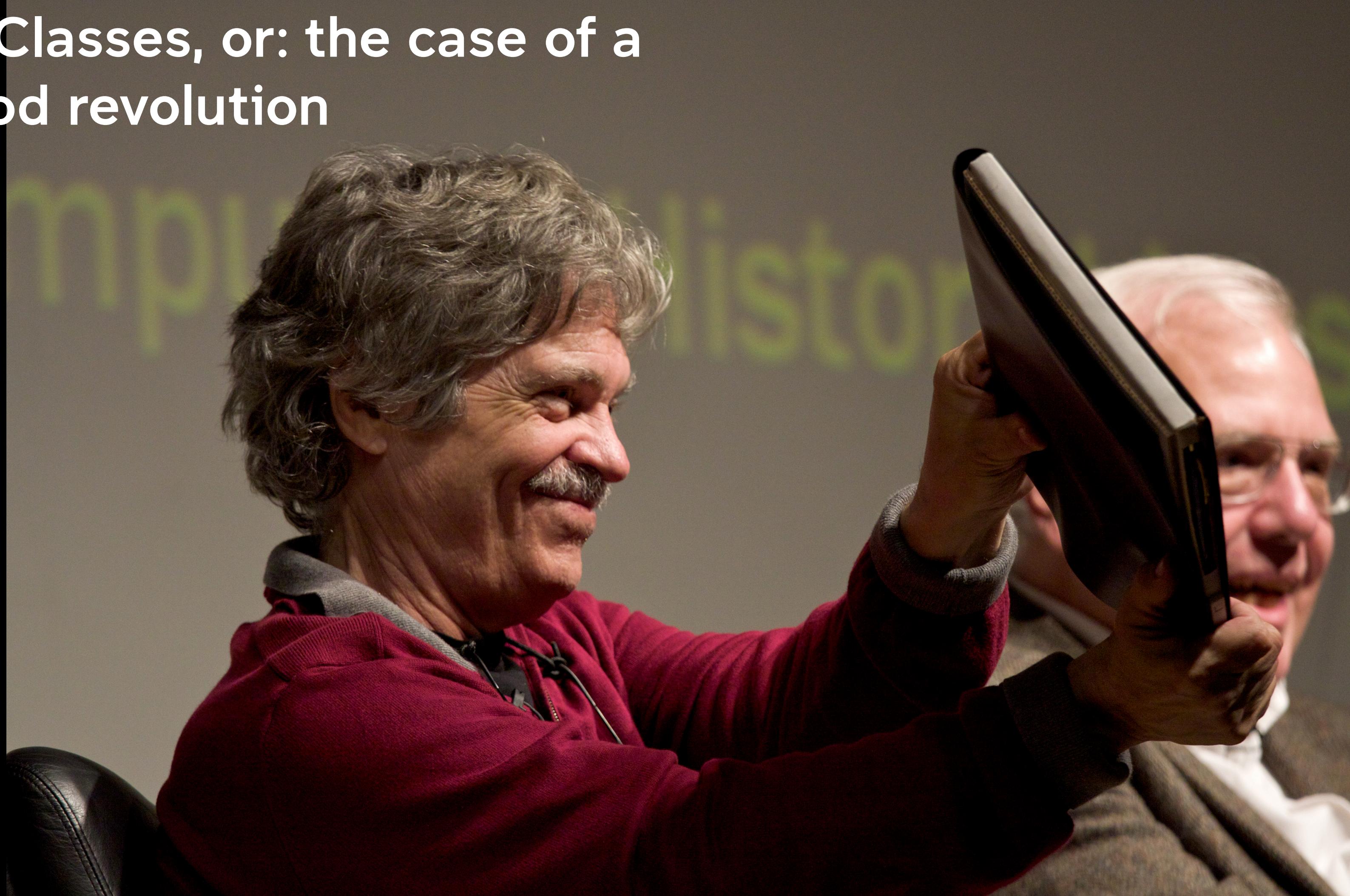


COMP1531

8.3 - Python Classes, or: the case of a
misunderstood revolution



UNSW
SYDNEY

1950s: the first
languages were
imperative

Imperative: Giving an authoritative command

this worked well
for many years....

And still does

```
#include <stdio.h>
/* Created a structure here. The name of the structure is
 * FarmAnimal.
 */
struct FarmAnimal
{
    char *breed;
    char *name;
    int age;
};
int main()
{
    /* animal is the variable of structure FarmAnimal*/
    struct FarmAnimal animal;

    /*Assigning the values of each struct member here*/
    animal.breed = "dog";
    animal.name = "Spot";
    animal.age = 7;

    /* Displaying the values of struct members */
    printf("animal breed is: %s\n", animal.breed);
    printf("animal Name is: %s\n", animal.name);
    printf("animal Age is: %d\n", animal.age);
    return 0;
}
```

So why OOP?

question: why do
we avoid globals?

as programs grew,
so did complexity

enter:
Alan Kay

**“I made up the term
‘object-oriented’, and
I can tell you I didn’t
have C++ in mind.” ~
Alan Kay, OOPSLA ‘97**



The big idea was to use encapsulated mini-computers in software which communicated via message passing rather than direct data sharing — to stop breaking down programs into separate “data structures” and “procedures”.

Alan Kay

1970s: Alan Kay's SmallTalk

SmallTalk: An OOP language

result := myObject getAge

result := myObject getAge

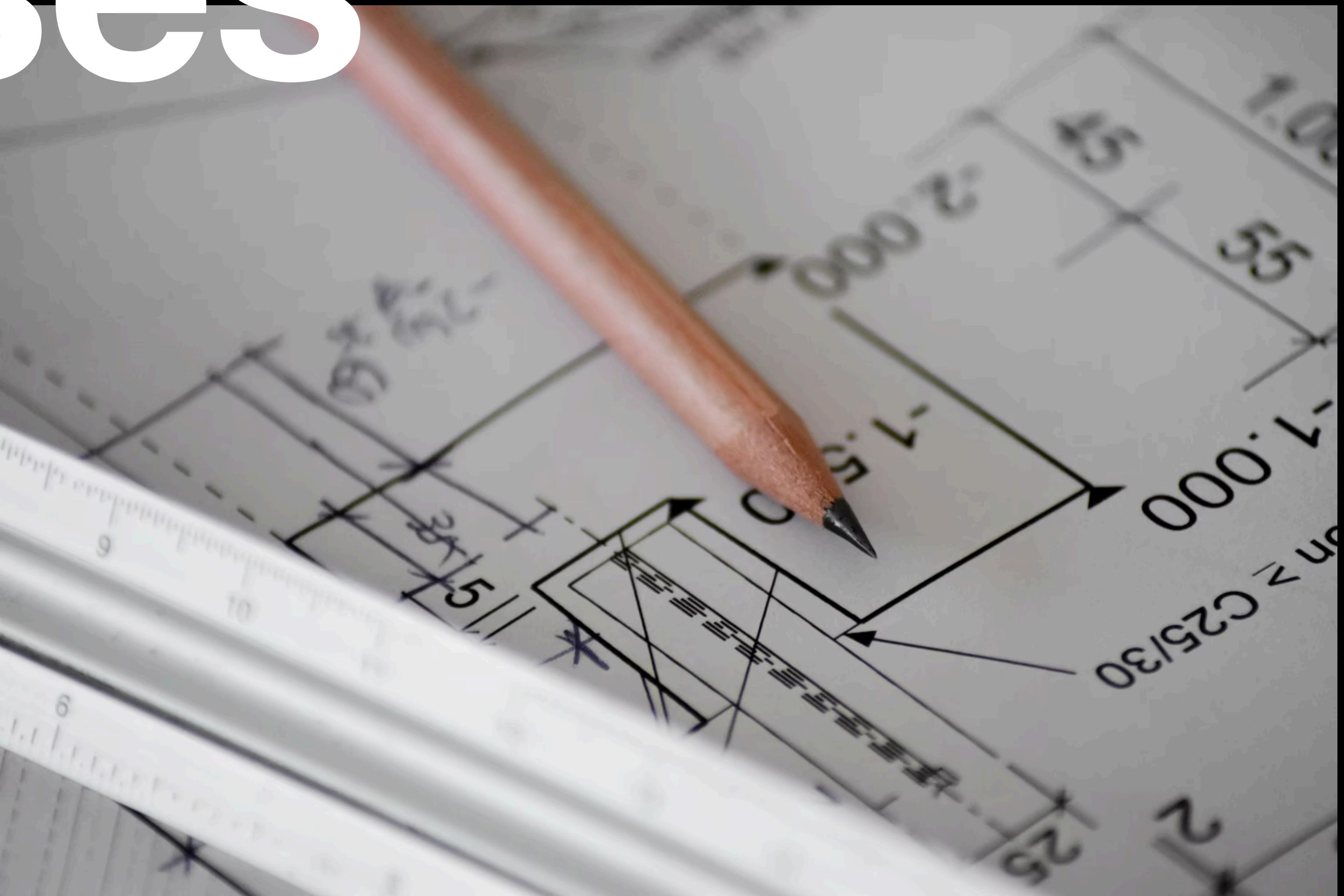
```
result := myObject getAge
```

```
result := myObject getAge
```

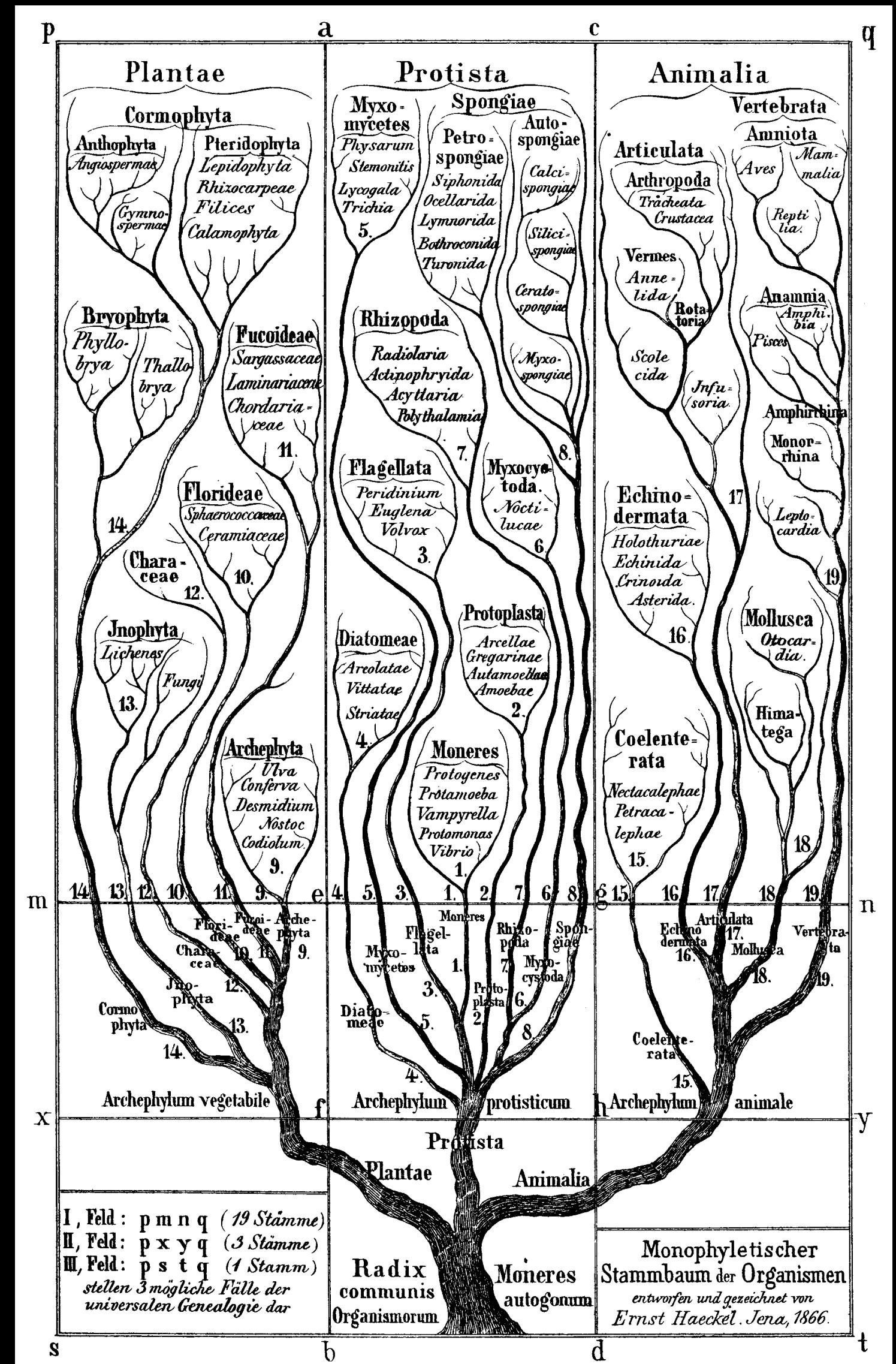
```
getAge:
    self hasAge ifTrue: [^self age].
```

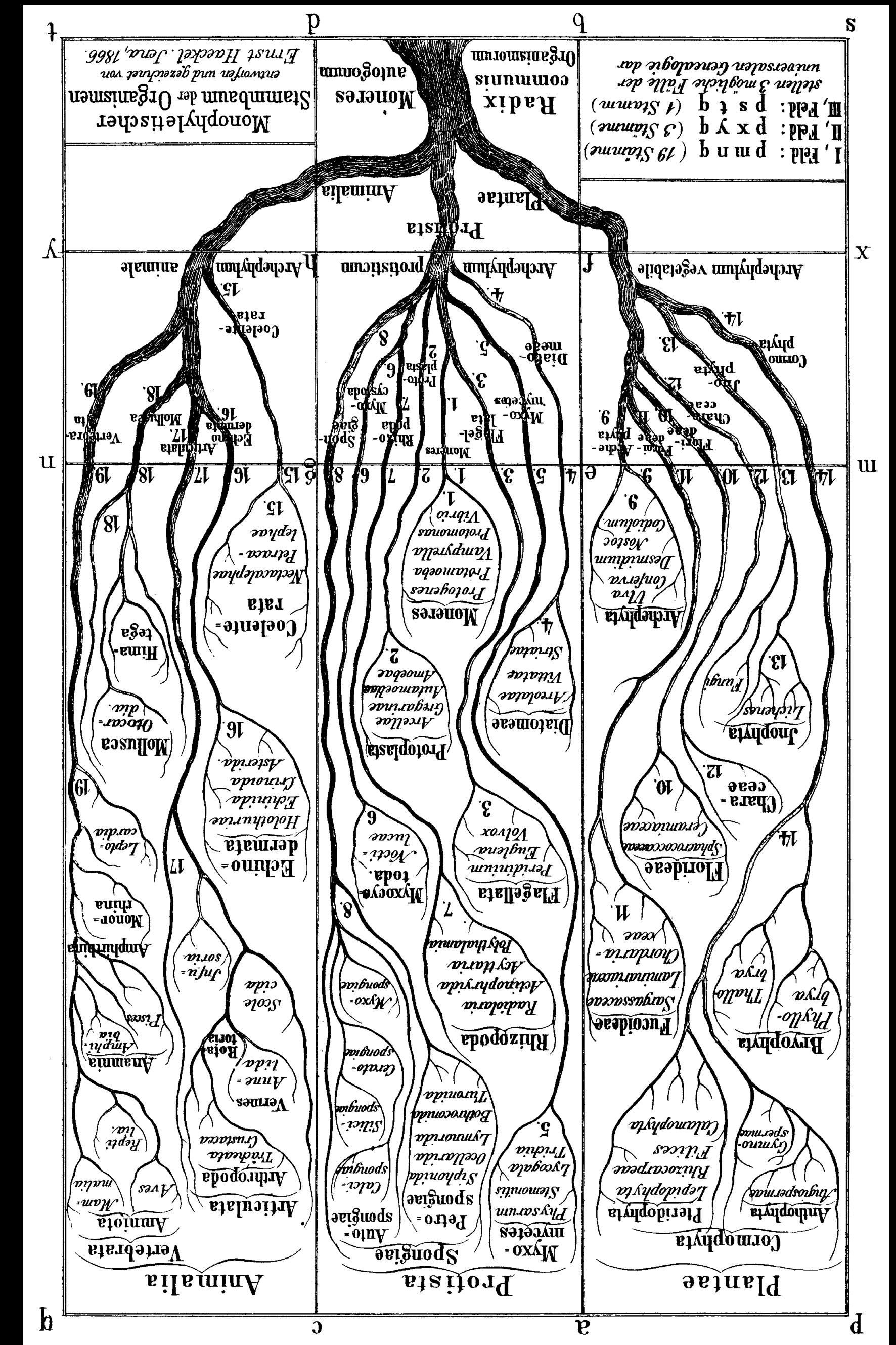
OOP Concepts

classes



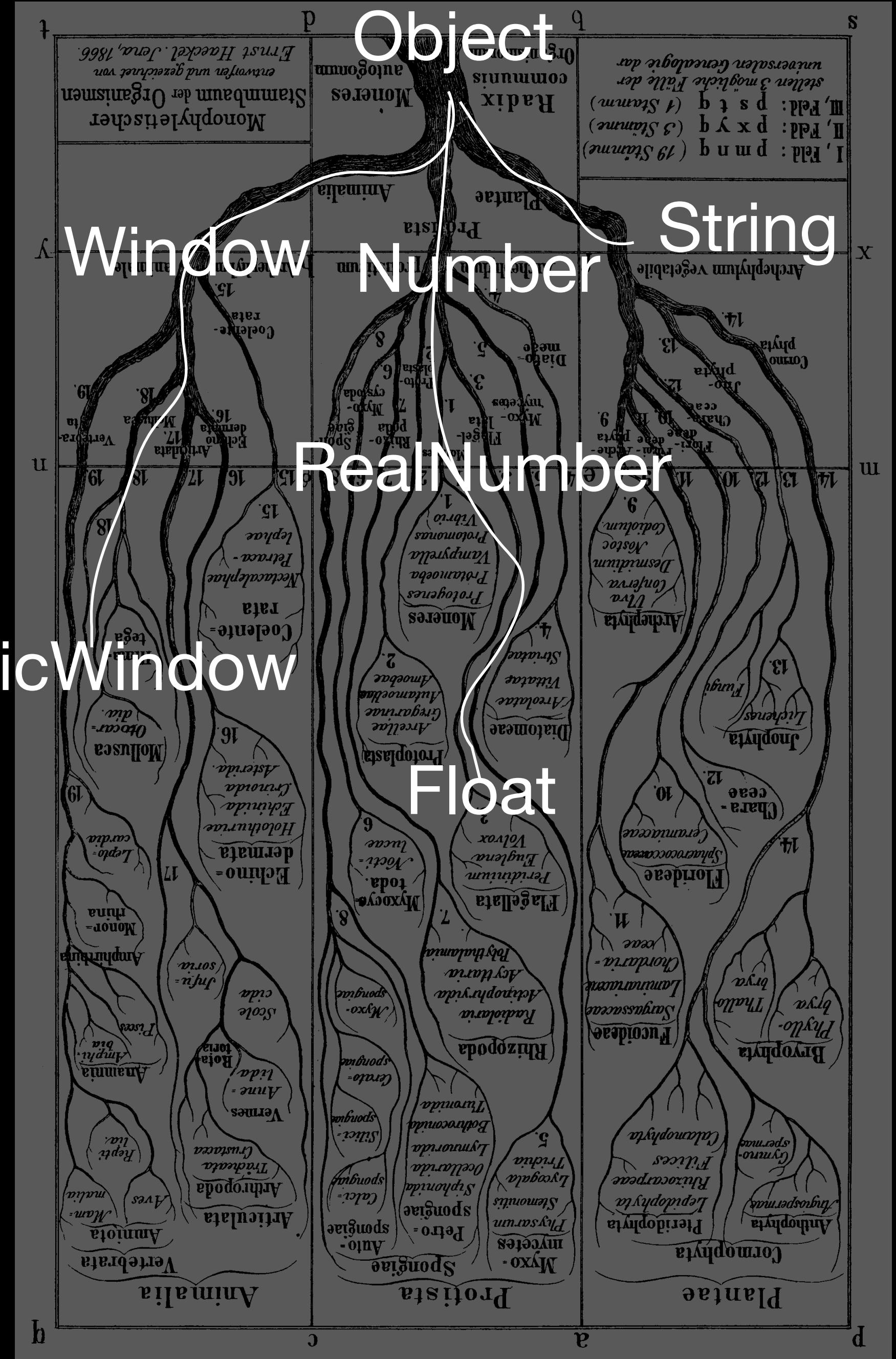
classes





GraphicWindow

Float



Objects



Objects

Objects know things,
and can do things



Demo