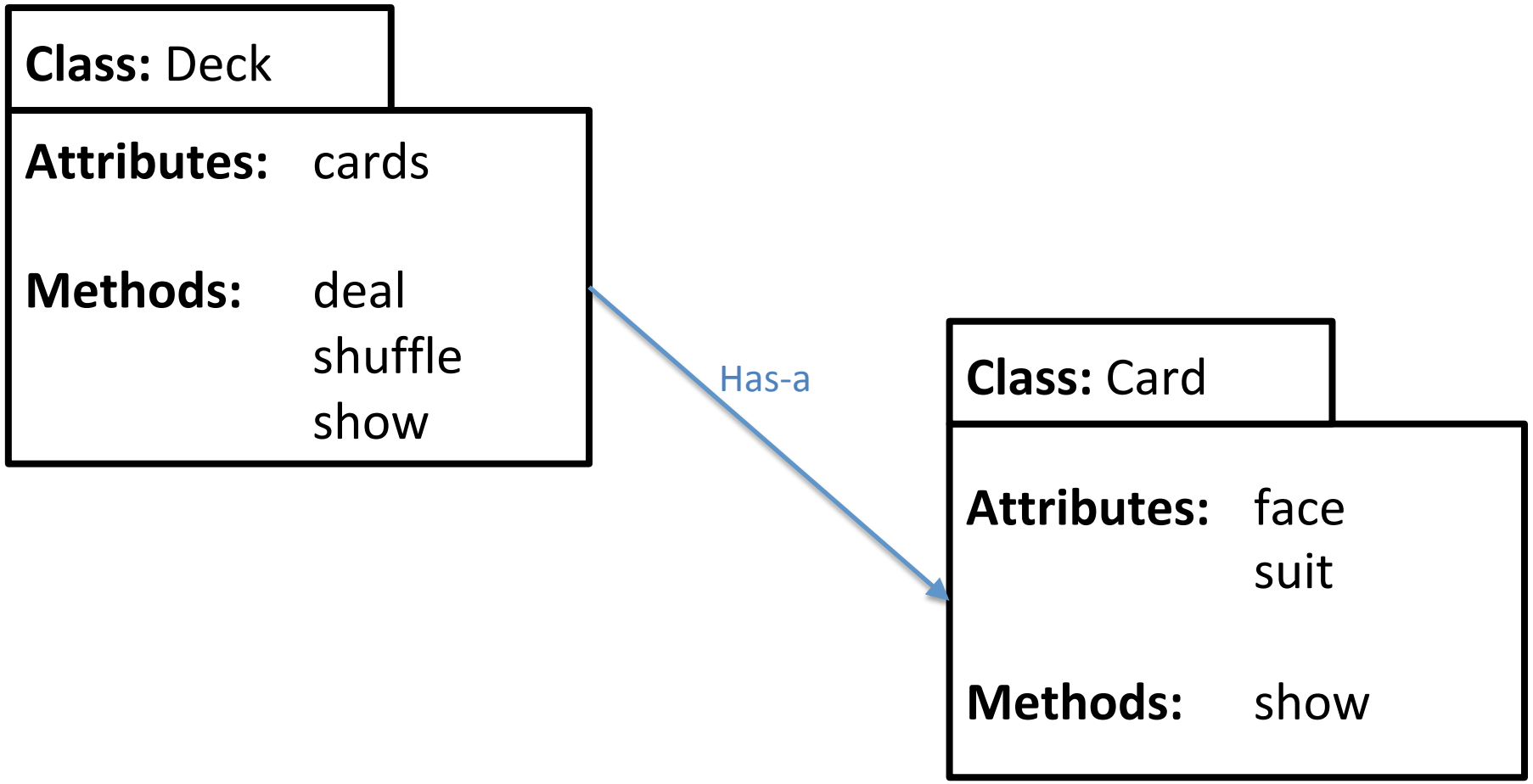
COMP10020
Introduction to Programming II
Combining Objects 2

Dr. Brian Mac Namee
brian.macnamee@ucd.ie
School of Computer Science
University College Dublin

A Simpler Composition Example



INHERITANCE

UCD
DUBLIN
an coláiste ollscoile baile átha cliath
UNIVERSITY COLLEGE DUBLIN

Class: Lecturer

Attributes:

- name
- age
- number
- courses
- office

Methods:

- addCourse
- removeCourse
- moveOffice
- show

Class: UGStudent

Attributes:

- name
- age
- number
- stage
- programme
- pathway

Methods:

- progress
- changeProg
- choosePath
- show

Class: PhDStudent

Attributes:

- name
- age
- number
- stage
- title

Methods:

- progress
- changeTitle
- show

Class: Lecturer

Attributes:

- **name**
- **age**
- **number**
- courses
- office

Methods:

- addCourse
- removeCourse
- moveOffice
- **show**

Class: UGStudent

Attributes:

- **name**
- **age**
- **number**
- stage
- programme
- pathway

Methods:

- progress
- changeProg
- choosePath
- **show**

Class: PhDStudent

Attributes:

- **name**
- **age**
- **number**
- stage
- title

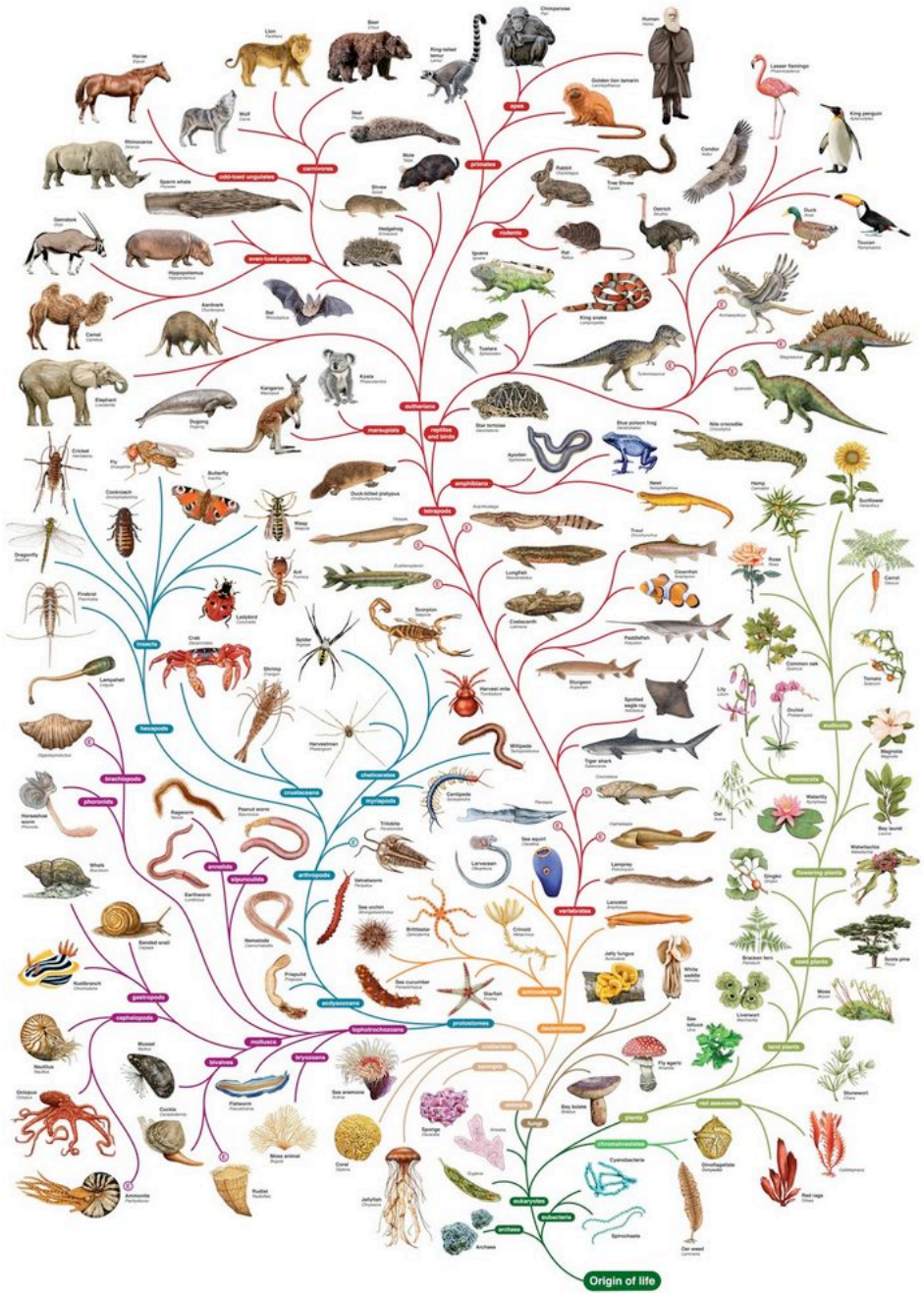
Methods:

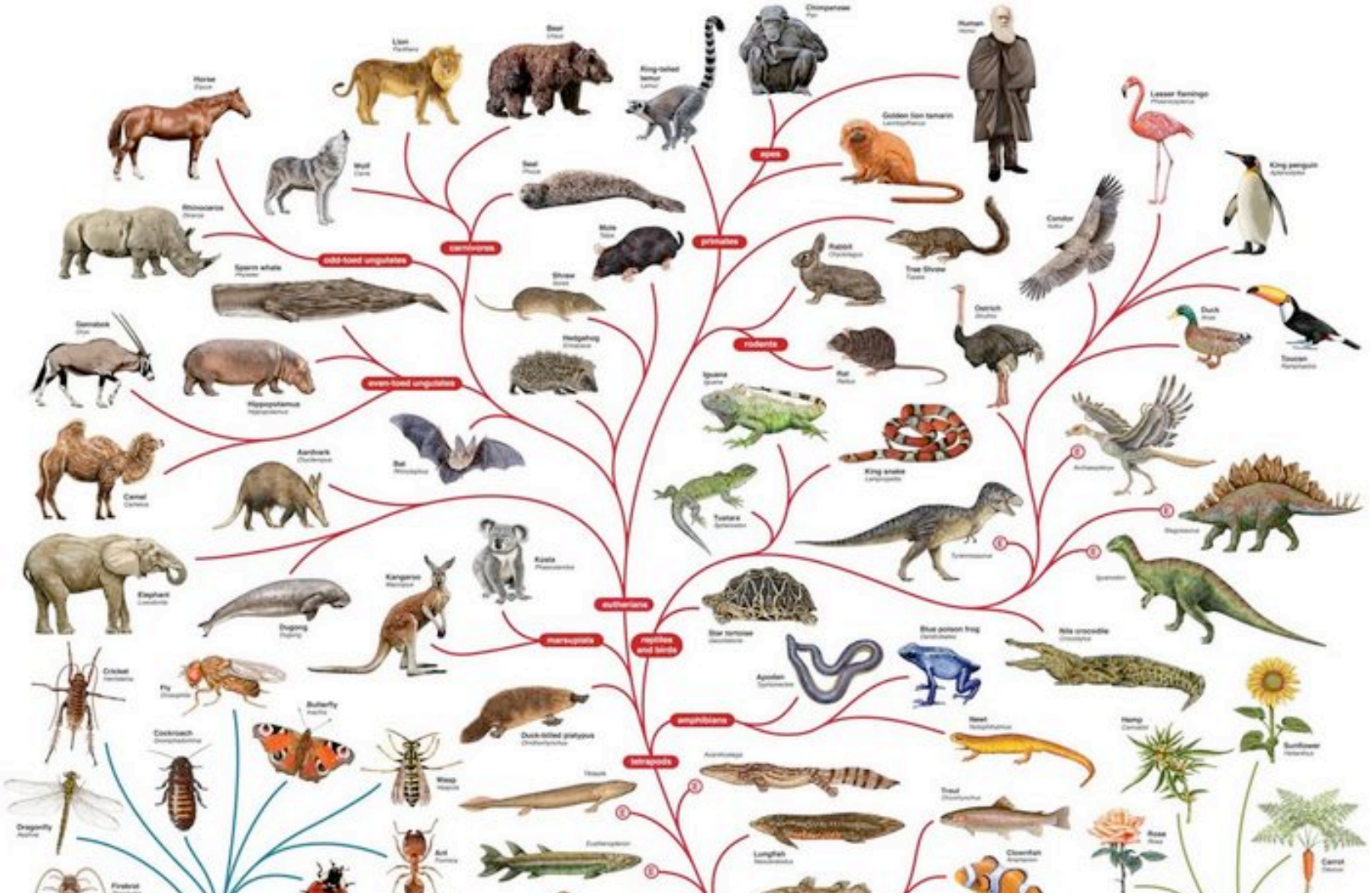
- progress
- changeTitle
- **show**

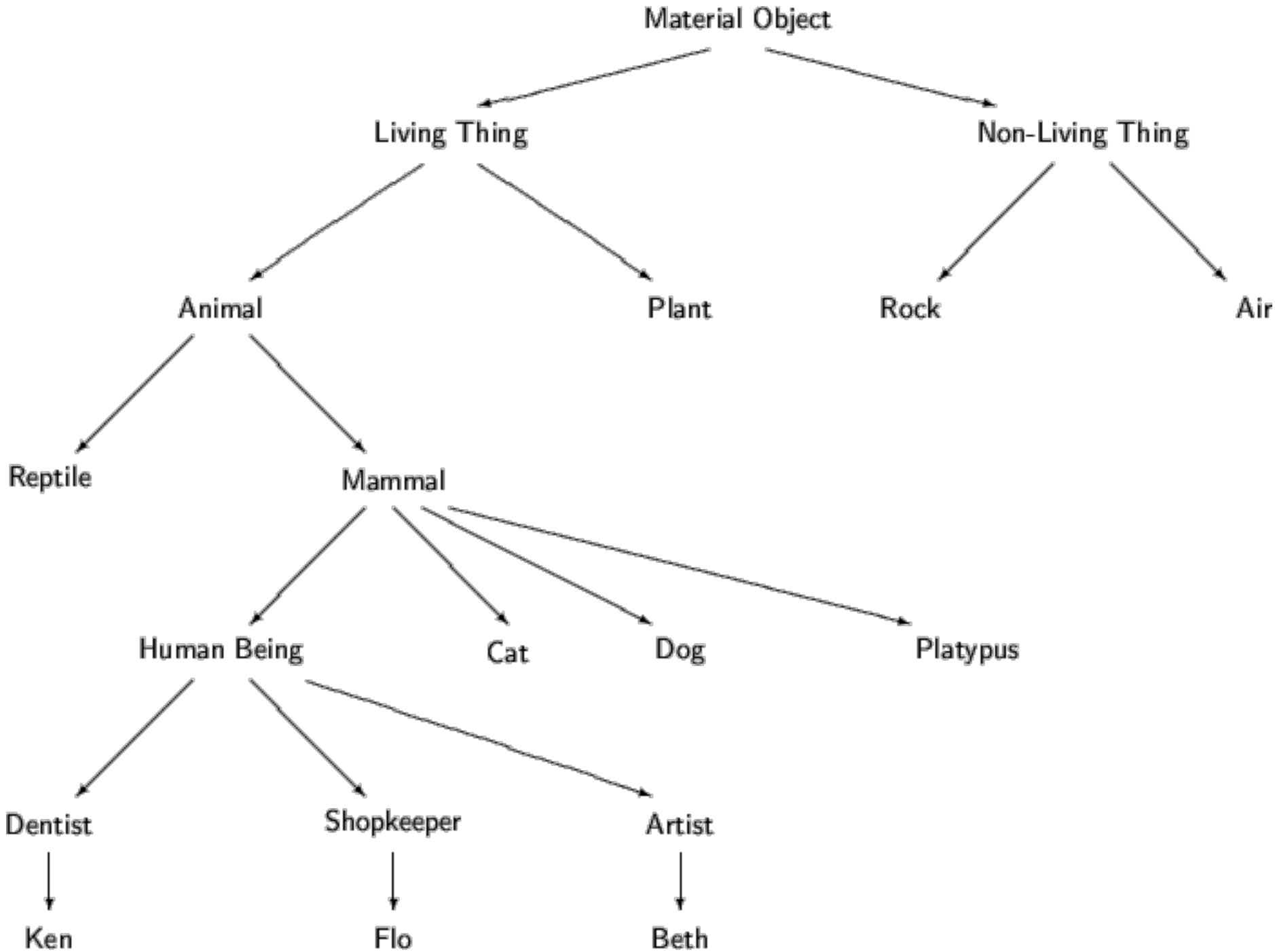
Key Ideas in OOP

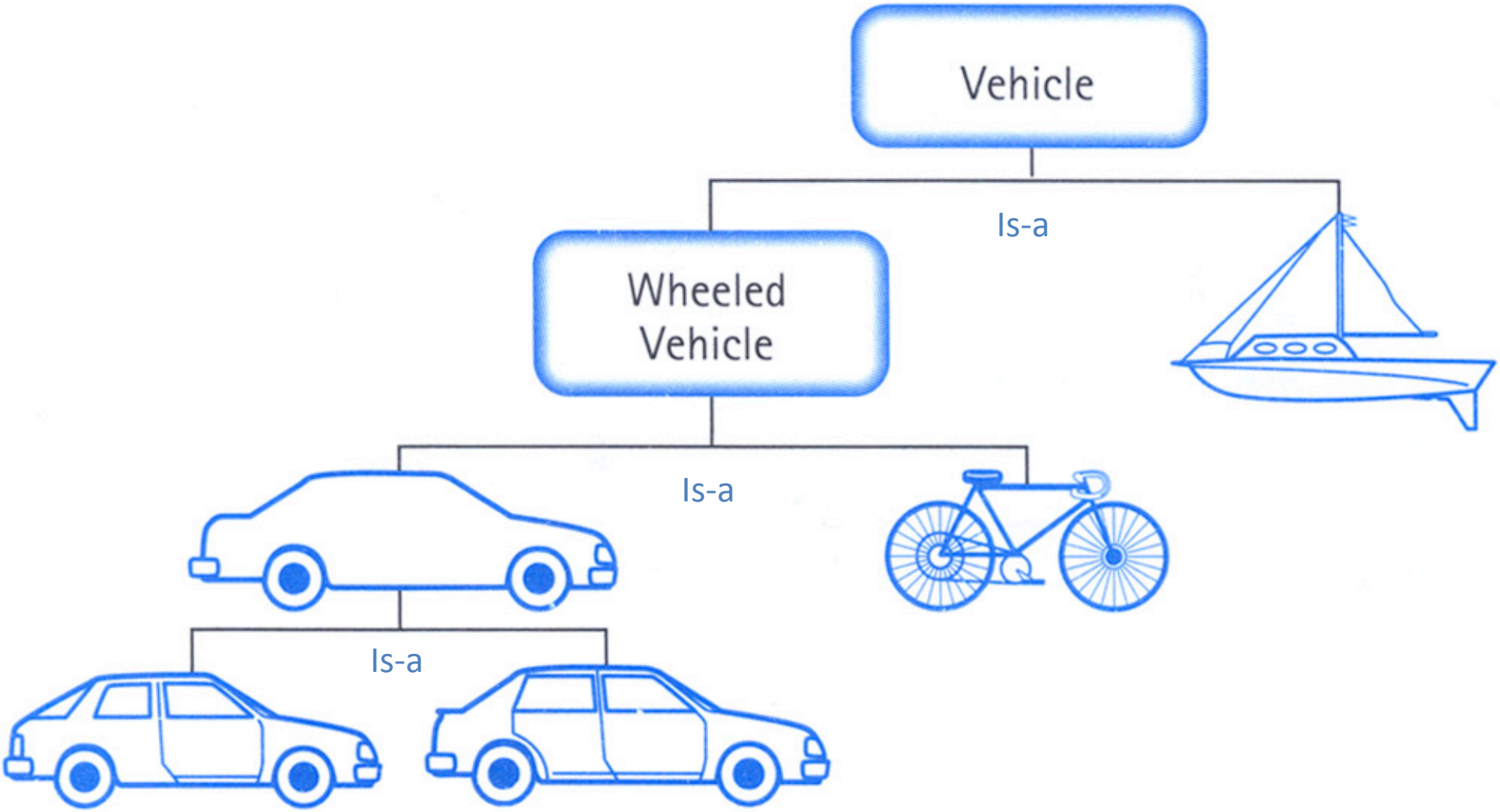
In diving deeper into OO programming we will look at the four major principles:

- Encapsulation
- Composition
- **Inheritance**
- Polymorphism


Horse
Lion
Bear
Ring-tailed lemur
Chimpanzee
Human
Lesser flamingo
Golden lion tamarin
King penguin
Wolf
Seal
Rhinoceros
Mole
Sperm whale
Shrew
Tree Shrew
Condor
Ostrich
Duck
Toucan
Hedgehog
Iguana
Ref
Gemsbok
Hippopotamus
King snake
Aardvark
Bat
Camel
Tuatara
Kiwi
Elephant
Kangaroo
Koala
Dugong
Star tortoise
Blue poison frog
Nile crocodile
Cricket
Fly
Butterfly
Cockroach
Wasp
Newt
Hemp
Sunflower
Dragonfly
Duck-billed platypus
Ant
Trout
Rose
Firefrat
Ladybird
Scorpion
Lungfish
Clownfish
Carrot
Crab
Spider
Coelacanth
Paddlefish
Lampshell
Shrimp
Harvest mite
Sturgeon
Spotted eagle ray
Common oak
Lily
Tomato
Oligochaetes
Harvestman
Millipede
Tiger shark
Magnolia
Horseshoe worm
Fagworm
Pencil worm
Centipede
Tribolite
Sea squirt
Waterlily
Bay laurel
Whelk
Earthworm
Larvacean
Lamprey
Lancelet
Sea urchin
Brittlestar
Crinoid
Sea cucumber
Starfish
Jelly fungus
White saddle
Bracken fern
Sitka pine
Sea lettuce
Liverwort
Moss
Fly agaric
Bay bolete
Sponge
Sea anemone
Coral
Jellyfish
Moss animal
Flatworm
Cuttle
Octopus
Nautilus
Ammonite
Rudist
Cyanobacteria
Spirochaete
Archaea
Red rage
Oar weed
Stonewort
Origin of life


Horse
Lion
Bear
King-tailed lemur
Chimpanzee
Human
Lesser flamingo
Golden lion tamarin
Wolf
Seal
King penguin
Rhinoceros
Condor
Sperm whale
Mole
Shrew
Rabbit
Tree Shrew
Ostrich
Duck
Toucan
Gemsbok
Hedgehog
Hippopotamus
Iguana
Rat
Bat
Aardvark
King snake
Camel
Tuatara
Kiwi
Elephant
Kangaroo
Koala
Dugong
Star tortoise
Blue poison frog
Nile crocodile
Cricket
Fly
Butterfly
Ayoolan
Cockroach
Duck-billed platypus
Newt
Hemp
Sunflower
Dragonfly
Wasp
Trout
Rose
Ant
Lungfish
Clownfish
Carrot
Firelord
odd-toed ungulates
carnivores
primates
apes
even-toed ungulates
rodents
eutherians
marsupials
reptiles and birds
amphibians
tetrapods


Material Object
Living Thing
Non-Living Thing
Animal
Plant
Rock
Air
Reptile
Mammal
Human Being
Cat
Dog
Platypus
Dentist
Shopkeeper
Artist
Ken
Flo
Beth


Vehicle
Is-a
Wheeled Vehicle
Is-a
Is-a

Class: Lecturer

Attributes:

- **name**
- **age**
- **number**
- courses
- office

Methods:

- addCourse
- removeCourse
- moveOffice
- **show**

Class: UGStudent

Attributes:

- **name**
- **age**
- **number**
- stage
- programme
- pathway

Methods:

- progress
- changeProg
- choosePath
- **show**

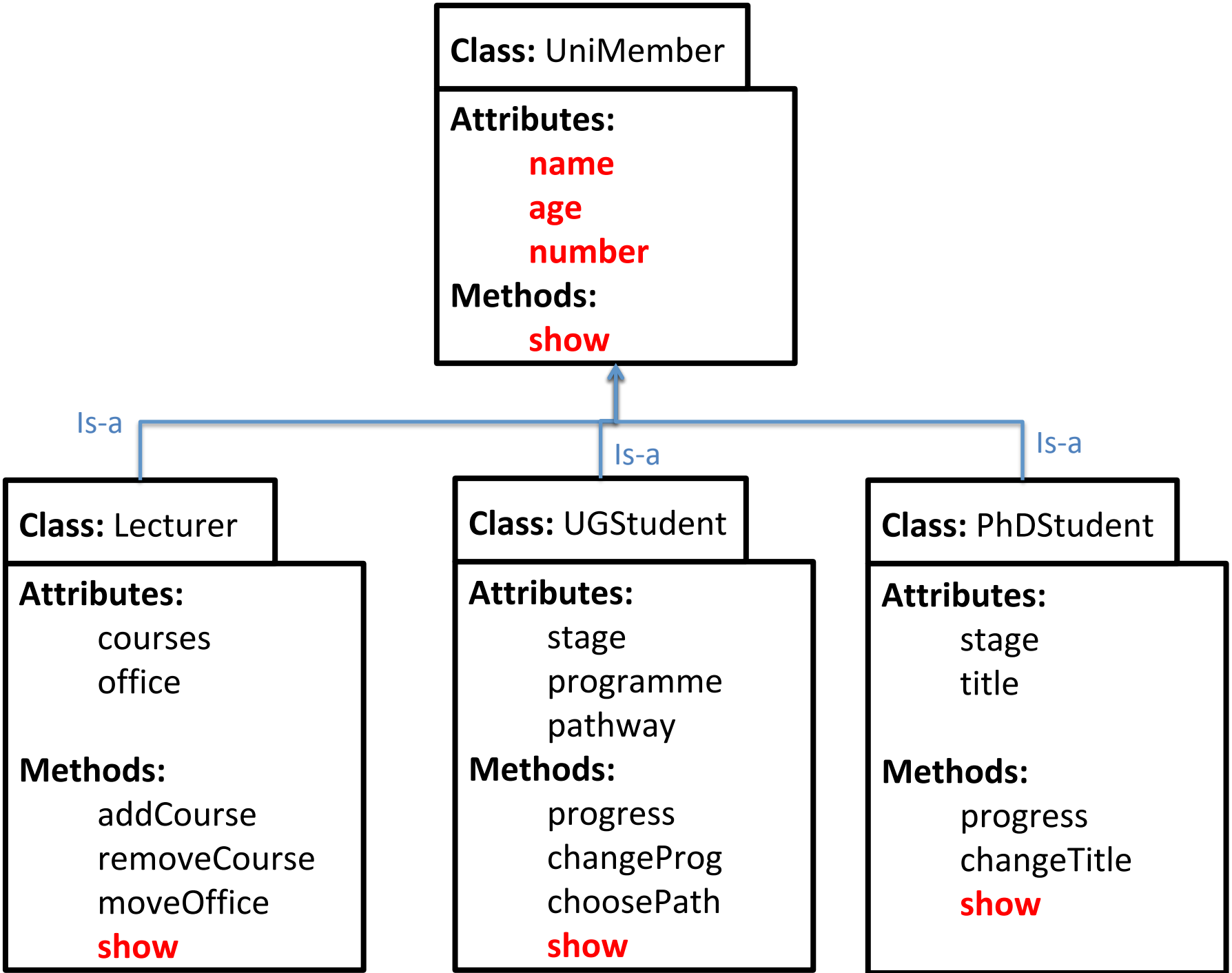
Class: PhDStudent

Attributes:

- **name**
- **age**
- **number**
- stage
- title

Methods:

- progress
- changeTitle
- **show**


Class: UniMember
Attributes:
name
age
number
Methods:
show
Is-a
Is-a
Is-a
Class: Lecturer
Attributes:
courses
office
Methods:
addCourse
removeCourse
moveOffice
show
Class: UGStudent
Attributes:
stage
programme
pathway
Methods:
progress
changeProg
choosePath
show
Class: PhDStudent
Attributes:
stage
title
Methods:
progress
changeTitle
show

Lecturer *inherits* from **UniMember**

Class: UniMember

Attributes:
- **name**
- **age**
- **number**

Methods:
- **show**

Is-a

Is-a

Is-a

Class: Lecturer

Attributes:
- courses
- office

Methods:
- addCourse
- removeCourse
- moveOffice
- **show**

Class: UGStudent

Attributes:
- stage
- programme
- pathway

Methods:
- progress
- changeProg
- choosePath
- **show**

Class: PhDStudent

Attributes:
- stage
- title

Methods:
- progress
- changeTitle
- **show**

UniMember
is a *parent*

Class: UniMember

Attributes:
- **name**
- **age**
- **number**

Methods:
- **show**

Is-a

Is-a

Is-a

Class: Lecturer

Attributes:
- courses
- office

Methods:
- addCourse
- removeCourse
- moveOffice
- **show**

Class: UGStudent

Attributes:
- stage
- programme
- pathway

Methods:
- progress
- changeProg
- choosePath
- **show**

Class: PhDStudent

Attributes:
- stage
- title

Methods:
- progress
- changeTitle
- **show**

Lecturer is a *child*

Class: UniMember

Attributes:
- **name**
- **age**
- **number**

Methods:
- **show**

Is-a

Is-a

Is-a

Class: Lecturer

Attributes:
- courses
- office

Methods:
- addCourse
- removeCourse
- moveOffice
- **show**

Class: UGStudent

Attributes:
- stage
- programme
- pathway

Methods:
- progress
- changeProg
- choosePath
- **show**

Class: PhDStudent

Attributes:
- stage
- title

Methods:
- progress
- changeTitle
- **show**

The child classes override the **show** method

Class: UniMember

Attributes:
- **name**
- **age**
- **number**

Methods:
- **show**

Is-a

Is-a

Is-a

Class: Lecturer

Attributes:
- courses
- office

Methods:
- addCourse
- removeCourse
- moveOffice
- **show**

Class: UGStudent

Attributes:
- stage
- programme
- pathway

Methods:
- progress
- changeProg
- choosePath
- **show**

Class: PhDStudent

Attributes:
- stage
- title

Methods:
- progress
- changeTitle
- **show**

Inheritance Syntax

```
class Parent:

    def __init__(self, attr1, attr2):
        self.attr1 = attr1
        self.attr2 = attr2



class Child(Parent):

    def __init__(self, attr1, attr2, attr3):
        Parent.__init__(self, attr1, attr2)
        self.attr3 = attr3
```

Inheritance Syntax

```
class Parent:

    def __init__(self, attr1, attr2):
        self.attr1 = attr1
        self.attr2 = attr2
```

Parent class is defined as normal - nothing special about it.

```
class Child(Parent):

    def __init__(self, attr1, attr2, attr3):
        Parent.__init__(self, attr1, attr2)
        self.attr3 = attr3
```

Inheritance Syntax

```
class Parent:

    def __init__(self, attr1, attr2):
        self.attr1 = attr1
        self.attr2 = attr2


class Child(Parent):

    def __init__(self, attr1, attr2, attr3):
        Parent.__init__(self, attr1, attr2)
        self.attr3 = attr3
```

To specify the parent-child relationship we use the parent class name in brackets after the child class definition

Inheritance Syntax

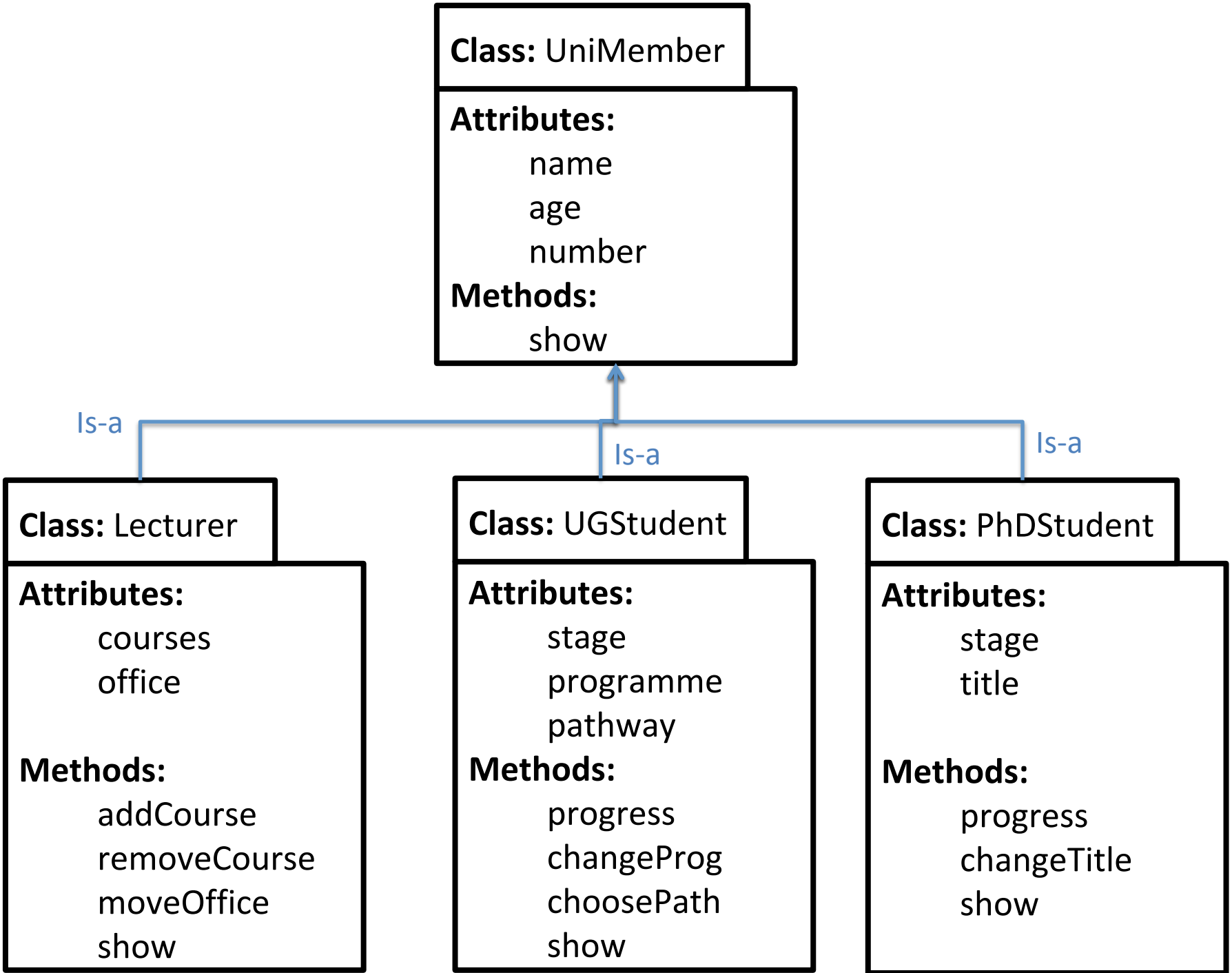
```
class Parent:

    def __init__(self, attr1, attr2):
        self.attr1 = attr1
        self.attr2 = attr2


class Child(Parent):

    def __init__(self, attr1, attr2, attr3):
        Parent.__init__(self, attr1, attr2)
        self.attr3 = attr3
```

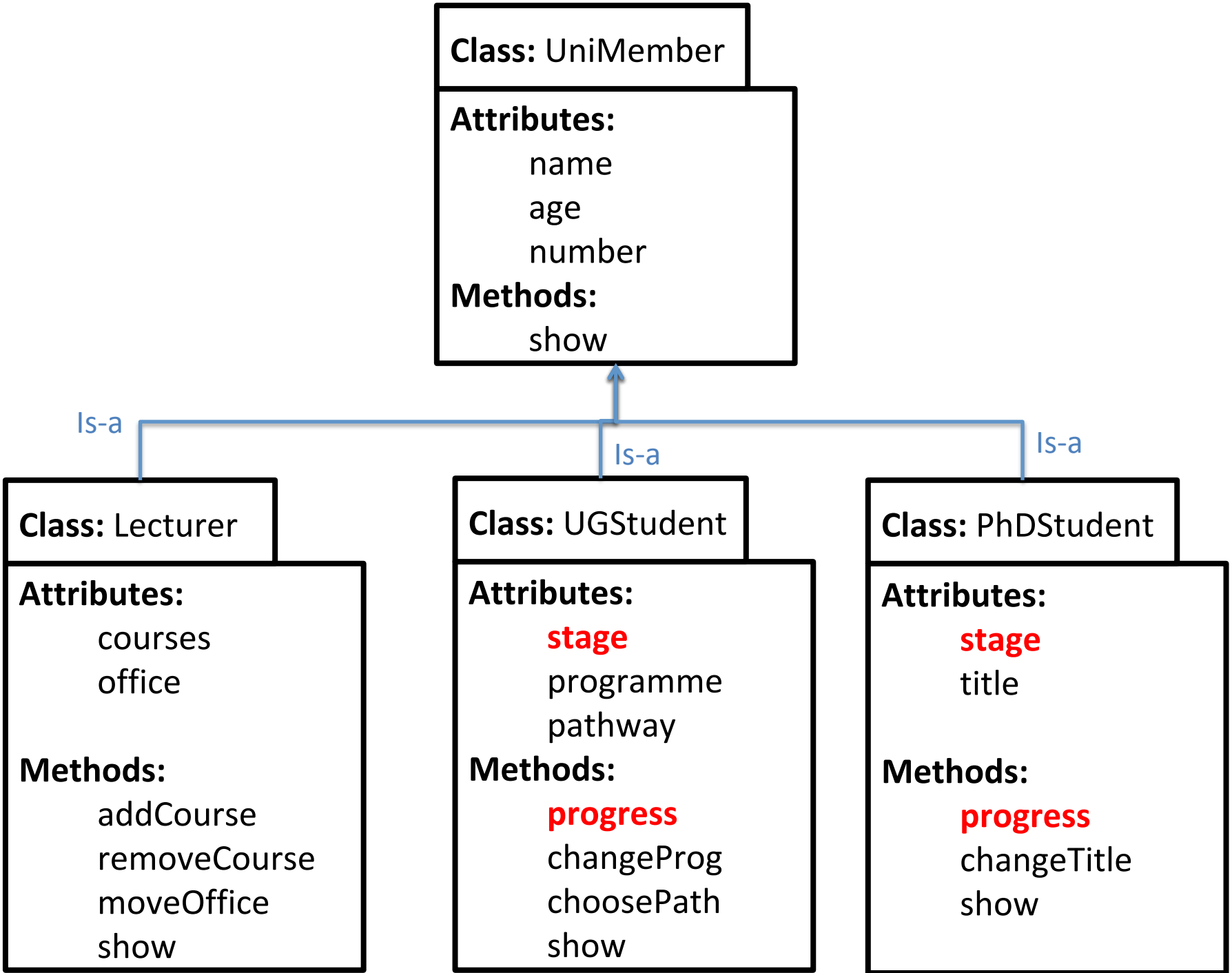
The child constructor takes the same parameters as the parent constructor, plus whatever extra is required

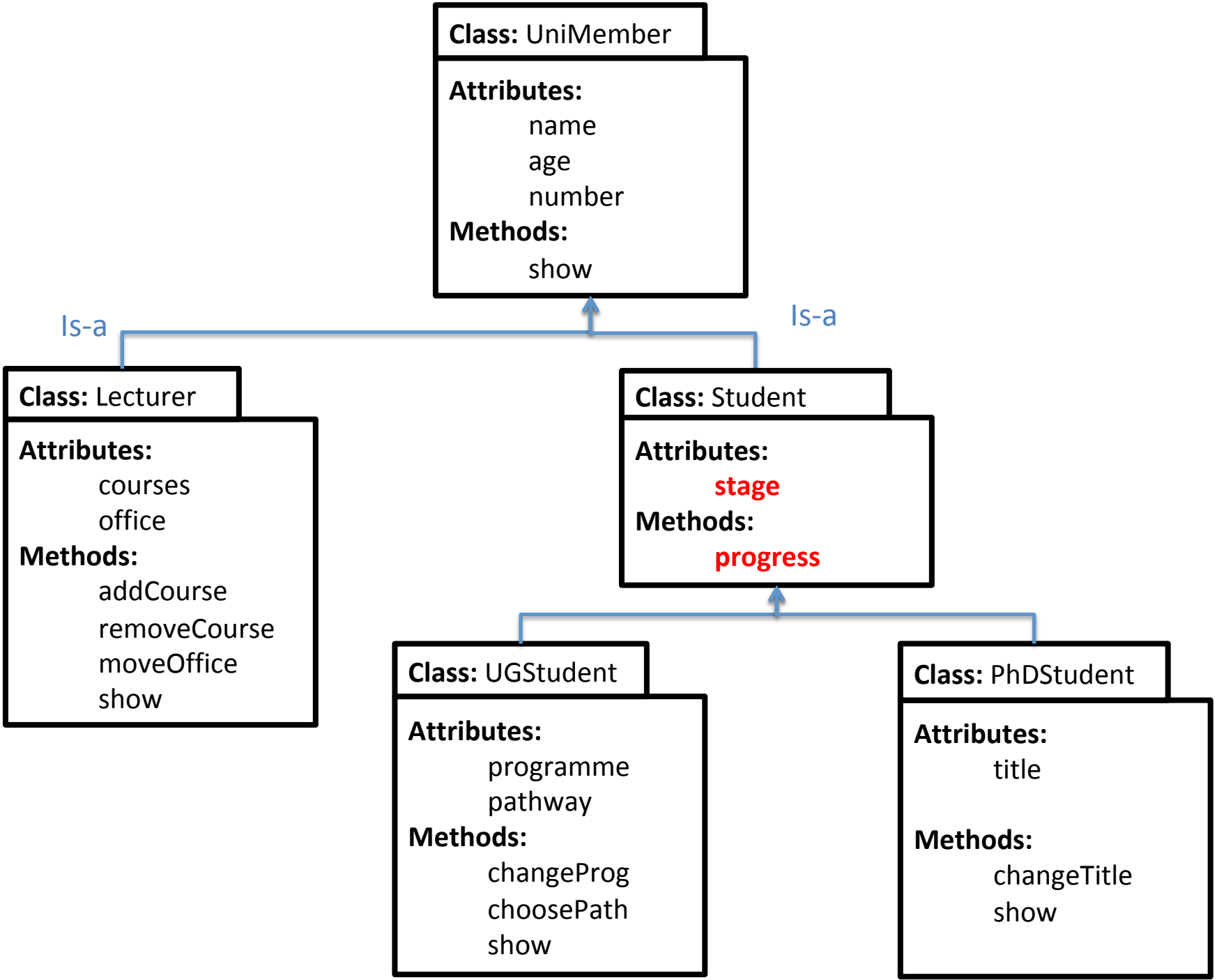
Inheritance Syntax

```
class Parent:

    def __init__(self, attr1, attr2):
        self.attr1 = attr1
        self.attr2 = attr2


class Child(Parent):

    def __init__(self, attr1, attr2, attr3):
        Parent.__init__(self, attr1, attr2)
        self.attr3 = attr3
```

We call the parent constructor to deal with the parent class attributes


Class: UniMember
Attributes:
name
age
number
Methods:
show
Is-a
Is-a
Is-a
Class: Lecturer
Attributes:
courses
office
Methods:
addCourse
removeCourse
moveOffice
show
Class: UGStudent
Attributes:
stage
programme
pathway
Methods:
progress
changeProg
choosePath
show
Class: PhDStudent
Attributes:
stage
title
Methods:
progress
changeTitle
show


Class: UniMember
Attributes:
name
age
number
Methods:
show
Is-a
Is-a
Is-a
Class: Lecturer
Attributes:
courses
office
Methods:
addCourse
removeCourse
moveOffice
show
Class: UGStudent
Attributes:
stage
programme
pathway
Methods:
progress
changeProg
choosePath
show
Class: PhDStudent
Attributes:
stage
title
Methods:
progress
changeTitle
show


Class: UniMember
Attributes:
name
age
number
Methods:
show
Is-a
Is-a
Class: Lecturer
Attributes:
courses
office
Methods:
addCourse
removeCourse
moveOffice
show
Class: Student
Attributes:
stage
Methods:
progress
Class: UGStudent
Attributes:
programme
pathway
Methods:
changeProg
choosePath
show
Class: PhDStudent
Attributes:
title
Methods:
changeTitle
show

Using Parent Methods Syntax

```
class Parent:

    def __init__(self, attr1, attr2):
        self.attr1 = attr1
        self.attr2 = attr2

    def show( )
        print(self.attr1 + " " + self.attr2)
```

Using Parent Methods Syntax

```
class Parent:

    def __init__(self, attr1, attr2):
        self.attr1 = attr1
        self.attr2 = attr2

    def show( )
        print(self.attr1 + " " + self.attr2)
```

Parent class with a **show** method

Inheritance Syntax

```
class Child(Parent):

    def __init__(self, attr1, attr2, attr3):
        Parent.__init__(self, attr1, attr2)
        self.attr3 = attr3

    def show( )
        print(self.attr1 + " " + self.attr2)
        print(self.attr3)
```

Inheritance Syntax

```
class Child(Parent):

    def __init__(self, attr1, attr2, attr3):
        Parent.__init__(self, attr1, attr2)
        self.attr3 = attr3

    def show( )
        print(self.attr1 + " " + self.attr2)
        print(self.attr3)
```

Child class overrides the parent class method with a new version

Inheritance Syntax

```
class Child(Parent):

    def __init__(self, attr1, attr2, attr3):
        Parent.__init__(self, attr1, attr2)
        self.attr3 = attr3

    def show( )
        Parent.show(self)
        print(self.attr3)
```

Inheritance Syntax

```
class Child(Parent):

    def __init__(self, attr1, attr2, attr3):
        Parent.__init__(self, attr1, attr2)
        self.attr3 = attr3

    def show( )
        Parent.show(self)
        print(self.attr3)
```

Use the **Parent** version of **show** instead of repeating this code

SUMMARY

Summary

In diving deeper into OO programming we will look at the four major principles:

- Encapsulation
- Composition
- **Inheritance** <- looked at today
- **Polymorphism** <- looked at today