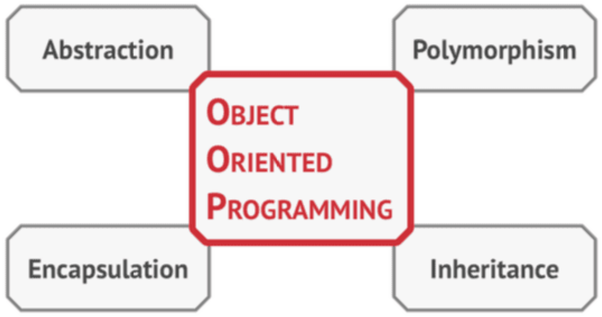
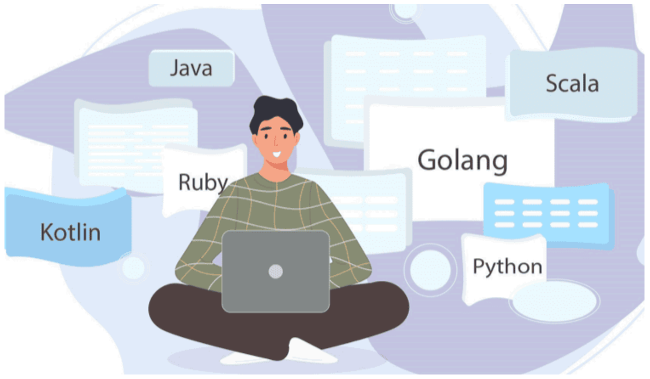
UML Diagrams

Structural Diagrams





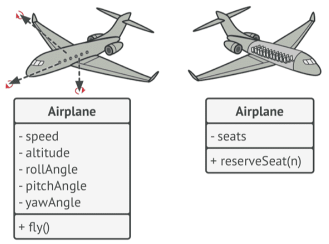
- APIE forms the foundation of the OOP paradigm.
- Any OOP language can implement the four core OOP concepts.

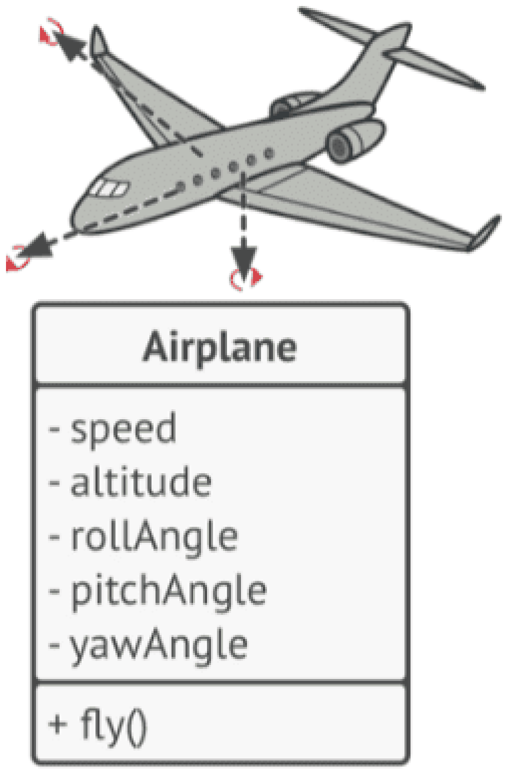
UML Diagrams

- UML, created for OOP modeling, is the standard language for software design.

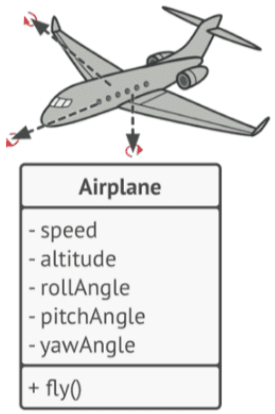
Class Diagram

- An airplane can be modeled as an object for flying or for making flight reservations.

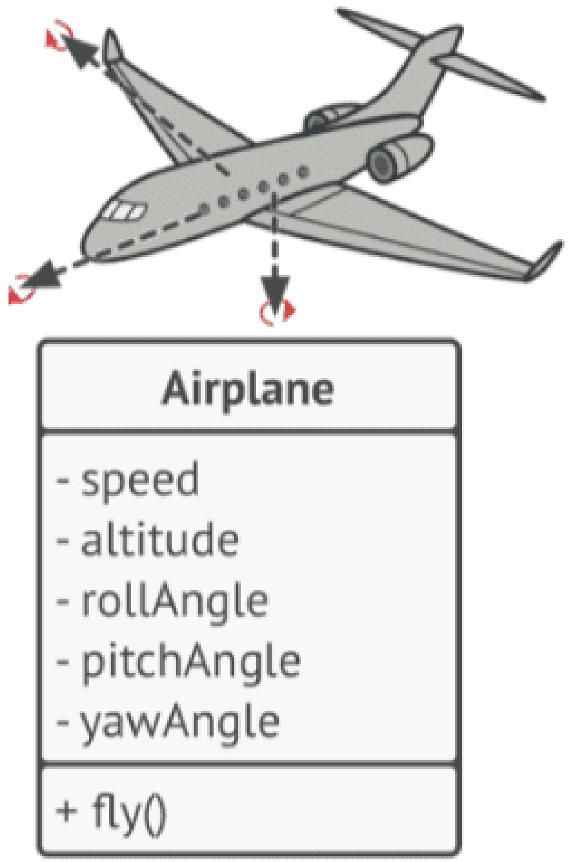




- A class UML diagram has three sections.
  - Class name.
  - Fields (states)
  - Methods (behaviors)



- In UML, - means private, + means public, and in typed languages you can specify types (e.g., -speed:int).



```python
class Airplane(object):
  def __init__(self):
    self.speed = 0
    self.altitude = 0
    self.rollAngle = 0
    self.pitchAngle = 0
    self.yawAngle = 0
  def fly(self):
    print("I'm flying")
```

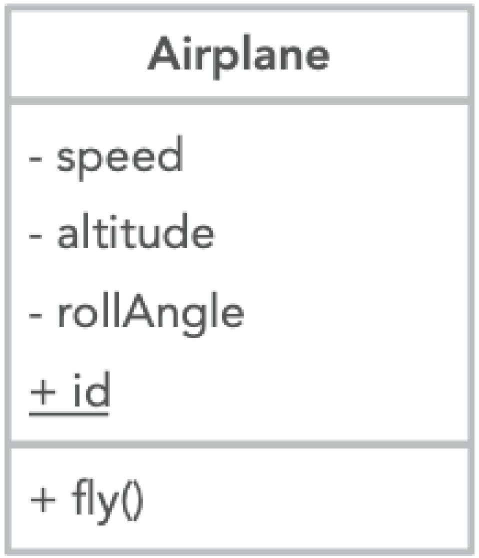
Static Fields

Variables declared in a class (but not a part of methods) are class (static) variables.

- They are shared by all objects.

```
class Airplane(object):
  id = 0
  ...
```

- They are underlined in UML.



```python
class Airplane(object):
  id = 0 # class variable
  def __init__(self):
    self.speed = 0
    ...

a = Airplane()
b = Airplane()
b.id = 10
print(a.id) # 10
```

Object Diagram

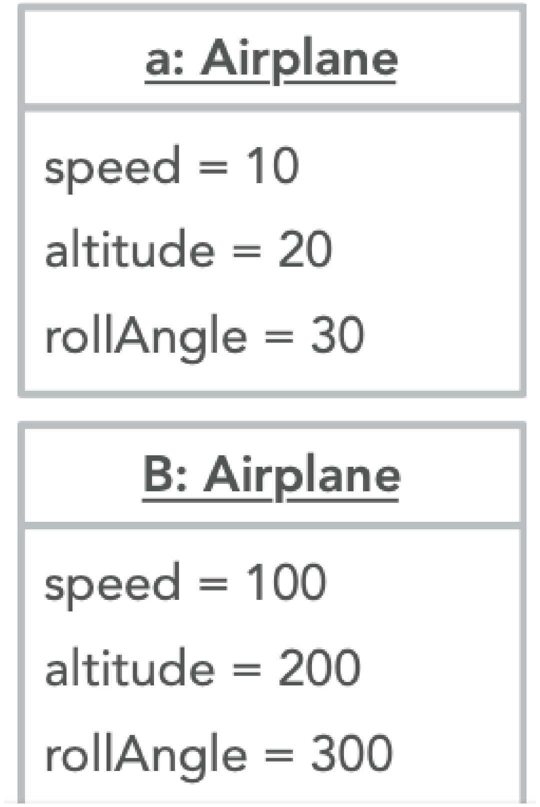
- Classes are blueprints; instantiating them creates objects.

```
class Airplane(object): # class
  ...
a = Airplane() # object instantiation
```

UML object diagrams have two sections:

1. Object name and class name (underlined)
2. Fields with assigned values

Python code



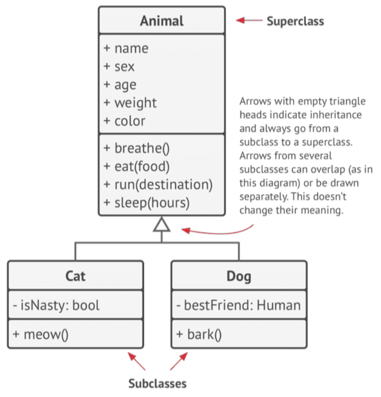
```python
a = Airplane(10, 20, 30)
b = Airplane(100, 200, 300)
```

Relationship among classes

- UML describes a single class or object using class and object diagrams.
- UML can describe the relationship among classes.

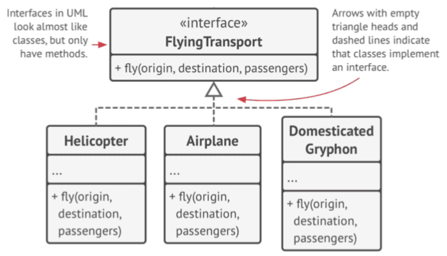
Inheritance

- In Python, extend a class by listing the superclass in parentheses
- In UML, show it with an arrow from subclass to superclass.


Animal
+ name
+ sex
+ age
+ weight
+ color
+ breathe()
+ eat(food)
+ run(destination)
+ sleep(hours)
Superclass
Arrows with empty triangle heads indicate inheritance and always go from a subclass to a superclass. Arrows from several subclasses can overlap (as in this diagram) or be drawn separately. This doesn't change their meaning.
Cat
- isNasty: bool
+ meow()
Dog
- bestFriend: Human
+ bark()
Subclasses

Interface

- In UML, we use a dotted line to indicate the implementation of an interface.


Interfaces in UML look almost like classes, but only have methods.
«interface»
FlyingTransport
+ fly(origin, destination, passengers)
Arrows with empty triangle heads and dashed lines indicate that classes implement an interface.
Helicopter
...
+ fly(origin, destination, passengers)
Airplane
...
+ fly(origin, destination, passengers)
Domesticated Gryphon
...
+ fly(origin, destination, passengers)

Association

It’s a structural relationship linking objects of different classes.

- Composition
- Aggregation
- Dependencies

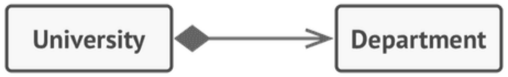
Composition

- Object A is composed of other objects (B and C): in other words A = B + C.
  - We call this relationship composition.
  - We use a solid diamond to express composition in UML.

```python
class University(object):
  def __init__(self):
    self.departments = []

class Department(object):
  def __init__(self, name):
    self.name = name

d1 = Department("cs")
d2 = Department("ase")
u = University()
u.departments.append(d1)
u.departments.append(d2)
```



- We cannot think of a university without departments.

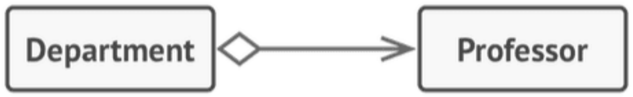
Aggregation

- Aggregation implies ownership.
  - A department can hire (own) or fire (disown) a professor.
  - We use an empty diamond to express composition in UML.

```python
class University(object):
  def __init__(self):
    self.professors = []

class Professor(object):
  def __init__(self, name):
    self.name = name

p = Professor("Dr. Cho")
u = University("NKU")
u.professors.append(p)
```



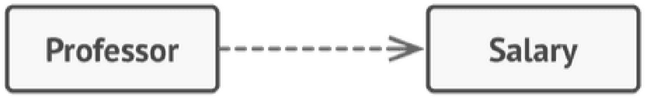
- It is possible that a university does not have a certain professor.
- There is no difference in Python coding.

Dependencies

- Dependency implies knowledge of other objects.
    - An object uses another object as an argument.
    - To show a weaker relationship, a dotted line is used.

```python
class Salary(object):
  def __init__(self, amount):
    self.amount = amount

s = Salary(1000)
p = Professor('Dr. Cho')
p.getSalary(s)
```



- The Professor object gets the Salary object as an argument.
- This is (much) weaker relationship compared to composition and aggregation.