Part 3:

Object Oriented Analysis

INHERITANCE

Inheritance

- A mechanism for a modular and hierarchical organization is **inheritance**
- This allows a new class to be defined based upon an existing class as the starting point
- The existing class is typically described as the **base class**, parent class, or superclass, while the newly defined class is known as the **subclass** or child class
- There are two ways in which a subclass can differentiate itself from its superclass:
  - A subclass may specialize an existing behavior by providing a new implementation that overrides an existing method.
  - A subclass may also extend its superclass by providing brand new methods

Very Simple Example

Base Class

```
class A:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def foo(self):
        return self.x + self.y + 1

    def bar(self):
        self.x += 1
        self.y += 1
```

Super Class

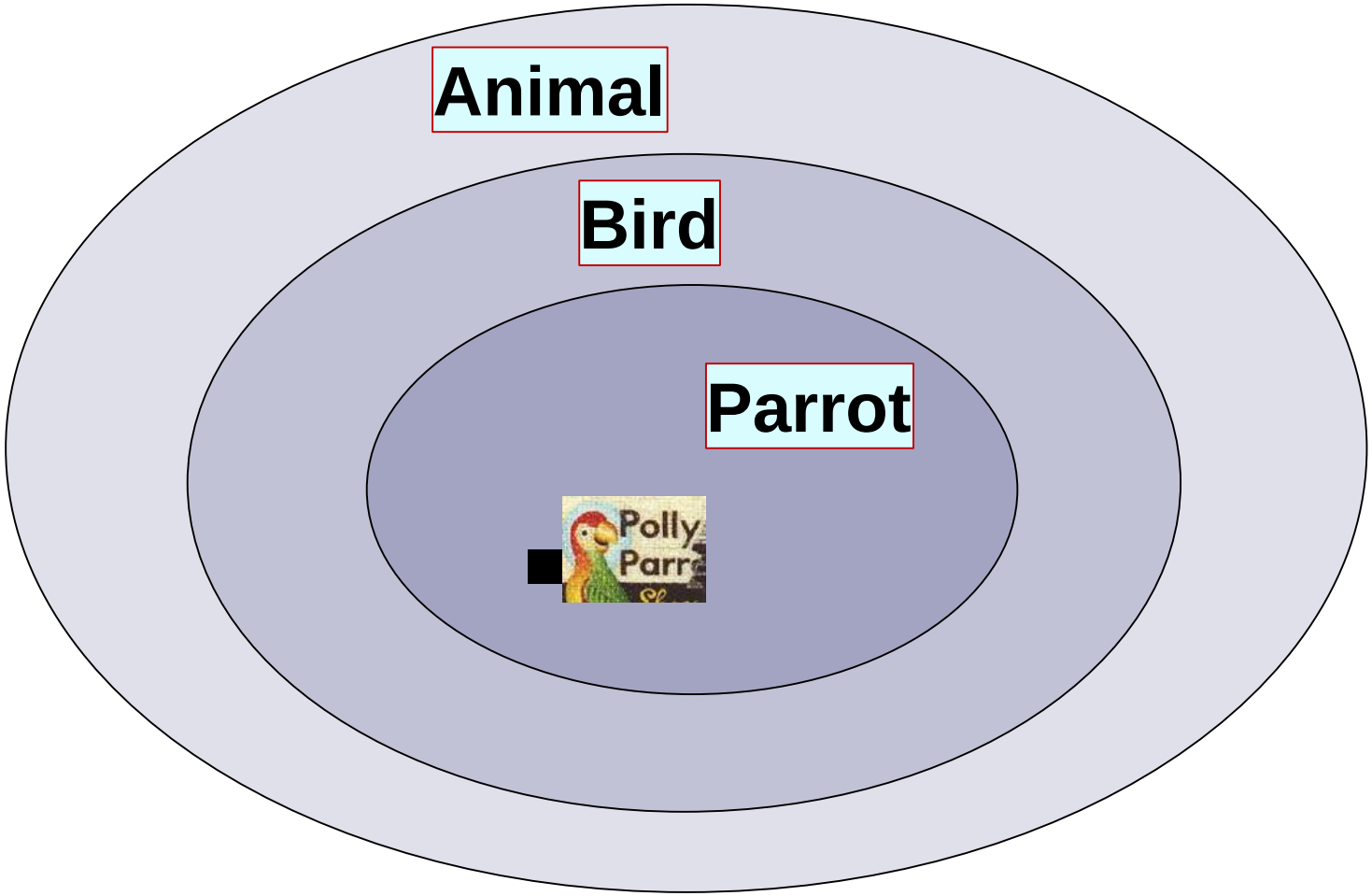
```
class B(A):
    def __init__(self, x, y, z):
        A.__init__(self, x, y)
        self.z = z

    def foo(self):
        return self.x + self.y + 2

    def move(self, m, n):
        self.x += m
        self.y += n
```

Animals, Birds, and Parrots

```
class Animal:
    "Our animal object"
    def __init__(self, name, id):
        self.name = name
        self.id = id

    def speak(self):
        print "My name is:", self.name
        print "My ID is:", self.id

    def eat(self, food):
        print "I am eating:", food

class Bird(Animal):
    "Our bird object"
    def __init__(self, name, id, gender):
        Animal.__init__(self, name, id)
        self.gender = gender

    def speak(self):
        Animal.speak(self)
        print "My gender is:", self.gender

    def sing(self, song):
        print "I am singing:", song
```

```
class Parrot(Bird):
    "Our parrot object"
    def __init__(self, name, id, gender, color):
        Bird.__init__(self, name, id, gender)
        self.color = color

    def speak(self):
        Bird.speak(self)
        print "My color is:", self.color

if __name__ == "__main__":
    polly = Parrot("Polly", 17, "male", "yellow")
    polly.speak()
    polly.sing("Haleluya")
    polly.eat("sesame ice cream")
```

Taxonomy



Multiple Inheritance

```
class Fly:
    "Flyer Class"
    def fly(self):
        print "I am flying!"

class Swim:
    "Swimmer Class"
    def swim(self):
        print "I am swimming!"

class Animal:
    "Our animal object"
    def __init__(self, name, id):
        self.name = name
        self.id = id

    def speak(self):
        print "My name is:", self.name
        print "My ID is:", self.id

    def eat(self, food):
        print "I am eating:", food
```

```
class Duck(Animal,Fly,Swim):
    def __init__(self, name, id):
        Animal.__init__(self, name, id)

    def quack(self, x):
        print "I am quacking:", x

if __name__ == "__main__":
    duffy = Duck("Daffy", 1946)
    duffy.speak()
    duffy.fly()
    duffy.swim()
    duffy.eat("corn")
    duffy.quack("Ouch...Ouch...")
    print '------------------------'
    don = Duck("Donald", 1947)
    don.speak()
    don.fly()
    don.swim()
    don.eat("fish")
    don.quack("Woooppsss ...")
```

Multiple Inheritance: Usage

```
duffy = Duck("Daffy", 1946)
duffy.speak()
duffy.fly()
duffy.swim()
duffy.eat("corn")
duffy.quack("Ouch...Ouch...")
print '-------------------------'
don = Duck("Donald", 1947)
don.speak()
don.fly()
don.swim()
don.eat("fish")
don.quack("Wooooppsss ...")
```

```
D:\BRAUDE\OOP\CODE> multi_inher.py
My name is: Daffy
My ID is: 1946
I am flying!
I am swimming!
I am eating: corn
I am quacking: Ouch...Ouch...
-------------------------
My name is: Donald
My ID is: 1947
I am flying!
I am swimming!
I am eating: fish
I am quacking: Wooooppsss ...
```

Loading Classes / Code Reuse

```
from credit_card import CreditCard

class PlatinumCard(CreditCard):
    def __init__(self, customer, bank, acnt, limit):
        CreditCard.__init__(self, customer, bank, acnt, limit)
        self._balance = 10000
```

```
c = PlatinumCard('John Goodman', 'Bank of America', '5295 7271 4727 3362', 200000)
c.charge(1000)
c.charge(1500)
c.charge(2500)
c.make_payment(3000)
print 'Customer =', c.get_customer()
print 'Bank =', c.get_bank()
print 'Account =', c.get_account()
print 'Limit =', c.get_limit()
print 'Balance =', c.get_balance()
```