상속 (Inheritance)

- 강아지에 더해서 고양이도 정의해보자
- 강아지와 고양이에 공통된 attributes와 methods를 묶어서 Pet으로 정의하면?

Attributes

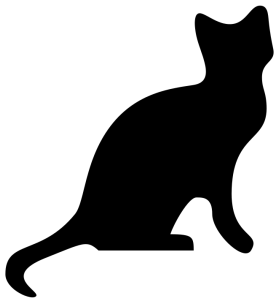
- 이름
- 나이
- 털색
- 견종

Methods

- 달려
- 짖어
- 물어

Attributes

- 이름
- 나이
- 털색
- 묘종



Methods

- 달려
- 울어
- 점프

상속 (Inheritance)

Attributes

- 이름
- 나이
- 털색

Methods

- 달려

Pet

Inheritance

Inheritance

Attributes

- 견종

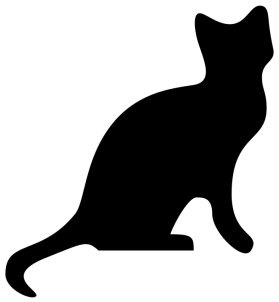
Methods

- 짖어
- 물어

Dog

Attributes

- 묘종



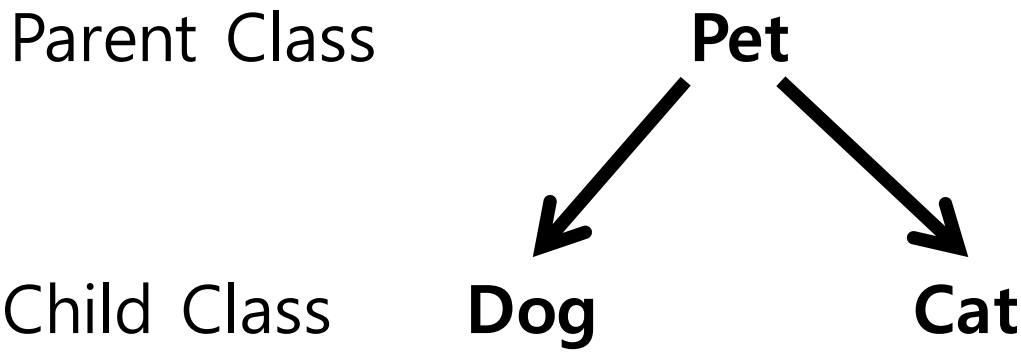
Methods

- 울어
- 점프

Cat

IS-A Relationship

- IS-A relationship
  - A class가 B class의 child class인가를 확인하는 방법 → A is B가 옳은지 살펴본다
  - A Dog is a Pet (O)
  - A Pet is a Dog (X)
  - A Cat is a Pet (O)



Pet Class

```
class Pet:
    def __init__(self, name, age, fur_color):
        self.name = name
        self.age = age
        self.fur_color = fur_color

    def run(self):
        print(self.name + ": 헥헥”)
```

Dog Class

A Dog is a Pet relation

```
class Dog(Pet):
    def __init__(self, name, age, fur_color,
dog_breed):
        Pet.__init__(self, name, age, fur_color)
        self.dog_breed = dog_breed
    def bark(self):
        print(self.name + ": 멍멍")

    def attack(self):
        print(self.name + ": 컹컹")
```

Initializing the Dog part

Initializing the Pet part

Cat Class

A Cat is a Pet relation

```
class Cat(Pet):
    def __init__(self, name, age, fur_color,
cat_breed):
        Pet.__init__(self, name, age, fur_color)
        self.cat_breed = cat_breed
    def mew(self):
        print(self.name + ": 야옹")

    def jump(self):
        print(self.name + ": 휘리릭")
```

Initializing the Cat part

Initializing the Pet part

Play with Cat

```
>>> from pet import *
>>> byul = Cat("별이", 1, "흰색", "노르웨이숲")
>>> byul.jump()
별이: 휘리릭
>>> byul.run()
별이: 헥헥
>>> byul.bark()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Cat' object has no attribute
'bark'
```

Method Overloading

```
class Cat(Pet):
    def __init__(self, name, age, fur_color,
cat_breed):
        Pet.__init__(self, name, age, fur_color)
        self.cat_breed = cat_breed

    def run(self):
        print(self.name + ": 휙")

    def mew(self):
        print(self.name + ": 야옹")

    def jump(self):
        print(self.name + ": 휘리릭")
```

Overloading Pet.run()

graphics.py

Visit http://mcsp.wartburg.edu/zelle/python/
and download graphics.py to C:\Python34

graphics.py

```
>>> from graphics import *
>>> win = GraphWin(width = 500, height = 500)
>>> p = Point(250, 250)
>>> p.draw(win)
>>> c = Circle(p, 50)
>>> c.draw(win)
>>> c.setFill('blue')
>>> c.move(100, 100)
>>> q = Point(10, 10)
>>> q.draw(win)
>>> l = Line(q, p)
>>> l.draw(win)
>>> l.undraw()
>>> c.undraw()
```

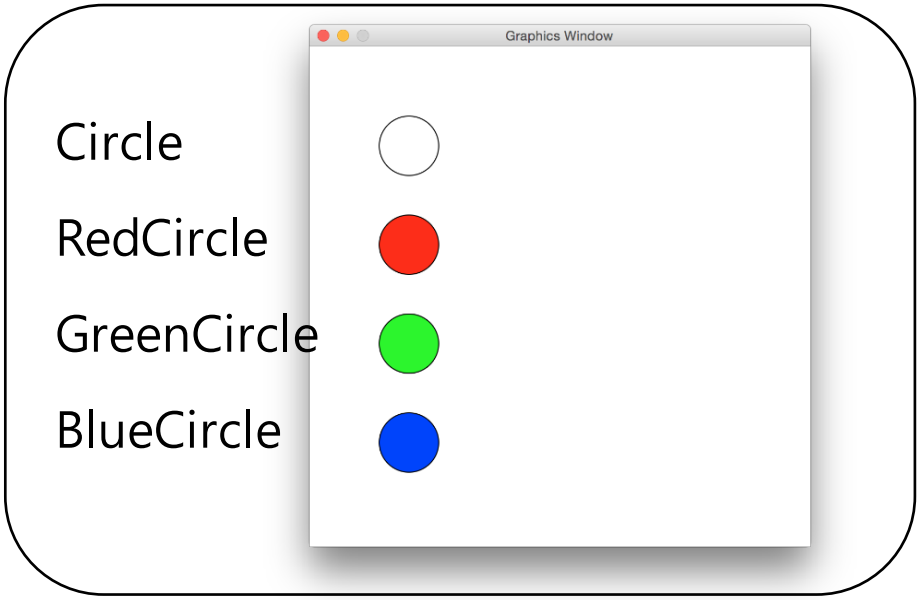
Practice #1

```
from graphics import *

win = GraphWin(width = 500, height = 500)

while True:
    p = win.getMouse()
    print(p.getX(), p.getY())
```

- 클릭할 때마다 그 위치에 반지름 30의 원을 그려라
- 원을 red, green, blue, yellow 중의 임의의 색으로 채워라
- 마지막에 그린 5개의 원만 보이도록 하라

Practice #2

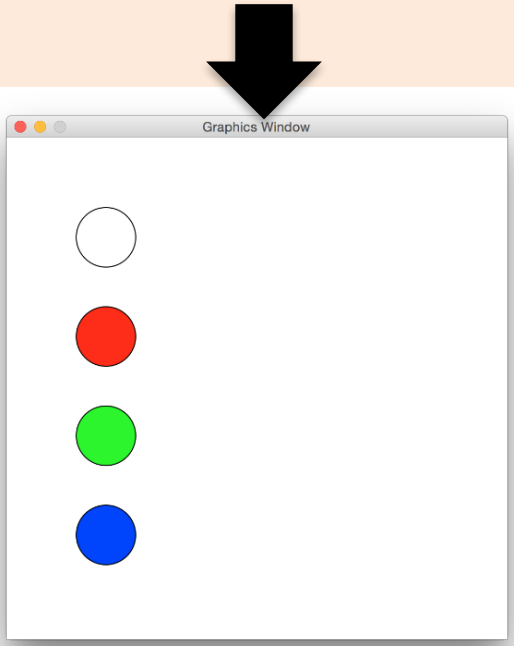
- graphics.py 파일을 수정하여 아래 3개의 Class를 추가하라
  - RedCircle
  - GreenCircle
  - BlueCircle



- 세 Class는 모두 Circle을 상속받도록 한다

Practice #2

```
>>> from graphics import *
>>> win = GraphWin(width = 500, height = 500)
>>> c = Circle(Point(100, 100), 30)
>>> red_c = RedCircle(Point(100, 200), 30)
>>> green_c = GreenCircle(Point(100, 300), 30)
>>> blue_c = BlueCircle(Point(100, 400), 30)
>>> c.draw(win)
>>> red_c.draw(win)
>>> green_c.draw(win)
>>> blue_c.draw(win)
```



Advanced Topics

- Operator Overloading
- 다중상속

Questions