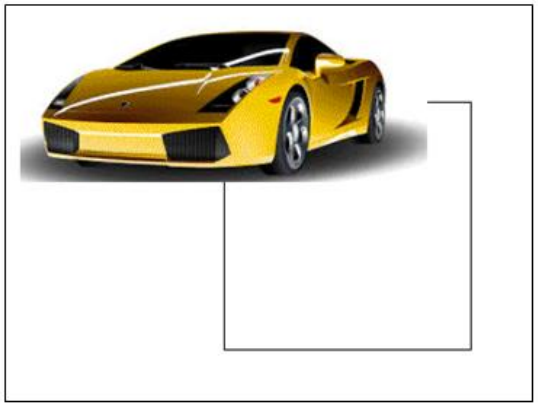
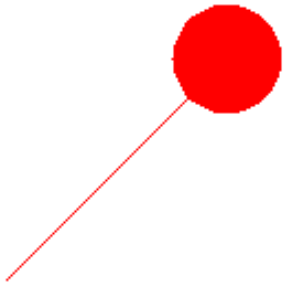
13장 객체란 무엇인가요?

이번 장에서 만들 프로그램

(1) 자동차를 나타내는 클래스를 정의하고 사용해본다.

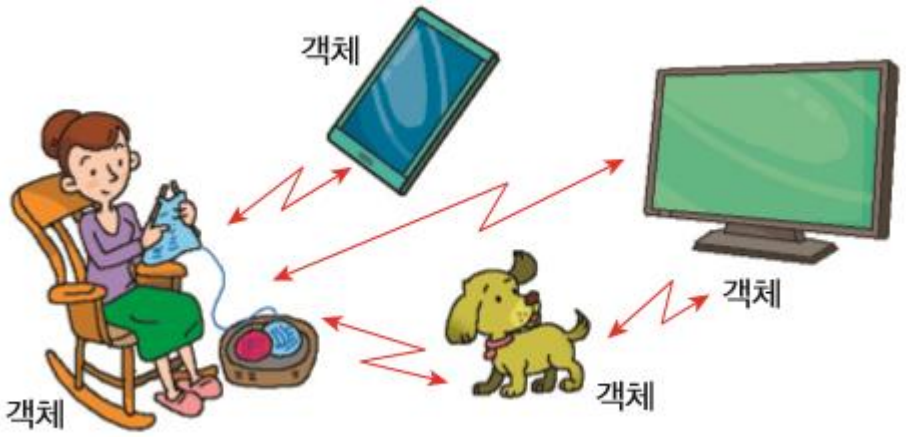


(2) 공을 나타내는 Ball 클래스를 정의하고 사용해본다.



객체 지향 프로그래밍

- 객체(object)는 함수와 변수를 하나의 단위로 묶 을 수 있는 방법이다. 이러한 프로그래밍 방식을 객체지향(object-oriented)이라고 한다.



객체들 사이의 상호작용

- 텔레비전 리모콘은 모두 특정한 기능을 수행하는 객체라고 생각할 수 있고 텔레비전과 리모콘은 메시지 를 통하여 서로 상호 작용하고 있다.

객체란?

- 객체는 하나의 물건이라고 생각하면 된다. 객체는 속성(attribute)과 동작(action)을 가지고 있다.

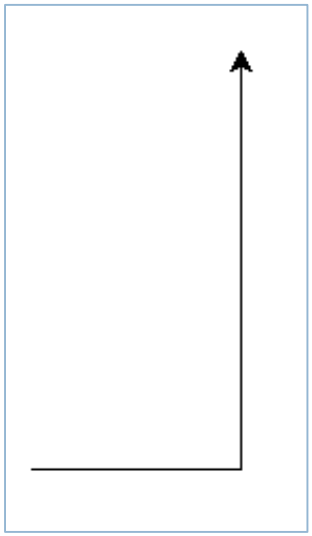
속성
메이커
모델
색상
연식
가격

동작
주행하기
방향바꾸기
주차하기

거북이도 객체

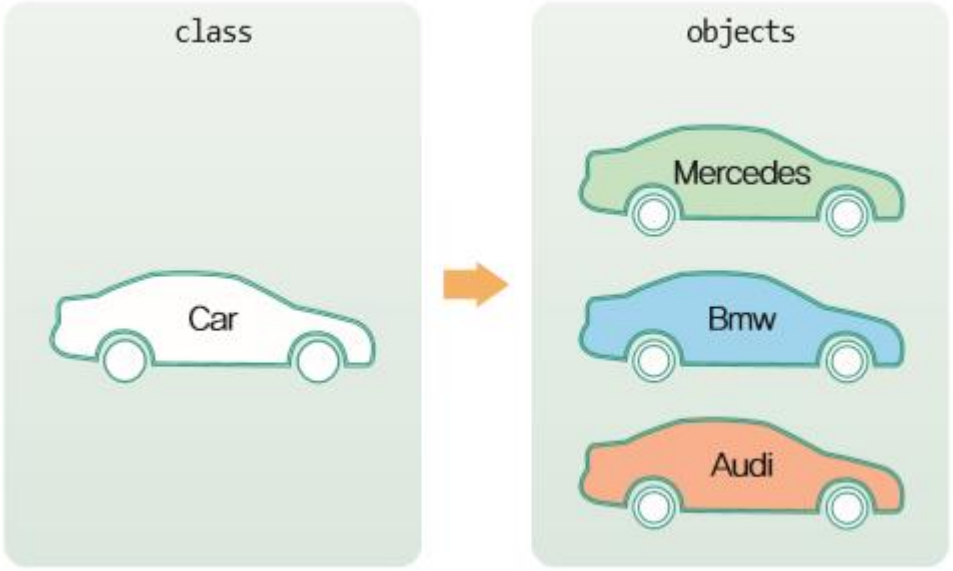
- 터틀 그래픽에서 거북이가 바로 객체

```
from turtle import *     # turtle 모듈에서 모든 것을 불러올것
alex = Turtle()          # 거북이 객체를 생성한다.

alex.forward(100)         # forward()는 거북이 객체의 메소드이다.
alex.left(90)            # left()는 거북이 객체의 메소드이다.
alex.forward(200)         # forward()는 거북이 객체의 메소드이다.
```



객체 생성하기

1. 객체의 설계도를 작성하여야 한다.
2. 클래스로부터 객체를 생성하여야 한다.



객체 생성 코드

```
class Car:
    def drive(self):
        self.speed = 10

myCar = Car()
myCar.color = "blue"
myCar.model = "E-Class"

myCar.drive()                    # 객체 안의 drive() 메소드가 호출된다.
print(myCar.speed)               # 10이 출력된다.
```

객체 생성 코드

```
class Car:
    def drive(self):
        self.speed = 60

myCar = Car()
myCar.speed = 0
myCar.model = "E-Class"
myCar.color = "blue"
myCar.year = "2017"

print("자동차 객체를 생성하였습니다.")
print("자동차의 속도는", myCar.speed)
print("자동차의 색상은", myCar.color)

print("자동차의 모델은", myCar.model)
print("자동차를 주행합니다.")
myCar.drive()
print("자동차의 속도는", myCar.speed)
```

```
자동차 객체를 생성하였습니다.
자동차의 속도는 0
자동차의 색상은 blue
자동차의 모델은 E-Class
자동차를 주행합니다.
자동차의 속도는 60
```

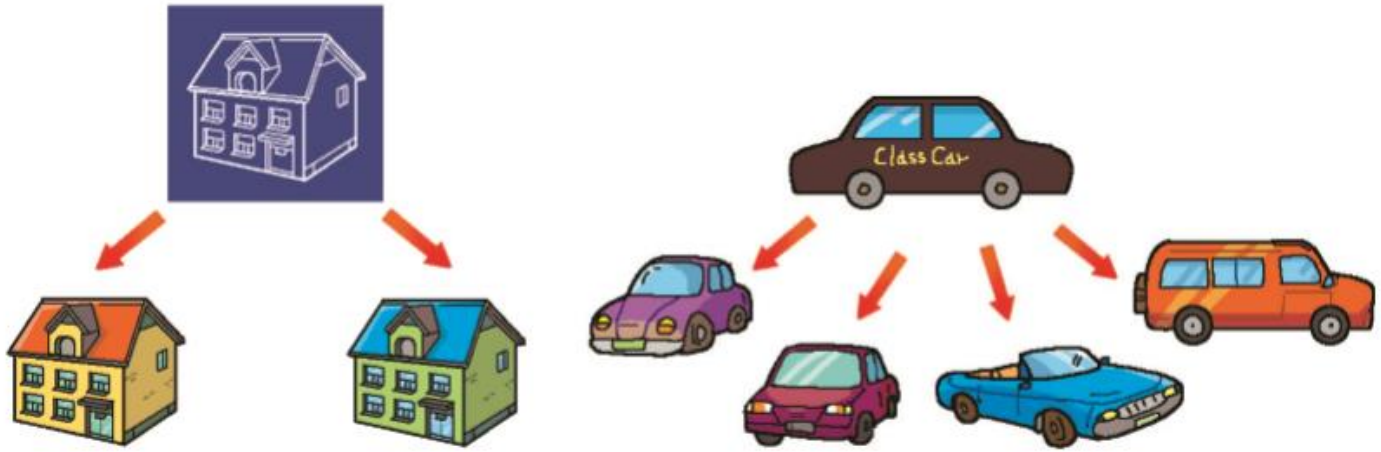
생성자

```
class Car:
    def __init__(self, speed, color, model):
        self.speed = speed
        self.color = color
        self.model = model

    def drive(self):
        self.speed = 60

myCar = Car(0, "blue", "E-class")

print("자동차 객체를 생성하였습니다.")
print("자동차의 속도는", myCar.speed)
print("자동차의 색상은", myCar.color)
print("자동차의 모델은", myCar.model)
print("자동차를 주행합니다.")
myCar.drive()
print("자동차의 속도는", myCar.speed)
```

```
자동차 객체를 생성하였습니다.
자동차의 속도는 0
자동차의 색상은 blue
자동차의 모델은 E-class
자동차를 주행합니다.
자동차의 속도는 60
```

객체 생성

- 우리는 하나의 클래스로 여러 개의 객체를 생성할 수 있다



객체 생성

```
class Car:
    def __init__(self, speed, color, model):
        self.speed = speed
        self.color = color
        self.model = model

    def drive(self):
        self.speed = 60

dadCar = Car(0, "silver", "A6")
momCar = Car(0, "white", "520d")
myCar = Car(0, "blue", "E-class")
```

__str__() 메소드

```
class Car:
    def __init__(self, speed, color, model):
        self.speed = speed
        self.color = color
        self.model = model

    def __str__(self):
        msg = "속도:"+ str(self.speed)+ " 색상:"+ self.color+ " 모델:"+ self.model
        return msg

myCar = Car(0, "blue", "E-class")
print(myCar)
```

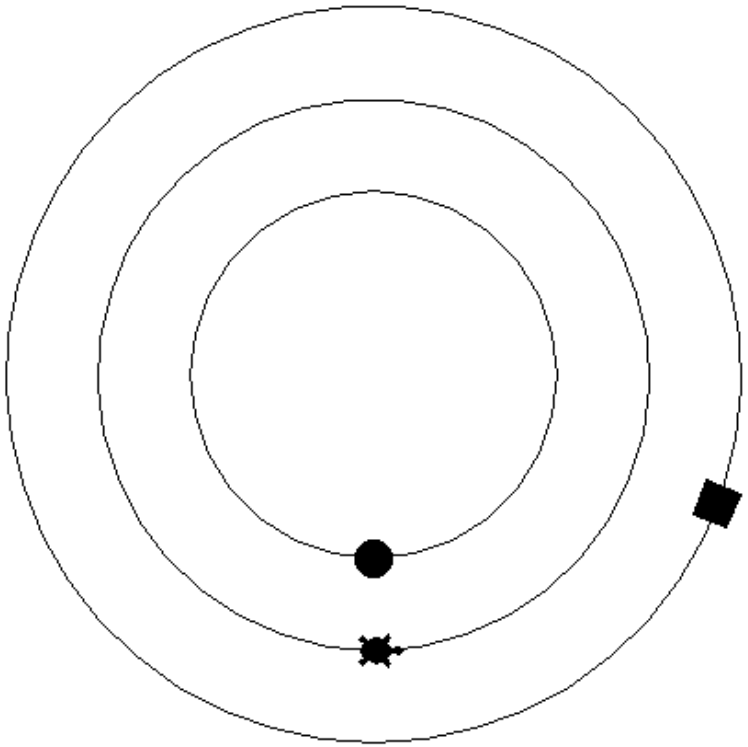
```
속도:0 색상:blue 모델:E-class
```

self는 무엇인가?

```
class Car:
    def __init__(self, speed, color, model):
        self.speed = speed
        self.color = color
        self.model = model

    def drive(self):
        self.speed = 60c

myCar = Car(0, "blue", "E-class")
yourCar = Car(0, "white", "S-class")



myCar.drive()
yourCar.drive()
```

Lab: 터틀 그래픽을 다시보자.

- 터틀 그래픽에서 각각의 거북이가 객체라는 것을 알았다. 거북이 객체를 여러 개 생성하여서 서로 다르게 반복해보자.



Solution

```
from turtle import *    # turtle 모듈에서 모든 것을 불러온다.

t1 = Turtle()          # 거북이 객체를 생성한다.
t1.shape("circle")

t2 = Turtle()          # 거북이 객체를 생성한다.
t2.shape("turtle")

t3 = Turtle()          # 거북이 객체를 생성한다.
t3.shape("square")

t1.penup()                  # 펜을 든다.
t2.penup()
t1.goto(0, 100)   # 거북이를 이동한다.
t2.goto(0, 50)
t1.pendown()                # 펜을 내린다.
t2.pendown()

while True:
    t1.circle(100)          # 원을 그린다.
    t2.circle(150)
    t3.circle(200)
```

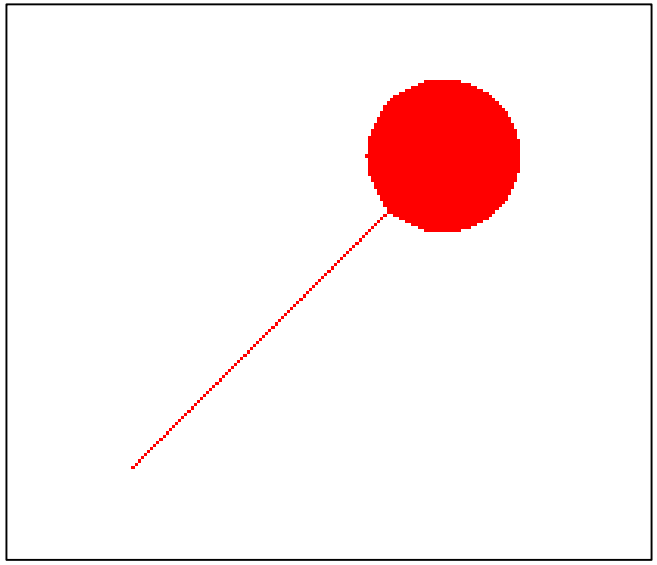
Car 클래스 + Turtle 클래스

- 터틀 그래픽을 사용하여서 화면에 자동차를 그리고 움직여 보자.

```
from turtle import *
class Car:
    def __init__(self, speed, color, model):
        self.speed = speed
        self.color = color
        self.model = model
        self.turtle = Turtle()
        self.turtle.shape("car.gif")

    def drive(self):
        self.turtle.forward(self.speed)

    def left_turn(self):
        self.turtle.left(90)

register_shape("car.gif")
myCar = Car(200, "red", "E-class")
for i in range(100):
    myCar.drive()
    myCar.left_turn()
```

Lab: Ball 클래스

- 컴퓨터 게임에서 많이 등장하는 것이 공을 나타내는 Ball 클래스이다. Ball 클래스의 속성과 메소드를 생각해보자.



Ball의 속성

- 공의 위치(x, y)
- 공의 색상(color)
- 공의 속도(xspeed, yspeed)
- 공의 크기(size)
- 공을 움직이는 메소드 move()

Solution

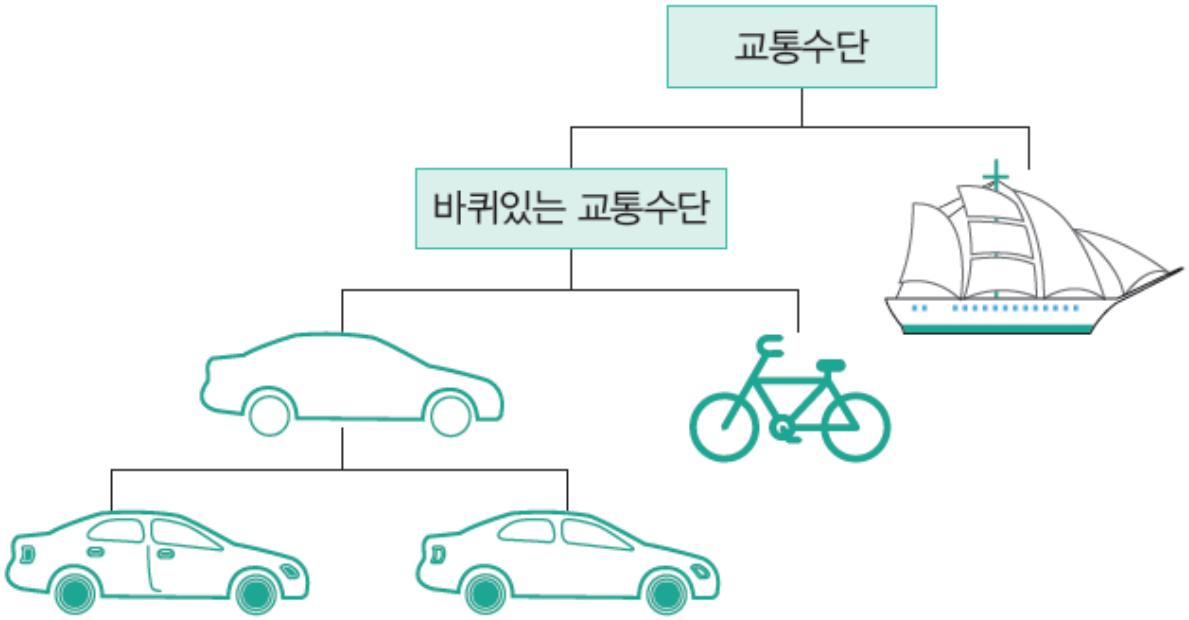
```
from turtle import *
class Ball:
    def __init__(self, color, size, speed):
        # 공의 위치
        self.x = 0
        self.y = 0

        # 공의 속도 벡터
        self.xspeed = speed
        self.yspeed = speed

        # 공의 크기
        self.size = size

        # 공의 색상
        self.color = color
```

Solution

```
        self.turtle = Turtle()
        self.turtle.shape("circle")
        self.turtle.color(color, color)
        self.turtle.resizemode("user")
        self.turtle.shapesize(size, size, 10)

    # 메소드 정의
    def move(self):
        self.x += self.xspeed
        self.y += self.yspeed
        self.turtle.goto(self.x, self.y)

ball = Ball("red", 2, 1)
for i in range(100):
    ball.move()
```

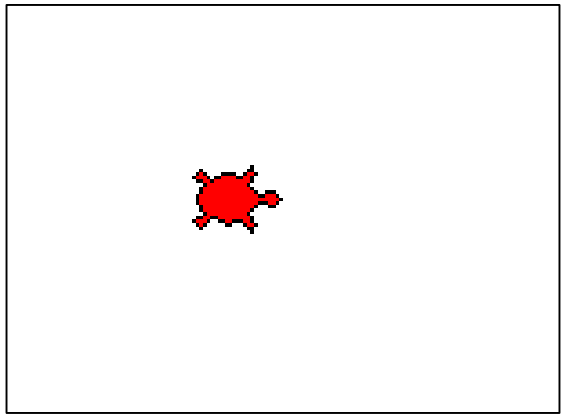
상속이란?

- 상속은 클래스를 정의할 때 부모 클래스를 지정하는 것이다. 자식 클래스는 부모 클래스의 메소드와 변수들을 사용할 수 있다.



MyTurtle 클래스

```
from turtle import *    # turtle 모듈에서 모든 것을 불러온다.

class MyTurtle(Turtle):
    def glow(self,x,y):
        self.fillcolor("red")

turtle = MyTurtle()
turtle.shape("turtle")
turtle.onclick(turtle.glow)     # 거북이를 클릭하면 색상이 빨강색으로 변경된
```



이번 장에서 배운 것

- *클래스는 속성과 동작으로 이루어진다. 속성은 인스턴스 변수로 표현되고 동작은 메소드로 표현된다.*
- *객체를 생성하려면 생성자 메소드를 호출한다. 생성자 메소드는 __init__() 이름의 메소드이다.*
- *인스턴스 변수를 정의하려면 생성자 메소드 안에서 self.변수이름 과 같이 생성한다.*

Q & A