5. 객체 지향 프로그램의 시작에 앞서[메소드]

1. 객체지향 이전의 프로그래밍은 〈절차지향〉이였다. 위에서부터 차근차근 실행되는 방식

2. 절차 지향의 한계를 느껴 함수 또는 메소드의 개념을 만들어, 코드를 재활용 하기 쉽도록 하였다.

3. 로직만 만들어 놓고, 그때 그때 데이터를 주면 메소드(함수)가 알아서 결과값을 반환하는 방식

4. 하지만 메소드를 이용한 방식에도 한계는 존재

* 데이터가 많아지고, 메소드가 많아지면 코드의 양이 방대해지고 어려워짐

* 이를 해결하기 위해 만들어진 방식이 객체지향 프로그래밍

객체란 동일한 성질의 데이터와 메소드를 한곳에 모아두고 필요한 곳에서 언제든지 이용할 수 있게 만들어 놓은 덩어리

```
String color ;
int cc;
int speed;

public void drive() {
    speed = 60;
    System.out.println(color + "색 차를 운전한다. 지금 속도 " + speed);
}
```

speed라는 변수를 위에서 선언해주고, drive()라는 이름을 가진 함수를 만든 후 , speed가 60일 경우

자동차의 color를 출력 한 후 speed 60을 뽑아내는 것

```
public static void main(String[] args) {
    Car mycar = new Car();
    mycar.color = "빨강";
    System.out.println(mycar.color + "색 차의 " + "배기량" + mycar.cc + "속도"+ mycar.speed);

    mycar.drive();
```

메소드를 만든 후 testmain에서 결과값을 출력하기 위해선 '객체'를 생성해주어야함

즉 함수를 만들 때 선언해놓은 변수들을 활용하여 Car라는 객체를 하나 만들고 그 객체가 가지고 있는 속성과 메소드를 new Car를 통해 넣어주기

Mycar.drive(); 즉 drive라는 메소드를 호출했을 경우 speed가 60으로 변경되고 출력창에서 red 색 차를 운전한다. 지금속도 60;이 출력될 것.

이렇듯 어떤 함수를 호출하냐에 따라 내가 뽑아낼수있는 호출값을 설정할수있음