

객체지향 프로그래밍

C언어

- 절차적 프로그래밍은 데이터에 대한 작업을 수행하는 절차나 메서드를 작성합니다
- 객체 지향 프로그래밍은 데이터와 메서드를 모두 포함하는 개체를 만드는 것입니다.

객체 지향 프로그래밍은 절차적 프로그래밍에 비해 몇 가지 장점이 있습니다.

- 더 빠르고 실행하기 쉽습니다.
- 프로그램에 대한 명확한 구조를 제공합니다.
- 코드를 더 쉽게 유지 관리, 수정 및 디버깅할 수 있도록 해줍니다.
- 더 적은 코드와 더 짧은 개발 시간으로 완전히 재사용 가능한 애플리케이션을 만들 수 있습니다.

클래스와 객체란 무엇입니까?

- 클래스는 객체의 템플릿이고 객체는 클래스의 인스턴스입니다.
- 개별 객체가 생성되면 클래스의 모든 변수와 메서드를 상속받습니다.

클래스 및 객체

Java는 객체 지향 프로그래밍 언어입니다.

Java의 모든 것은 해당 속성 및 메소드와 함께 클래스 및 객체와 연관되어 있습니다. 예를 들어 실생활에서 자동차는 객체입니다. 자동차에는 무게, 색상과 같은 속성 과 구동 및 브레이크와 같은 방법이 있습니다.

클래스를 만들려면 class 키워드를 사용하세요

```
Main.java
"Main"클래스를 사용하여 변수 x를 만듭니다.

public class Main {
    int x = 5;
}
```

클래스는 항상 대문자 첫 글자로 시작해야 하며 Java 파일 이름은 클래스 이름과 일치해야 한다는 점을 기억하세요.

객체 생성

Java에서는 클래스에서 객체가 생성됩니다.

Main클래스의 객체를 생성하려면 클래스이름과 객체 이름을 차례로 지정하고 new키워드를 사용하십시오 .

```
public class Main {
    public static void main(String[] args) {
        Main myObj = new Main();
    }
```

생성자

```
}
```

클래스 속성

- 변수라는 용어를 사용했습니다.
- 자바에서 클래스의 속성입니다.
- 클래스 속성이 클래스 내의 변수라고 말할 수도 있습니다.

```
public class Main {  
    int x = 5;  
    int y = 3;  
}
```

Main이라는 클래스안에 x, y 라는 두 가지 속성을 만듭니다

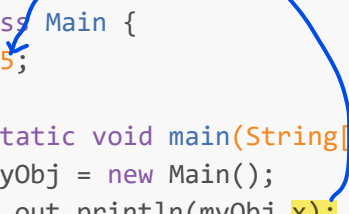
클래스 속성의 또 다른 용어는 필드 입니다.

속성에 접근하기

클래스의 객체를 생성하고 점 구문(.)을 사용하여 속성에 접근할 수 있습니다.

- 다음 예제에서는 이름이 Main인 클래스의 개체를 만듭니다
- 객체의 속성을 사용하여 myObj.x 해당 값을 출력합니다.

```
public class Main {  
    int x = 5;  
  
    public static void main(String[] args) {  
        Main myObj = new Main();  
        System.out.println(myObj.x); // 5  
    }  
}
```



결과 : 5

속성 수정

속성 값을 수정할 수도 있습니다.

x속성 값을 40으로 설정합니다.

```
public class Main {  
    int x;  
  
    public static void main(String[] args) {  
        Main myObj = new Main();
```

```

    myObj.x = 40;
    System.out.println(myObj.x);
}
}
값 : 40

```

기존 값을 재정의합니다.

```

public class Main {
    int x = 10;

    public static void main(String[] args) {
        Main myObj = new Main();
        myObj.x = 25;
        System.out.println(myObj.x);
    }
}
값 : 25

```

final: 기존 값을 재정의하는 기능을 원하지 않으면 속성을 다음과 같이 선언하세요 .

```

public class Main {
    final int x = 10;

    public static void main(String[] args) {
        Main myObj = new Main();
        myObj.x = 25; // 오류발생!! 최종 변수에 값을 할당할 수 없습니다.
        System.out.println(myObj.x);
    }
}
결과 : 오류발생

```

final 키워드는 PI(3.14159...)와 같이 변수가 항상 동일한 값을 저장하도록 하려는 경우에 유용합니다.

다중 객체

한 클래스의 여러 객체를 생성하는 경우 다른 객체의 속성 값에 영향을 주지 않고 한 객체의 속성 값을 변경할 수 있습니다.

```

public class Main {
    int x = 5;

    public static void main(String[] args) {
        Main myObj1 = new Main(); // Object 1
        Main myObj2 = new Main(); // Object 2
        myObj2.x = 25;
        System.out.println(myObj1.x); // 5
    }
}

```

```
System.out.println(myObj2.x); // 25
}
```

결과 :

5
25

myObj2.x값을 25로 변경하고 myObj1.x 변경하지 않고 그대로 둡니다

다중 속성

원하는 만큼 속성을 지정할 수 있습니다.

```
public class Main {
    String fname = "홍";
    String lname = "길동";
    int age = 24;

    public static void main(String[] args) {
        Main myObj = new Main();
        System.out.println("이름: " + myObj.fname + " " + myObj.lname);
        System.out.println("나이: " + myObj.age);
    }
}
```

결과 :

이름: 홍 길동
나이: 24

공백

클래스 메소드

클래스 메소드

메소드가 클래스 내에서 선언되고 특정 작업을 수행합니다

Main에 myMethod()이름이 지정된 메서드를 만듭니다.

```
public class Main {
    static void myMethod() {
        System.out.println("Hello World!");
    }
}
```

- 메소드를 호출하려면 메소드 이름 뒤에 두 개의 괄호 () 와 세미콜론을 적습니다.
- myMethod()호출 되면 텍스트(작업)를 출력합니다

```
public class Main {
    static void myMethod() {
        System.out.println("Hello World!");
    }

    public static void main(String[] args) {
        myMethod(); // 호출
    }
}
```

정적 vs 공개 (static vs public)

static : 객체에 의해서만 접근할 수 있는 **public**와 달리 클래스의 객체를 생성하지 않고도 접근할 수 있는 메서드를 만들었습니다.

공유

```
public class Main {
    // Static method
    static void myStaticMethod() {
        System.out.println("Static methods");
    }

    // Public method
    public void myPublicMethod() {
        System.out.println("Public methods");
    }

    // Main method
    public static void main(String[] args) {
        myStaticMethod(); // 호출
        // myPublicMethod(); This would compile an error

        Main myObj = new Main(); // myObj 객체생성
        myObj.myPublicMethod(); // 객체.메소드 호출
    }
}
```

결과 :
Static methods
Public methods

객체를 사용한 접근 방법

- myCar라는 Car 객체를 만듭니다.
- myCar객체에 대해 fullSpeed()및 speed()메서드를 호출하고 프로그램을 실행합니다.

```
// Main class 생성
public class Main {
    //fullSpeed() 메소드 생성
```

```

public void fullSpeed() {
    System.out.println("자동차는 빨리가고 있다!");
}

// 매개변수(maxSpeed)가 있는 메소드(speed)를 생성
public void speed(int maxSpeed) { // maxSpeed = 200
    System.out.println("최대 속도: " + maxSpeed);
}

public static void main(String[] args) {
    Main myCar = new Main(); // myCar 객체 생성
    myCar.fullSpeed(); // 함수 호출 => 자동차는 빨리가고 있다!
    myCar.speed(200); // 함수 호출 => 최대 속도: 200
}
}

```

결과 :

자동차는 빨리가고 있다!

최대 속도: 200

1. class키워드를 사용하여 Main사용자 정의 클래스를 만들었습니다 .
2. Main클래스에 fullSpeed()및 speed()메소드를 만들었습니다 .
3. fullSpeed()메소드와 speed() 메소드는 호출될 때 일부 텍스트를 출력합니다.
4. speed()메소드는 (정수형)maxSpeed 이라는 매개변수를 사용합니다 .
5. Main 클래스와 해당 메소드를 사용하려면 Main클래스의 객체를 생성해야 합니다 .
6. Java 프로그램을 실행하는 내장 main()메소드인 메소드로 이동하십시오(main 내부의 모든 코드가 실행 됨).
7. new키워드를 사용하여 이름이 myCar인 개체를 만들었습니다 .
8. myCar개체에 대해 fullSpeed() 및 speed() 메서드를 호출합니다
9. 개체 이름(myCar), 점(.), 메서드 이름(fullSpeed(); 및 speed(200);)을 사용하여 프로그램을 실행합니다.
10. speed()메서드 내부에 int매개변수 200 을 추가한다는 점에 유의하세요 .

점(.)은 객체의 속성과 메소드에 접근하는 데 사용됩니다.

Java에서 메소드를 호출하려면 메소드 이름, 괄호(), 세미콜론(;)을 차례로 작성합니다.

여러 클래스 사용

클래스의 개체를 만들고 다른 클래스에서 액세스하는 것이 좋습니다.

Java 파일의 이름은 클래스 이름과 일치해야 합니다.

```

-- Main.java 파일
public class Main {
    public void fullSpeed() {
        System.out.println("자동차는 빨리가고 있다!!");
    }

    public void speed(int maxSpeed) {
        System.out.println("최대 속도: " + maxSpeed);
    }
}

```

```
}  
}
```

```
-- Second.java 파일  
class Second {  
    public static void main(String[] args) {  
        Main myCar = new Main();  
        myCar.fullSpeed();  
        myCar.speed(200);  
    }  
}
```

결과 :
자동차는 빨리가고 있다!!
최대 속도: 200