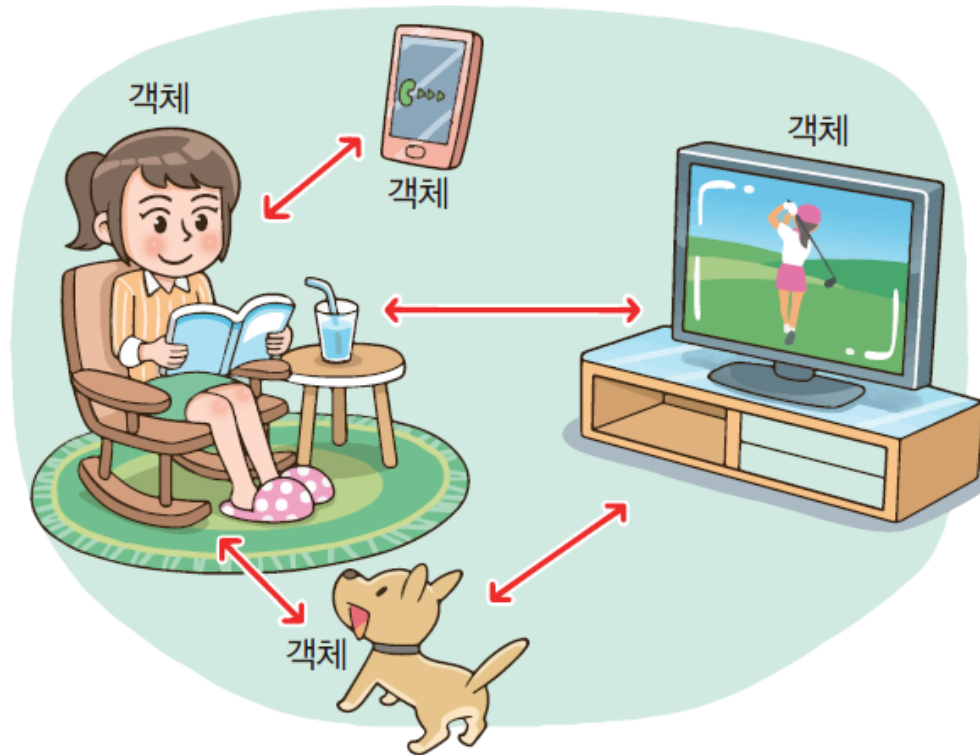


## 05. 클래스



강동기

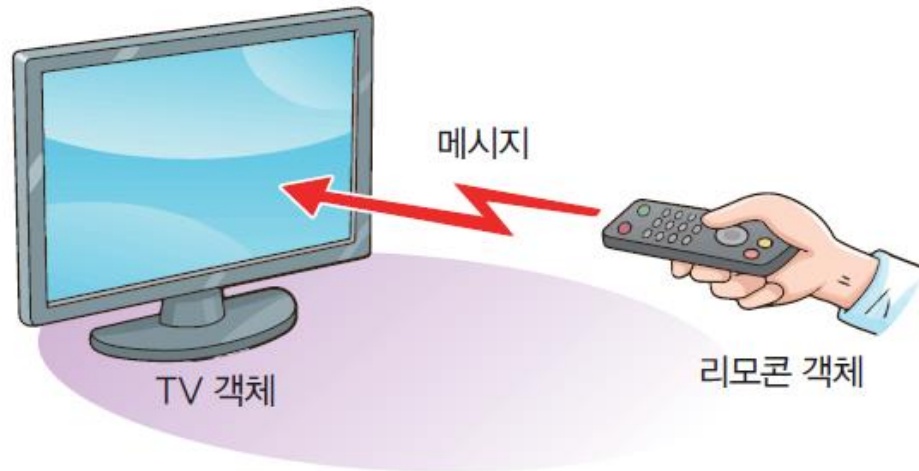
# 실제 세계는 객체로 이루어진다



실제 세계는 객체들로  
이루어져 있죠!



# 객체와 메시지

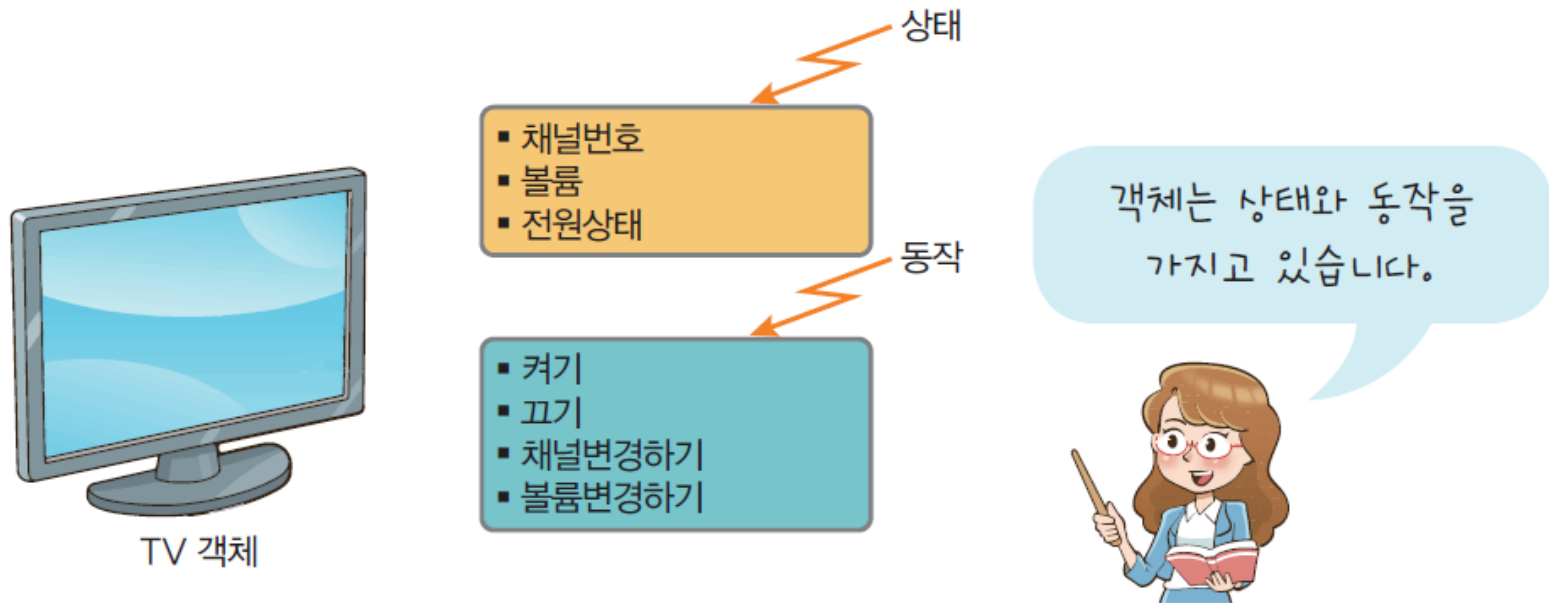


객체들은 메시지를  
보내고 받으면서  
상호 작용합니다.



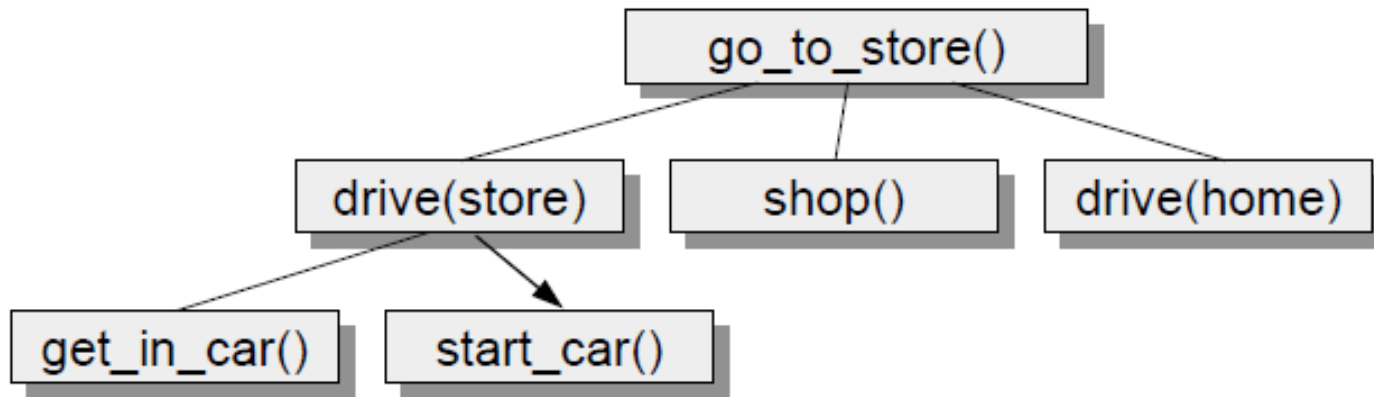
# 객체의 상태와 동작

- 객체(Object)는 상태(State)와 동작(Behavior)을 가지고 있다.



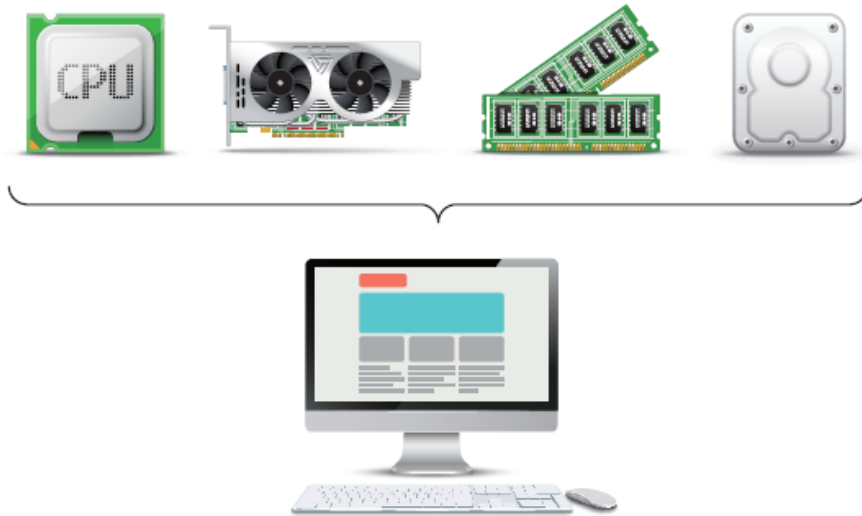
# 절차지향과 객체지향

- 절차 지향 프로그래밍(Procedural Programming)
  - 문제를 해결하는 순서를 중요하게 생각하는 방법이다.

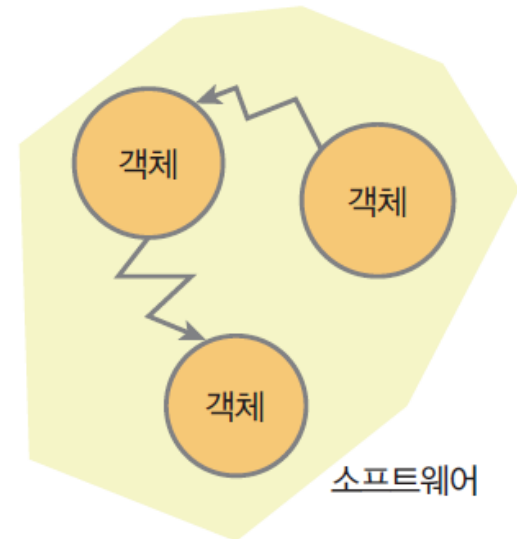


# 절차지향과 객체지향

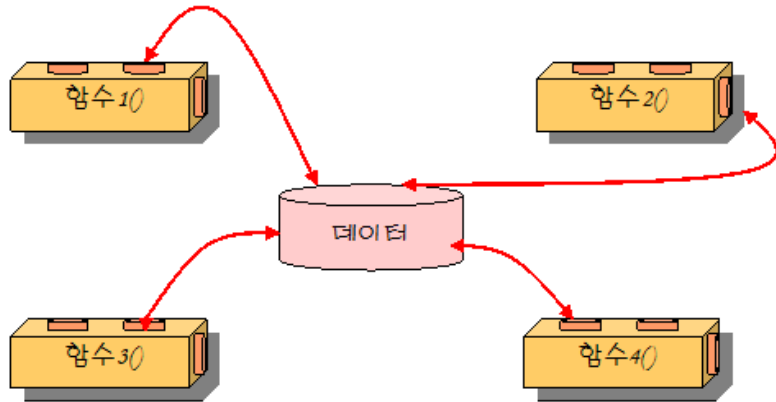
- 객체 지향 프로그래밍(Object-Oriented Programming)
  - 데이터와 절차를 하나의 덩어리(객체)로 묶어서 생각하는 방법이다.



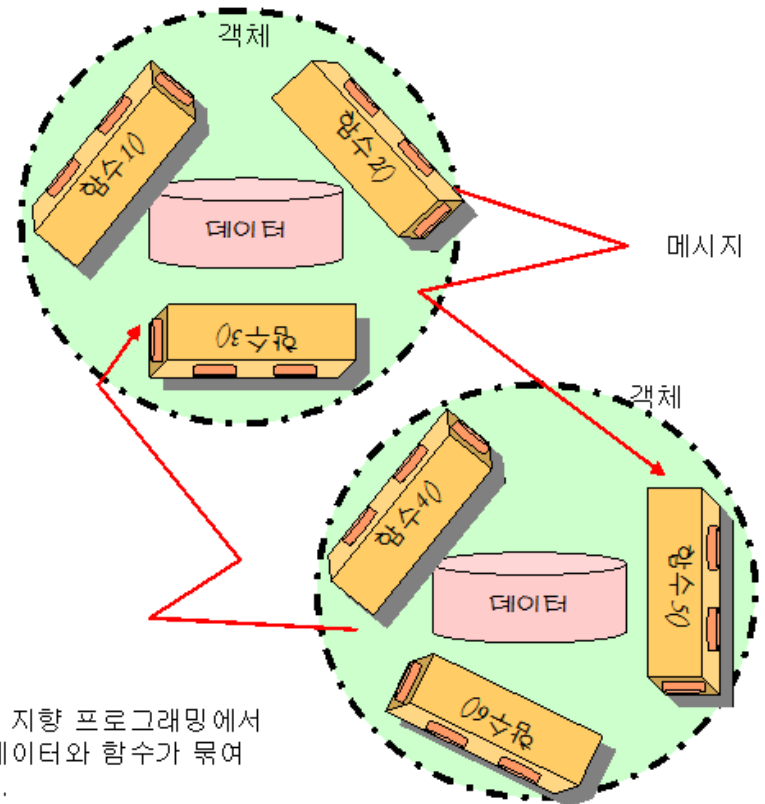
부품을 조립하여 제품을 만들듯이  
객체를 조합하여 소프트웨어를 만든다



# 절차지향과 객체지향



절차 지향 프로그래밍에서  
는 데이터와 함수가 묶여  
있지 않다.



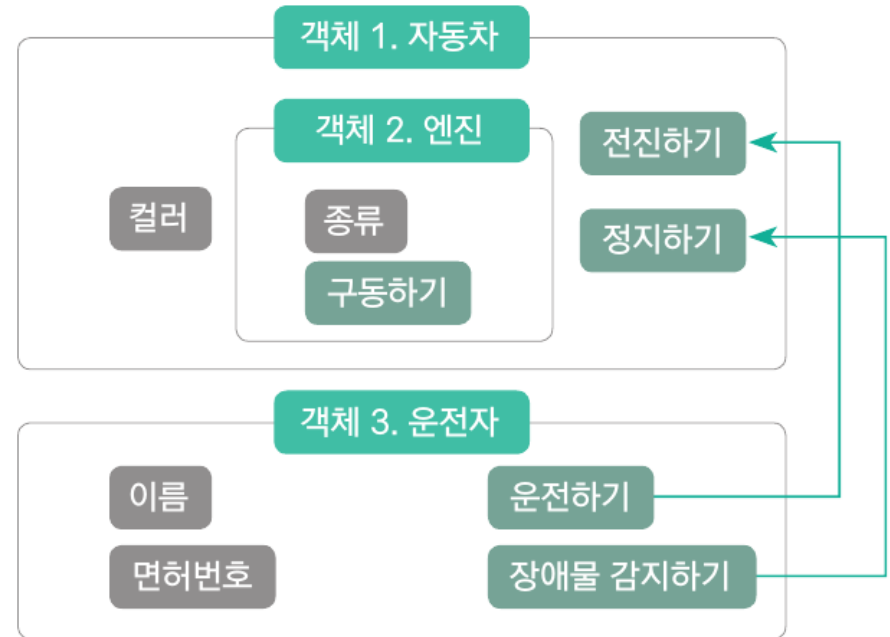
객체 지향 프로그래밍에서  
는 데이터와 함수가 묶여  
있다.

# 절차지향과 객체지향

- 자동차를 운전하는 프로그램



절차지향형 프로그램



객체지향형 프로그램



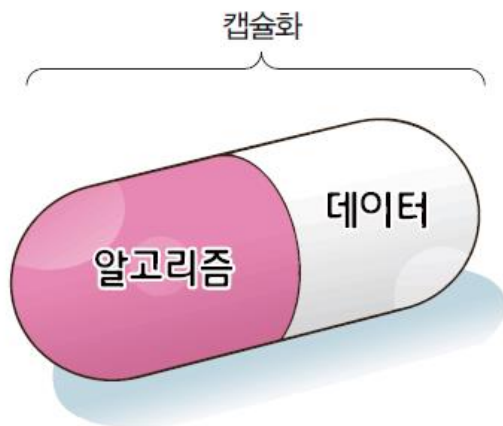
# 객체 지향의 특징

---

- 캡슐화(Encapsulation)
- 상속(Inheritance)
- 다형성(Polymorphism)
- 추상화(Absraction)

# 캡슐화(Encapsulation)

- 관련된 데이터와 코드(알고리즘)이 하나의 묶음으로 정리되어 있는 것



캡슐화는 데이터와 알고리즘을 하나로 묶는 것입니다.



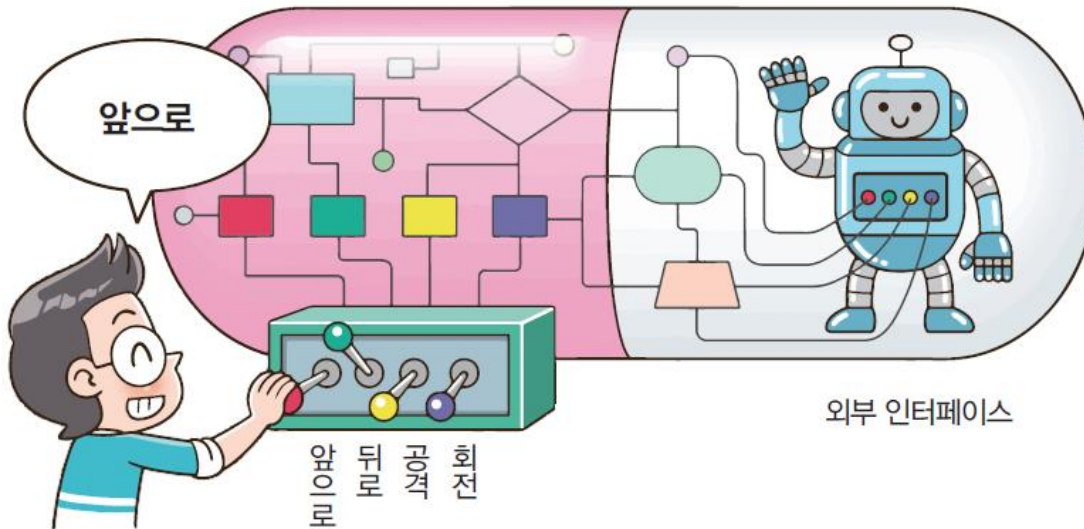
캡슐화 되어 있지 않은 데이터와 코드는 사용하기 어렵겠죠!



# 캡슐화와 정보 은닉

- 정보 은닉(Information Hiding)

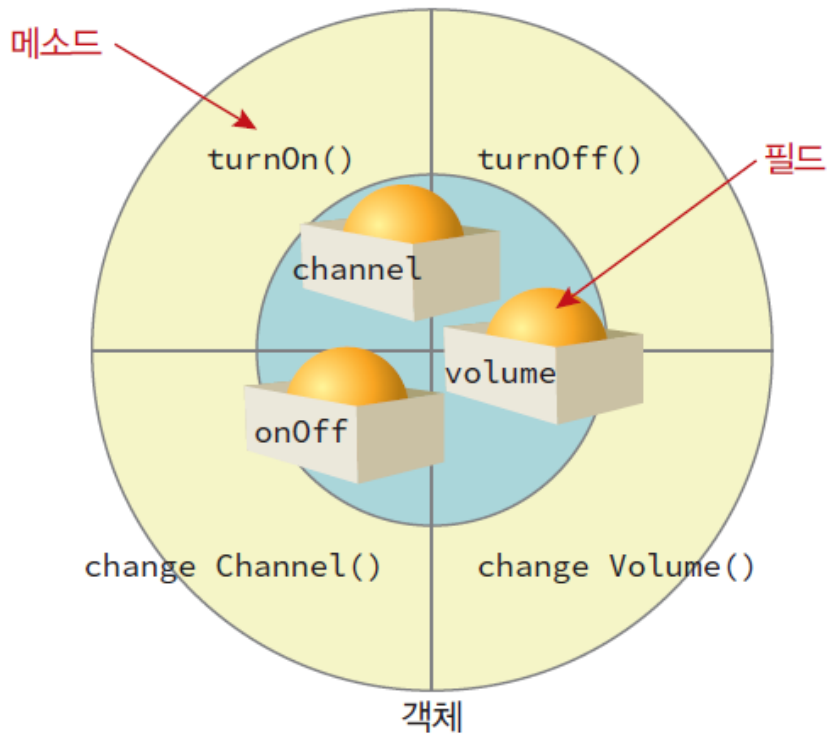
- 객체를 캡슐과 같이 싸서 객체의 내부(상세한 구현 내용)를 숨긴다.
- 별도의 인터페이스를 통해 객체를 접근하고 사용한다.



객체는 공개된  
인터페이스를  
통하여 사용  
하여야 합니다.



# 캡슐화와 정보 은닉

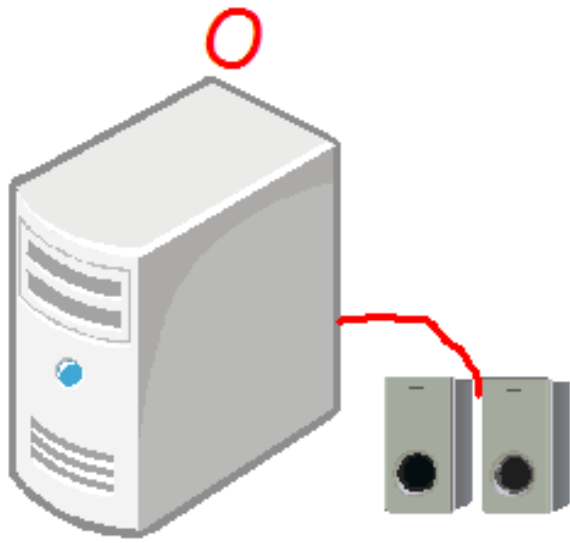


보통은 데이터들은 공개되지 않고 몇 개의 메소드만이 외부로 공개됩니다.

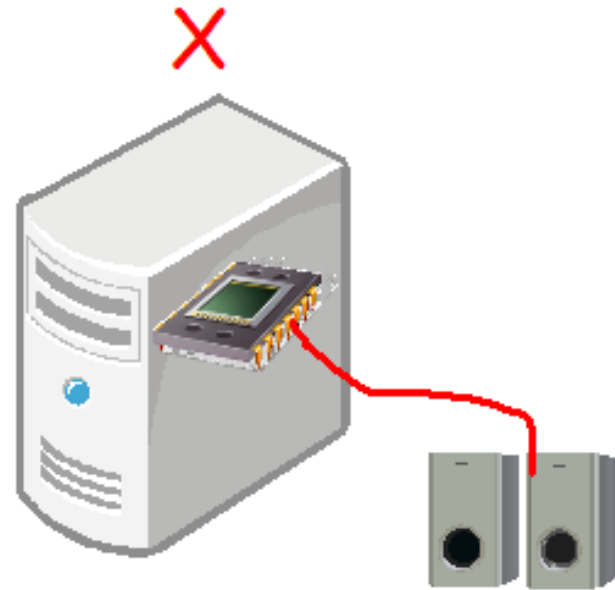


# 캡슐화로 인한 객체의 분리

- 객체가 서로 분리되므로 독자적인 업그레이드가 가능하다.



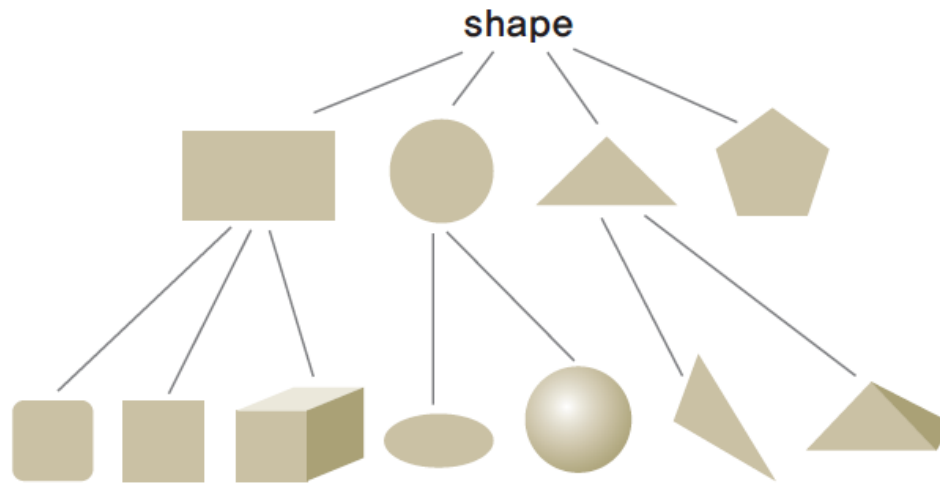
만약 외부의 표준 오디오 단자를 이용하였으면 내부의 사운드 카드를 변경할 수 있다.



만약 내부의 오디오 제어 칩의 단자에 연결하였으면 내부의 사운드 카드를 변경할 수 없다.

# 상속(Inheritance)

- 이미 작성된 클래스를 이어받아서 새로운 클래스를 생성하는 기법
  - 이미 작성된 클래스를 "부모 클래스" 라고 한다.
  - 새로운 클래스를 "자식 클래스" 라고 한다.
- 기존에 만든 코드를 재활용 할 수 있다.

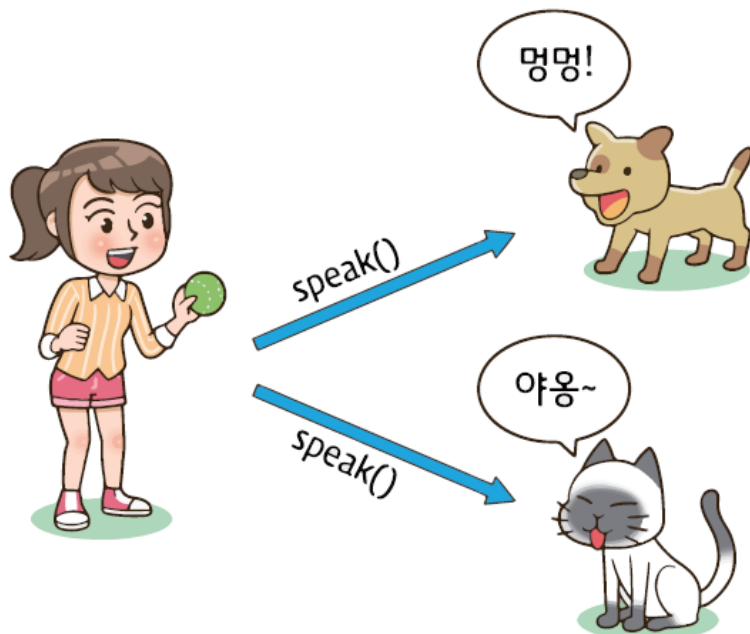


상속은 기존에 만들어진 코드를 이어받아서 보다 쉽게 코드를 작성하는 기법입니다.



# 다형성(Polymorphism)

- 하나의 객체가 여러 가지 타입을 가질 수 있다.
- 부모 클래스 객체로 자식 클래스 객체를 참조할 수 있다.



다형성은 객체의 동작이 상황에 따라서 달라지는 것을 말합니다. “speak”라는 메시지를 받은 객체들이 모두 다르게 소리를 내는 것이 바로 다형성입니다.



# 추상화(Abstraction)

- 클래스들이 공통적으로 가지고 있는 특성을 추출하여 상위 클래스를 정의하는 것
- 자바에서는 클래스외에 인터페이스의 개념을 활용하여 추상화를 구현한다.



실제 객체



추상화된 객체

추상화는 필요한 것만을 남겨놓는 것입니다. 추상화 과정이 없다면 사소한 것도 신경 써야 합니다.





# 클래스

- 클래스(Class)란?
  - 객체를 만들기 위한 설계도를 의미한다.
- 클래스 관점에서의 객체란?
  - 메모리에 적재하여, 코드로서 사용할 수 있는 클래스의 실체가 된다.
  - 클래스로부터 만들어지는 각각의 객체를 클래스의 인스턴스라고 한다.



# 클래스는 왜 사용할까?

---

- 학생 정보를 저장하고 관리하는 프로그램 만들기
  - 학생의 신상과 여러 정보를 저장할 변수가 필요하다.
  - 정보를 출력 및 관리할 수 있는 함수가 필요하다.

# 클래스는 왜 사용할까?

- NoClassEx.java
  - 클래스를 사용하지 않고 짤다면

```
package ch05;

public class NoClassEx {
    public static void main(String[] args) {
        // 학생 1
        String name1 = "Alice";
        int age1 = 20;
        double grade1 = 3.8;

        // 학생 2
        String name2 = "Bob";
        int age2 = 22;
        double grade2 = 3.5;

        double total = grade1 + grade2;
        double avg = total / 2;

        System.out.println(name1 + " (" + age1 + "세), 학점: " + grade1);
        System.out.println(name2 + " (" + age2 + "세), 학점: " + grade2);
        System.out.println("학점 평균: " + avg);
    }
}
```

# 클래스는 왜 사용할까?

- NoClassEx.java
  - 클래스를 사용하지 않고 짤다면

```
// 학생 1
String name1 = "Alice";
int age1 = 20;
double grade1 = 3.8;

// 학생 2
String name2 = "Bob";
int age2 = 22;
double grade2 = 3.5;

// 학생 3
String name3 = "Faker";
int age3 = 21;
double grade3 = 4.5;

// 학생 4
String name4 = "Chovy";
int age4 = 19;
double grade4 = 4.3;
```

...

**학생 수가 늘어나면 변수도 같이 늘려야 함. 변수이름을 짓고 관리하기 어려움.**

# 클래스는 왜 사용할까?

- NoClassEx.java
  - 배열을 활용한다면

```
package ch05;

public class NoClassEx {
    public static void main(String[] args) {
        String[] names = {"Alice", "Bob", "Faker", "Chovy", ...};
        int[] ages = {20, 21, 21, 19, ...};
        double[] grades = {3.8, 3.5, 4.5, 4.2, ...};

        for(int i=0; i<names.length; i++) {
            double total_ages += ages[i];
            double total_grades += grades[i];
        }

        double avg_ages = total_ages / ages.length;
        double avg_grades = total_grades / grades.length;

        for(int i=0; i<names.length; i++) {
            System.out.println(names[i] + " (" + ages[i] + "세), 학점: " + grades[i]);
        }

        System.out.println("나이 평균: " + avg_ages);
        System.out.println("학점 평균: " + avg_grades);
    }
}
```

같은 자료형의 변수들을 1개의 새로운 자료형으로 묶을 수 있어 사용과 관리가 편함.  
그러나 여전히 데이터와 함수가 따로 놀기 때문에 프로그램 덩치가 커질수록 확장 및 디버깅이 어려움.

# 클래스는 왜 사용할까?

- YesClassEx.java
  - 클래스를 활용한다면

```
package ch05;

class Student {
    String name;
    int age;
    double grade;

    Student(String name, int age, double grade) {
        this.name = name;
        this.age = age;
        this.grade = grade;
    }

    void printInfo() {
        System.out.println(name + " (" + age + "세), 학점: " + grade);
    }
}

public class YesClassEx {
    public static void main(String[] args) {
        Student s1 = new Student("Alice", 20, 3.8);
        Student s2 = new Student("Bob", 22, 3.5);
        Student s3 = new Student("Faker", 22, 3.5);
        Student s4 = new Student("Chovy", 22, 3.5);

        ...

        s1.printInfo();
        s2.printInfo();
        s3.printInfo();
        s4.printInfo();
    }
}
```

**데이터와 함수(메서드)가 같은 그룹으로 묶이므로 코드 관리, 확장, 디버깅이 용이해짐.**  
**코드가 길어지고 메모리 사용량은 늘어나지만 최신 컴퓨터 스펙에서는 큰 문제가 안됨.**

# 클래스를 포함하는 자바 소스 파일 구조

- 패키지 (Package)
  - 여러개의 클래스를 묶은 라이브러리 집합
- 임포트 (Import)
  - 다른 패키지의 클래스를 사용하고 싶을 때 작성하는 키워드

A.java

```
package ...;           // ① 패키지
import ...;            // ② 임포트
class 클래스명 {...}    // ③ 외부 클래스
```

클래스의 밖에 올 수 있는 3가지

파일명과 동일해야 함.

```
public class A {
    int a = 3;           // ① 필드
    double abc() {...}   // ② 메서드
    A() {...}            // ③ 생성자
    class 클래스명 {...}  // ④ 이너 클래스
}
```

클래스의 안에 올 수 있는 4가지

# 클래스 내부 구성 요소

---

- 필드 (Field)
  - 클래스내에 포함된 변수 (예: human class 의 나이 변수 int age)
- 메서드 (Method)
  - 클래스내에 포함된 함수 (예: human class 의 먹기 함수 eat())
- 생성자 (Constructor)
  - 클래스와 똑같은 이름의 메서드. 객체를 생성하는 역할 수행
- 이너 클래스 (Inner Class)
  - 클래스 내부에 포함된 클래스



# 객체 생성하기

- new 키워드로 객체 생성하기
  - 모든 클래스는 객체를 new 키워드로 생성함
  - 메서드로 객체를 생성하는 경우에도 해당 메서드 내부에서 new 키워드를 사용함

## 클래스의 객체 생성

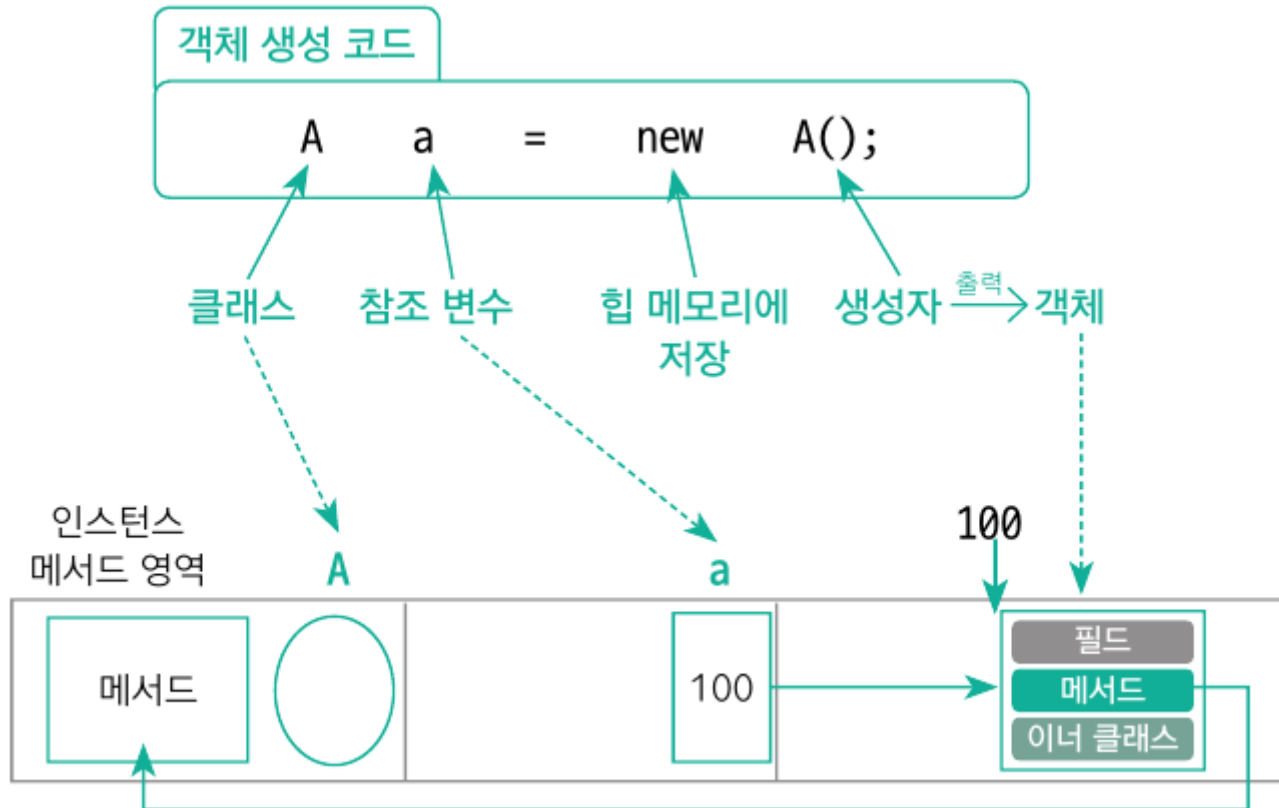
```
클래스명 참조 변수명 = new 생성자();
```

예

```
A a = new A();
```

# 객체 생성하기

- 객체 생성에 따른 메모리 구조
  - 객체 참조변수는 스택영역에 저장됨
  - 객체의 필드, 이너클래스는 힙영역에 저장됨
  - 구현코드는 JVM 메서드영역(클래스영역)에 저장됨



# 객체 활용하기

- 포인트 연산자 사용하기
  - 소스코드로 힙영역에 직접 접근할수는 없음
  - 포인트 연산자를 통해 참조 변수를 이용하여 객체에 접근함

## 필드와 메서드의 활용

참조 변수명.필드명

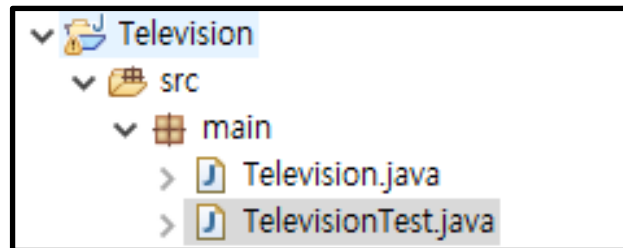
참조 변수명.메서드명()

## 필드와 메서드의 활용 예

```
A a = new A();  
System.out.println(a.m);    // 필드 활용  
a.print();                  // 메서드 활용
```

# 예제: 텔레비전 클래스

- 프로젝트 구조



# 예제: 텔레비전 클래스

- Television.java

```
package main;

public class Television {
    int channel;
    int volume;
    boolean onOff;
}
```

# 예제: 텔레비전 클래스

- TelevisionTest.java

```
package main;

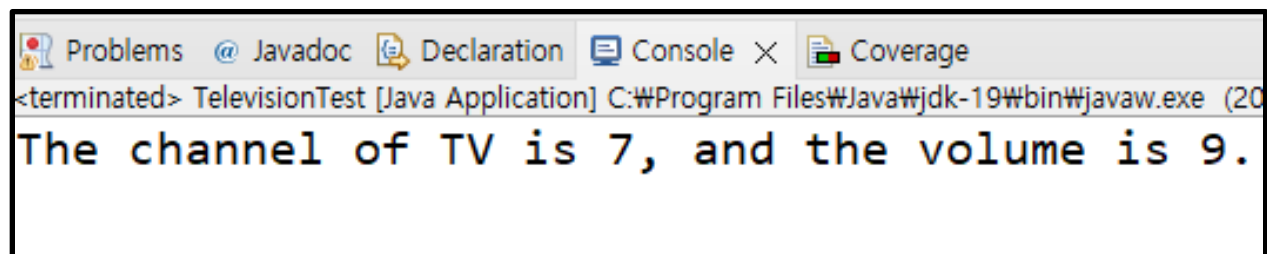
public class TelevisionTest {
    public static void main(String[] args) {
        Television tv = new Television();

        tv.channel = 7;
        tv.volume = 9;
        tv.onOff = true;

        System.out.println("The channel of TV is " + tv.channel +
            ", and the volume is " + tv.volume + ".");
    }
}
```

# 예제: 텔레비전 클래스

- 실행결과



The screenshot shows an IDE's console window with the following tabs: Problems, Javadoc, Declaration, Console, and Coverage. The Console tab is active, displaying the output of a Java application. The text in the console is: `<terminated> TelevisionTest [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (20` followed by the output line `The channel of TV is 7, and the volume is 9.`

```
<terminated> TelevisionTest [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (20
The channel of TV is 7, and the volume is 9.
```

# 예제: 텔레비전 클래스

- TelevisionTest.java 변경

```
package main;

public class TelevisionTest {
    public static void main(String[] args) {
        Television myTv = new Television();
        myTv.channel = 7;
        myTv.volume = 9;
        myTv.onOff = true;

        Television yourTv = new Television();
        yourTv.channel = 9;
        yourTv.volume = 12;
        yourTv.onOff = true;

        System.out.println("The channel of myTV is " +
            myTv.channel + ", and the volume is " + myTv.volume + ".");
        System.out.println("The channel of yourTV is " +
            yourTv.channel + ", and the volume is " + yourTv.volume + ".");
    }
}
```



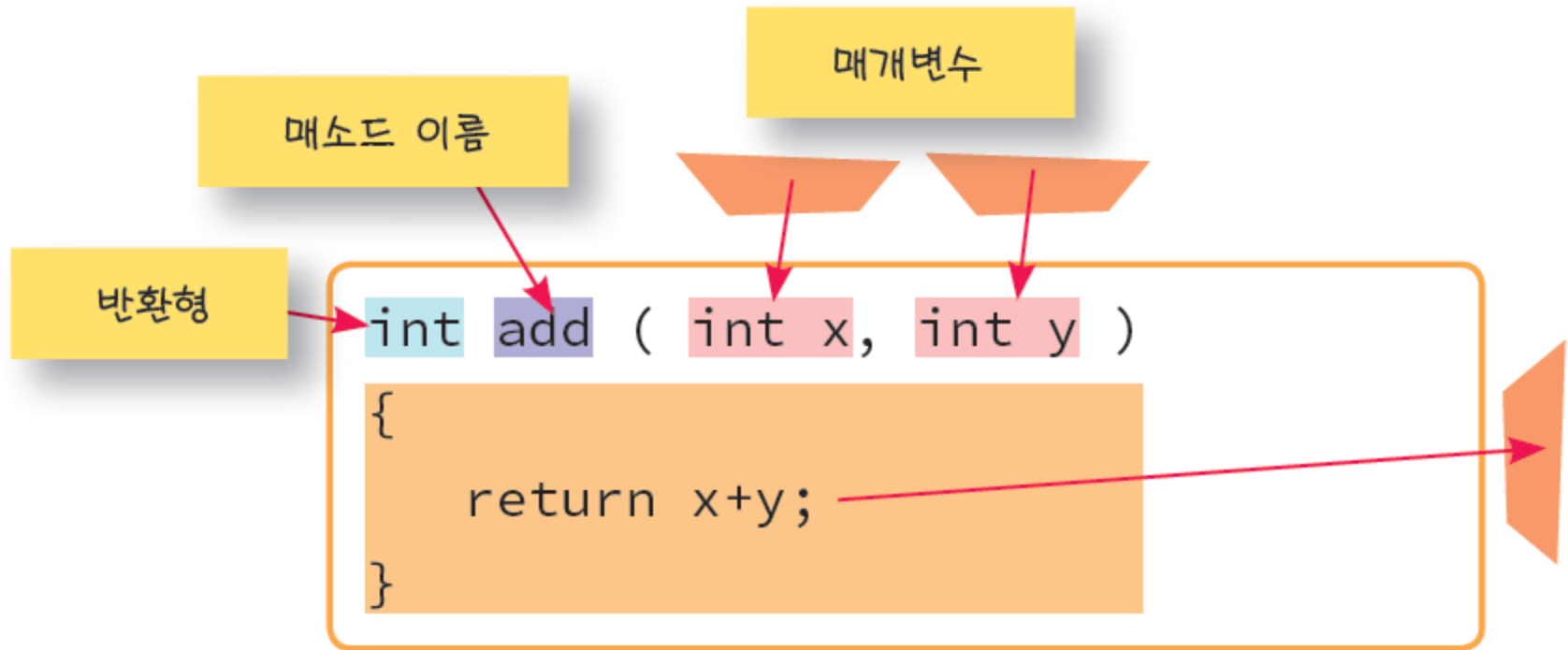
# 예제: 텔레비전 클래스

---

- 실행결과

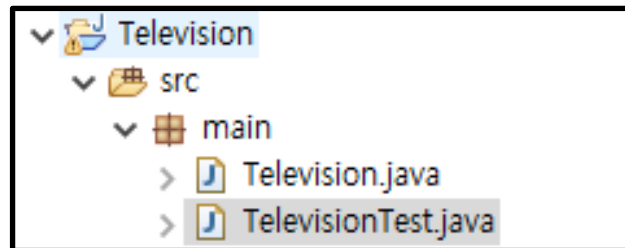
```
The channel of myTV is 7, and the volume is 9.  
The channel of yourTV is 9, and the volume is 12.
```

# 클래스의 메서드



# 예제: 텔레비전 메서드

- 프로젝트 구조



# 예제: 텔레비전 메서드

- Television.java

```
package main;

public class Television {
    int channel;
    int volume;
    boolean onOff;

    void print() {
        System.out.println("The channel is " +
            channel + ", and the volume is " + volume);
    }
}
```

# 예제: 텔레비전 메서드

- TelevisionTest.java

```
package main;

public class TelevisionTest {
    public static void main(String[] args) {
        Television myTv = new Television();
        myTv.channel = 7;
        myTv.volume = 9;
        myTv.onOff = true;
        myTv.print();

        Television yourTv = new Television();
        yourTv.channel = 9;
        yourTv.volume = 12;
        yourTv.onOff = true;
        yourTv.print();
    }
}
```

# 예제: 텔레비전 메서드

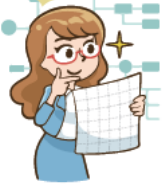
---

- 실행결과

```
The channel is 7, and the volume is 9  
The channel is 9, and the volume is 12
```

# 클래스 메서드의 반환값

전체적인 구조



형식

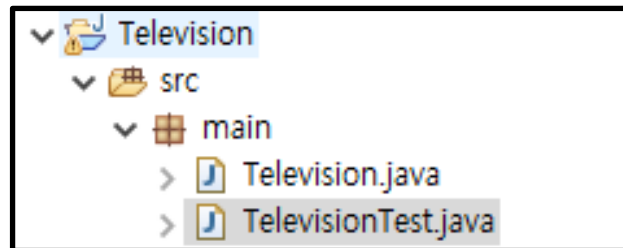
```
return 반환값;
```

return 뒤에 수식을 적으면  
수식의 값이 반환됩니다.



# 예제: 텔레비전 메서드의 반환값

- 프로젝트 구조





# 예제: 텔레비전 메서드의 반환값

- Television.java

```
package main;

public class Television {
    int channel;
    int volume;
    boolean onOff;

    void print() {
        System.out.println("The channel is "
            + channel + ", and the volume is " + volume);
    }

    int getChannel() {
        return channel;
    }
}
```

# 예제: 텔레비전 메서드의 반환값

- TelevisionTest.java

```
package main;

public class TelevisionTest {
    public static void main(String[] args) {
        Television myTv = new Television();
        myTv.channel = 7;
        myTv.volume = 9;
        myTv.onOff = true;

        int ch = myTv.getChannel();
        System.out.println("The current channel is " + ch);
    }
}
```

# 예제: 텔레비전 메서드의 반환값

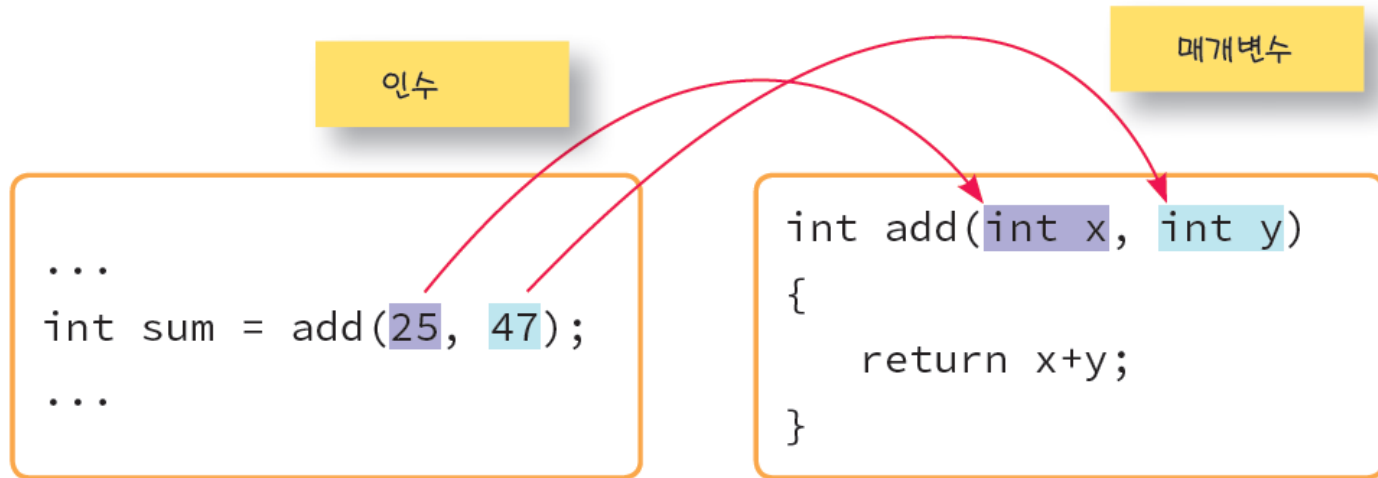
---

- 실행결과

```
The current channel is 7
```

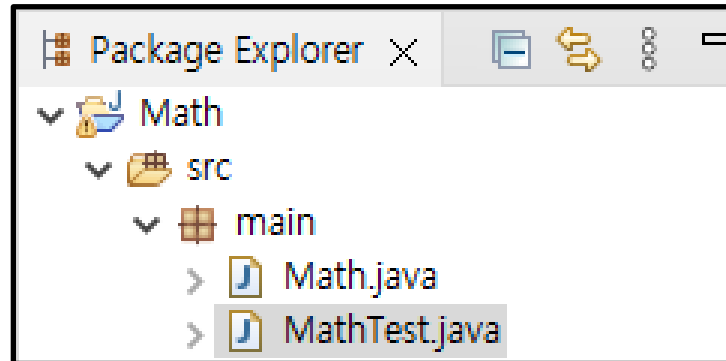
# 인수(Argument)와 매개변수(Parameter)

- 메서드 호출 시 전달하는 값을 인수(Argument)라 함
- 메서드에서 값을 받을 때 사용하는 변수를 매개변수(Parameter)라 함



# 예제: 수학 클래스와 덧셈 메서드

- 프로젝트 구조



# 예제: 수학 클래스와 덧셈 메서드

- Math.java

```
package main;

public class Math {
    int add(int x, int y) {
        return x + y;
    }
}
```

# 예제: 수학 클래스와 덧셈 메서드

- MathTest.java

```
package main;

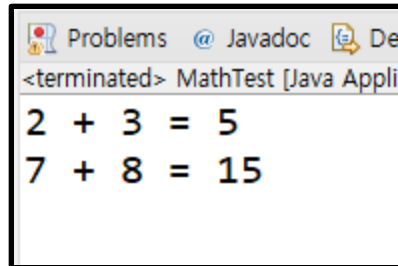
public class MathTest {
    public static void main(String[] args) {
        int sum;
        Math obj = new Math();

        sum = obj.add(2, 3);
        System.out.println("2 + 3 = " + sum);

        sum = obj.add(7, 8);
        System.out.println("7 + 8 = " + sum);
    }
}
```

# 예제: 수학 클래스와 덧셈 메서드

- 실행결과

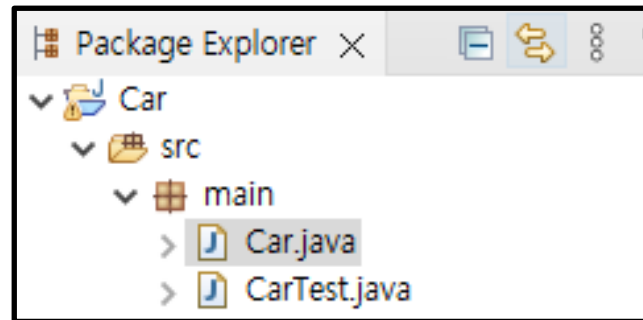


```
Problems @ Javadoc De
<terminated> MathTest [Java Appli
2 + 3 = 5
7 + 8 = 15
```



# 예제: 자동차 클래스

- 프로젝트 구조



# 예제: 자동차 클래스

- Car.java

```
package main;

public class Car {
    String color;
    int speed;
    int gear;

    @Override
    public String toString() {
        return "Car [color=" + color + ", speed="
            + speed + ", gear=" + gear + "];"
    }

    void changeGear(int g) {
        gear = g;
    }

    void speedUp() {
        speed = speed + 10;
    }

    void speedDown() {
        speed = speed - 10;
    }
}
```

# 예제: 자동차 클래스

- CarTest.java

```
package main;

public class CarTest {
    public static void main(String[] args) {
        Car myCar = new Car();

        myCar.changeGear(1);
        myCar.speedUp();

        System.out.println(myCar);
    }
}
```

# 예제: 자동차 클래스

---

- 실행결과

```
Car [color=null, speed=10, gear=1]
```

# 연습문제 1

- 텔레비전 클래스
  - myTv, yourTv 외에 다른 TV 를 더 만들어보자

```
package main;

public class TelevisionTest {
    public static void main(String[] args) {
        Television myTv = new Television();
        myTv.channel = 7;
        myTv.volume = 9;
        myTv.onOff = true;

        Television yourTv = new Television();
        yourTv.channel = 9;
        yourTv.volume = 12;
        yourTv.onOff = true;

        ??????

        System.out.println("The channel of myTV is " +
            myTv.channel + ", and the volume is " + myTv.volume + ".");
        System.out.println("The channel of yourTV is " +
            yourTv.channel + ", and the volume is " + yourTv.volume + ".");
        ??????
    }
}
```

## 연습문제 2

- 텔레비전 클래스
  - Television 클래스에 print() 메서드 외에 다른 메서드를 넣고 호출해보자

```
package main;

public class Television {
    int channel;
    int volume;
    boolean onOff;

    void print() {
        System.out.println("The channel is " +
            channel + ", and the volume is " + volume);
    }

    ???
}
```

# 연습문제 3

- 텔레비전 클래스
  - 멤버 변수 onOff 정보를 반환하는 메서드를 추가하고 호출해보자

```
package main;

public class Television {
    int channel;
    int volume;
    boolean onOff;

    void print() {
        System.out.println("The channel is "
            + channel + ", and the volume is " + volume);
    }

    int getChannel() {
        return channel;
    }

    ????? ?????(){
        ?????
    }
}
```

## 연습문제 4

- 수학 클래스와 덧셈 메서드
  - 곱셈과 나눗셈도 구현하고 호출해보자

```
package main;

public class Math {
    int add(int x, int y) {
        return x + y;
    }
    ??? ???(???, ???){
        ???
    }
    ??? ???(???, ???){
        ???
    }
}
```



# 연습문제 5

- 자동차 클래스
  - 자동차 핸들 방향 및 감빡이와 관련된 멤버 변수를 넣고 메서드를 추가하자
  - 택시, 버스 클래스를 만들자
  - 클래스를 추가로 만들때 어떤 문제점이 있다고 생각되는가?

```
package main;

public class Car {
    String color;
    int speed;
    int gear;

    @Override
    public String toString() {
        return "Car [color=" + color + ", speed="
            + speed + ", gear=" + gear + "]";
    }
    void changeGear(int g) {
        gear = g;
    }
    void speedUp() {
        speed = speed + 10;
    }
    void speedDown() {
        speed = speed - 10;
    }
}
```

**감사합니다! XD**

