

어서와 *Java*는 처음이지!

제5장 클래스, 객체, 메소드

- 메소드
- 객체와 클래스

이번 장에서부터 드디어 객체
지향 프로그래밍이 시작되는 건
가요?

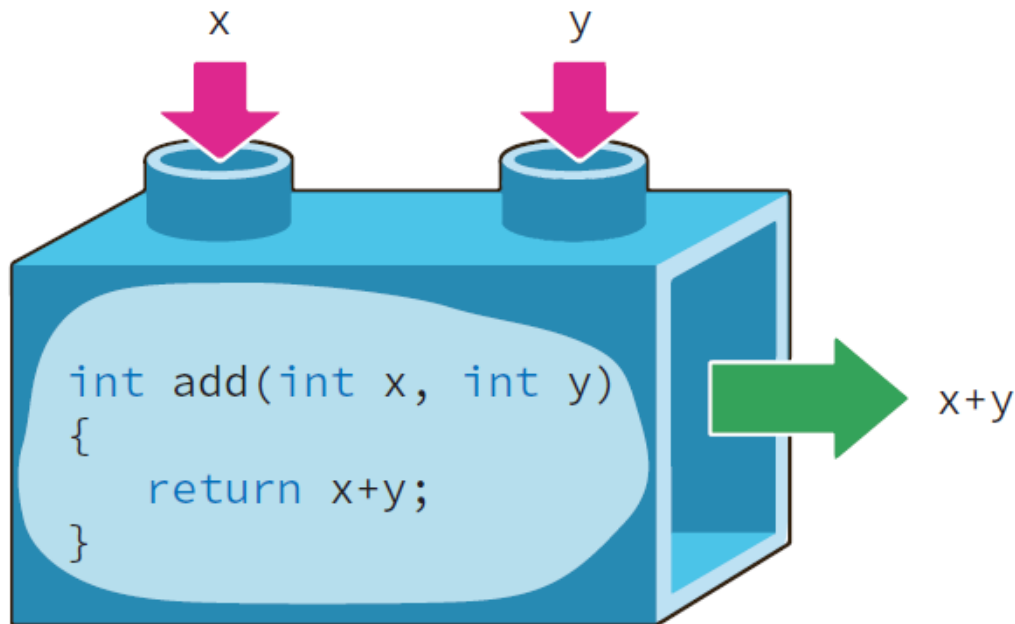
그렇습니다. 클래스, 객체, 메
소드는 자바 프로그래밍의 핵심입
니다. 이들 3가지에 대하여 확실히
이해하고 있어야 복잡한 프로
그램을 손쉽게 짤 수 있습니다.





메소드

- 메소드는 입력을 받아서 처리를 하고 결과를 반환하는 가상적인 상자와 같다.

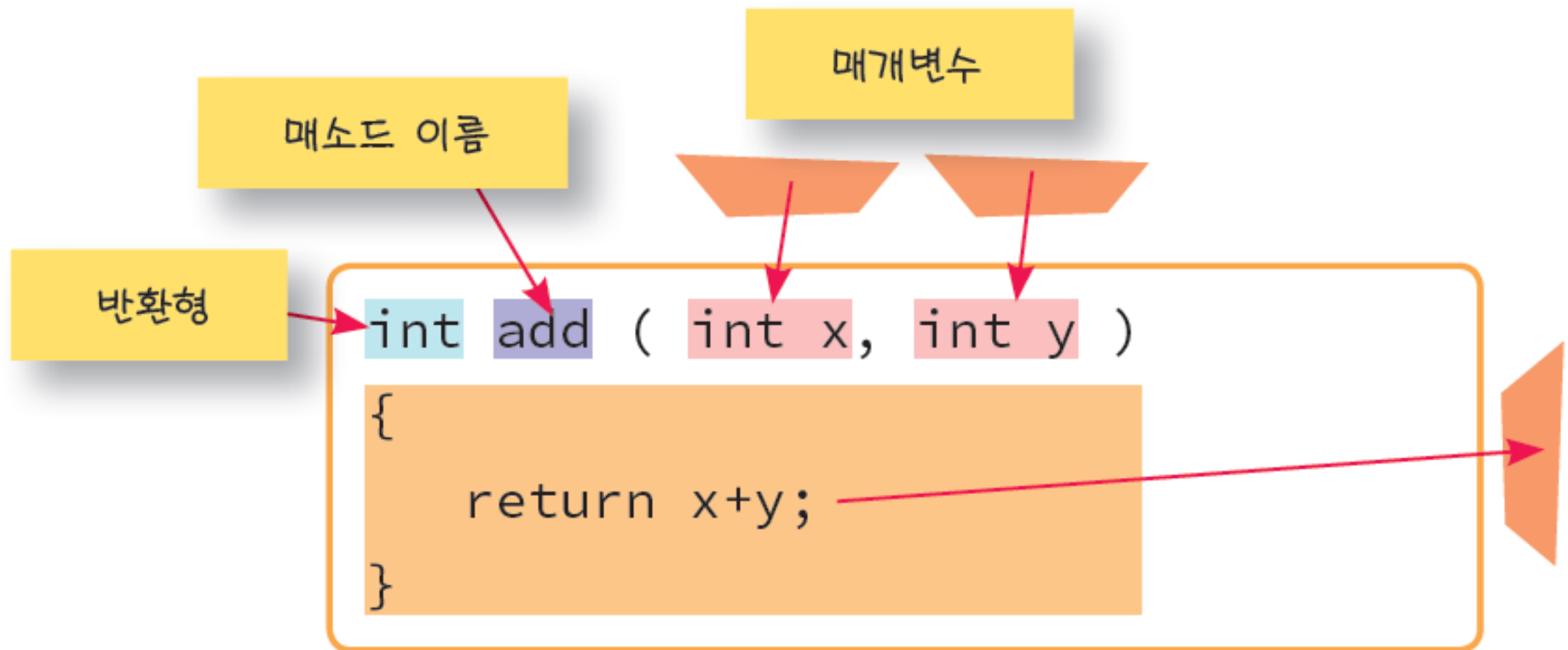


메소드는 입력을 받아서
처리결과를 반환하는 상자
로 생각하세요!





메소드의 구조





메소드의 반환값

형식

`return` 반환값;

`return` 뒤에 수식을 적으면
수식의 값이 반환됩니다.





인수와 매개 변수

- 메소드 호출시 전달하는 값을 인수(argument)
- 메소드에서 값을 받을 때 사용하는 변수를 매개 변수(parameter)

인수

매개변수

```
...  
int sum = add(25, 47);  
...
```

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



예제와 LAB#1

```
public class Sum {  
    static int sum(int a, int b) {  
        return a+b;  
    }  
    public static void main(String[] args) {  
        int a = sum(1,3);  
  
        System.out.println(a);  
    }  
}
```

사칙연산 각각에 대한 함수를 만들어 테스트하자!



예제

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

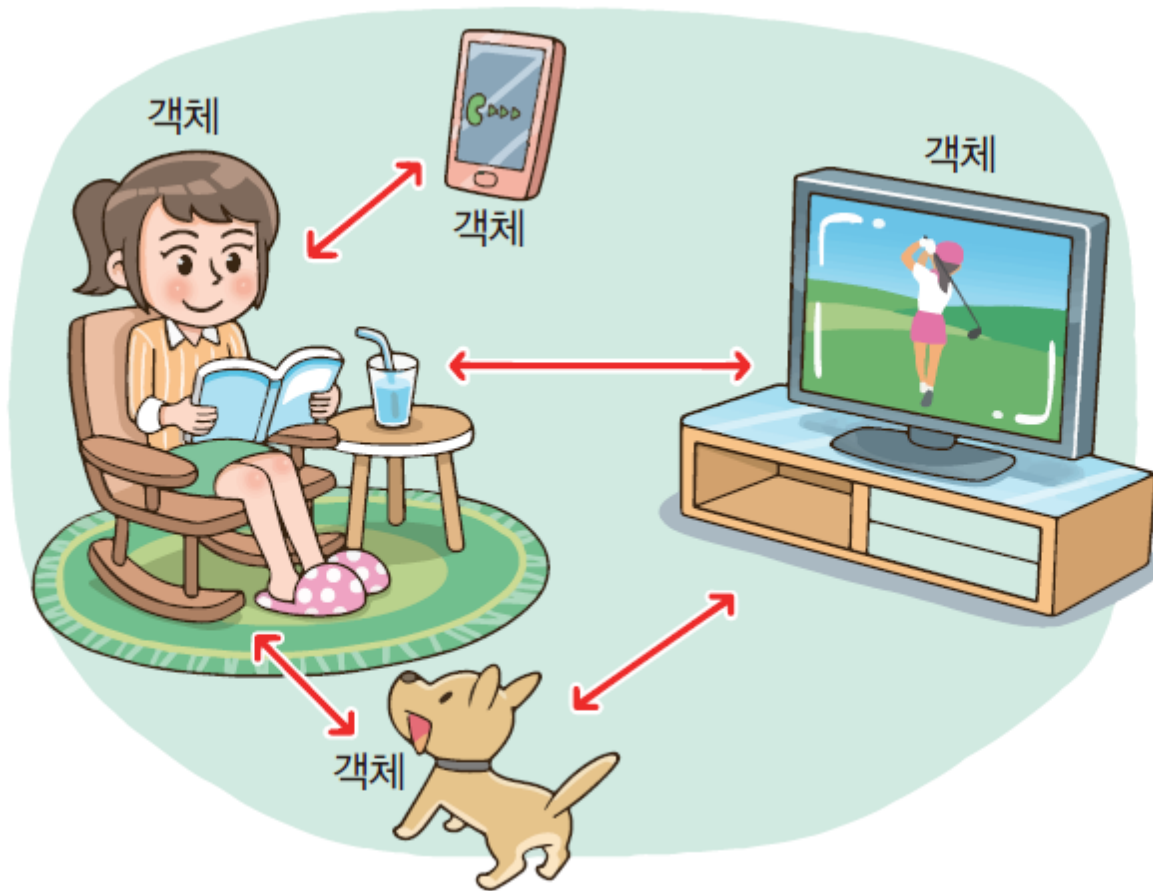
```
public class MathTest {  
    public static void main(String[] args) {  
        int sum;  
        Math obj = new Math();  
        sum = obj.add(2, 3);  
        System.out.println("2와 3의 합은 " + sum);  
        sum = obj.add(7, 8);  
        System.out.println("7와 8의 합은 " + sum);  
    }  
}
```

2와 3의 합은 5

7와 8의 합은 15



객체와 클래스



실제 세계는 객체들로
이루어져 있죠!





객체와 메시지



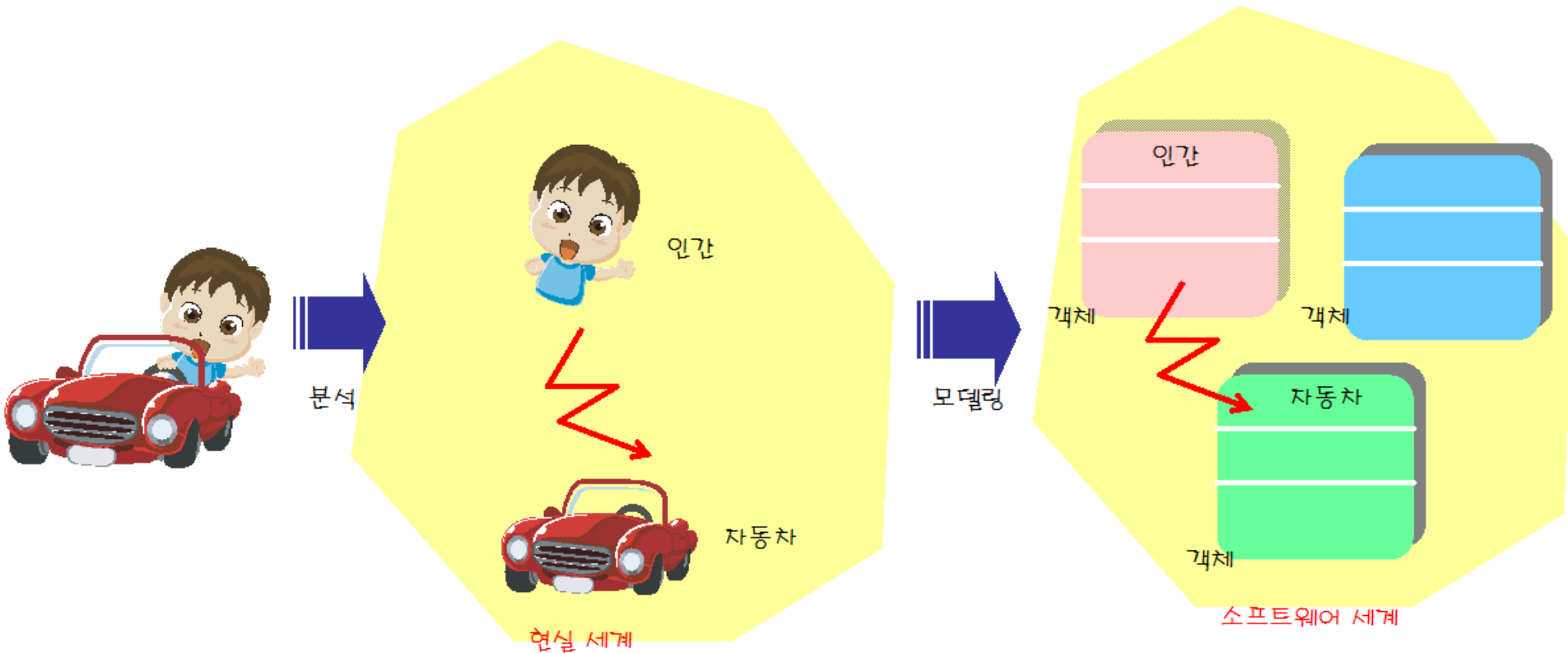
객체들은 메시지를 보내고 받으면서 상호 작용합니다.





객체 지향이란?

- 실제 세계를 모델링하여 소프트웨어를 개발하는 방법





객체



TV 객체

- 채널번호
- 볼륨
- 전원상태

상태

동작

- 켜기
- 끄기
- 채널변경하기
- 볼륨변경하기

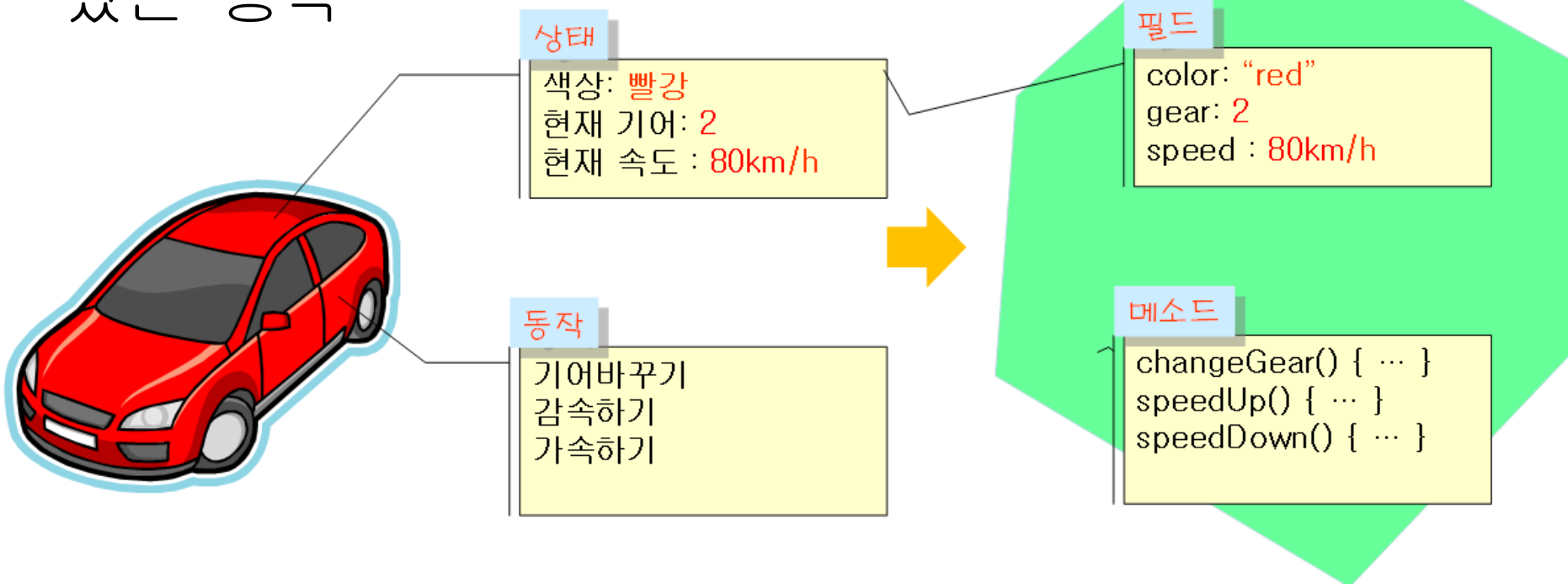
객체는 상태와 동작을 가지고 있습니다.





객체란?

- 객체(Object)는 상태와 동작을 가지고 있다.
- 객체의 상태(state)는 객체의 특징값(속성)이다.
- 객체의 동작(behavior) 또는 행동은 객체가 취할 수 있는 동작





필드와 메소드



TV 객체

- int channelNO;
- int volume;
- bool on Off

필드

- turnOn()
- turnOff()
- changeChannel()
- change Volume()

메소드

상태는 필드로, 동작은
메소드로 구현됩니다.





클래스

- 클래스(class): 객체를 만드는 설계도
- 클래스로부터 만들어지는 각각의 객체를 특별히 그 클래스의 인스턴스(instance)라고도 한다.

클래스는 객체를
찍어내는 틀과 같다



클래스



객체생성



객체



클래스의 구조

형식

class 클래스이름 {

자료형 필드1;
자료형 필드2;
....

메소드 정의
객체의 동작을 나타낸다.

반환형 메소드1() ... }
반환형 메소드2() ... }
...

필드 정의
객체의 속성을 나타낸다.

}



클래스의 예: 박스

○ 텔레비전

Television.java

```
01 public class Television {  
02     int channel;        // 채널 번호  
03     int volume;         // 볼륨  
04     boolean onOff;      // 전원 상태  
05 }
```

필드 정의
객체의 속성을 나타낸다.





예제: 객체 생성하기

TelevisionTest.java

```
01 public class TelevisionTest {  
02     public static void main(String[] args) {  
03         Television tv = new Television();  
04         tv.channel = 7;  
05         tv.volume = 9;  
06         tv.onOff = true;  
07         System.out.println("텔레비전의 채널은 " + tv.channel + "이고 볼륨은 "  
08             + tv.volume + "입니다.");  
09     }  
10 }
```

객체를 생성한다.

객체의 멤버에 접근할 때는
멤버 연산자(.)를 사용한다.

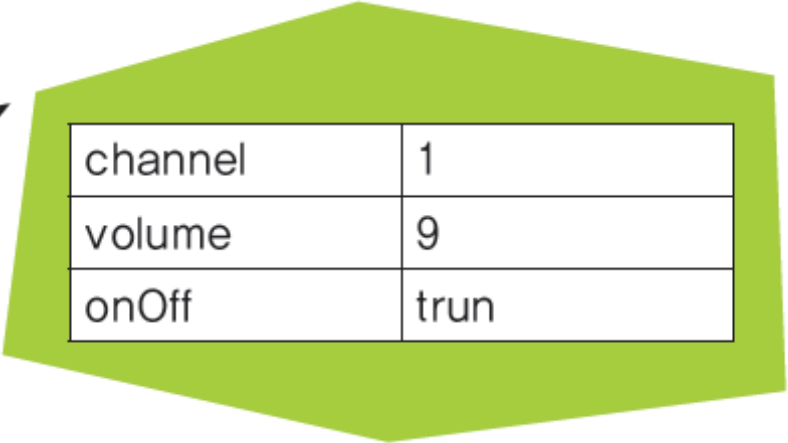
텔레비전의 채널은 7이고 볼륨은 9입니다.



당분간 소스 코드 다음
에는 소스 코드를 설명하는
그림이 등장할 것입니다!



TV



channel	1
volume	9
onOff	trun



객체의 필드와 메소드 사용

○ 도트(.) 연산자 사용!

형식

tv가 참조하는 객체로부터

```
tv.channel = 7;
```

channel이라는 필드에 접근



여러 개의 객체 생성하기

Television.java

```
01 public class Television {  
02     int channel;        // 채널 번호  
03     int volume;         // 볼륨  
04     boolean onOff;      // 전원 상태  
05 }
```

클래스

TelevisionTest.java

```
01 public class TelevisionTest {  
02     public static void main(String[] args) {  
03         Television myTv = new Television();  
04         myTv.channel = 7;  
05         myTv.volume = 10;  
06         myTv.onOff = true;  
07     }
```

myTv와 yourTv는 별개의
객체를 가리킨다.

객체



```
08     Television  yourTv = new Television();
09     yourTv.channel = 9;
10     yourTv.volume = 12;
11     yourTv.onOff = true;
12     System.out.println("나의 텔레비전의 채널은 " + myTv.channel +
13         "이고 볼륨은 " + myTv.volume + "입니다.");
14     System.out.println("너의 텔레비전의 채널은 " + yourTv.channel +
15         "이고 볼륨은 " + yourTv.volume + "입니다.");
16 }
17 }
```

나의 텔레비전의 채널은 7이고 볼륨은 10입니다.
너의 텔레비전의 채널은 9이고 볼륨은 12입니다.



실행 결과



my TV

channel	7
volume	9
onOff	trun



your TV

channel	9
volume	12
onOff	trun

여기서 중요한 것은 각 객체마다 별도의 변수 (필드)를 가진다는 점입니다.





예제

```
public class Television {  
    int channel;        // 채널 번호  
    int volume;         // 볼륨  
    boolean onOff;      // 전원 상태  
    void print() {  
        System.out.println("채널은 " + channel +  
            "이고 볼륨은 " + volume + "입니다.");  
    }  
}
```



예제

```
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();  
    }  
}
```

채널은 7이고 볼륨은 10입니다.
채널은 9이고 볼륨은 12입니다.



예제 설명



my TV

channel	7
volume	9
onOff	trun
print() {	
...	
}	

my Tv print()

myTv.print() 문장이 실행
되면 myTv 안의 print()
메소드가 실행되고 실행이
끝나면 myTv.print() 문장으로
되돌아옵니다.





예제

```
public class Television {  
    int channel; // 채널 번호  
    int volume; // 볼륨  
    boolean onOff; // 전원 상태  
    void print() {  
        System.out.println("채널은 " + channel +  
            "이고 볼륨은 " + volume + "입니다.");  
    }  
    int getChannel() {  
        return channel;  
    }  
}
```



예제

```
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("현재 채널은 " + ch  
            + "입니다.");  
    }  
}
```

실행결과

현재 채널은 7입니다.

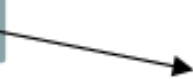




예제 설명



myTv



<i>channel</i>	<i>7</i>
<i>volume</i>	<i>9</i>
<i>onOff</i>	<i>true</i>
<i>void print() {</i> ... <i>}</i>	
<i>int getChannel() {</i> ... <i>}</i>	

ch = myTv.getChannel()





예제

```
public class Television {  
    int channel; // 채널 번호  
    int volume; // 볼륨  
    boolean onOff; // 전원 상태  
    void print() {  
        System.out.println("채널은 " + channel + "이고 볼륨은  
" + volume + "입니다.");  
    }  
    int getChannel() {  
        return channel;  
    }  
    void setChannel(int ch) {  
        channel = ch;  
    }  
}
```



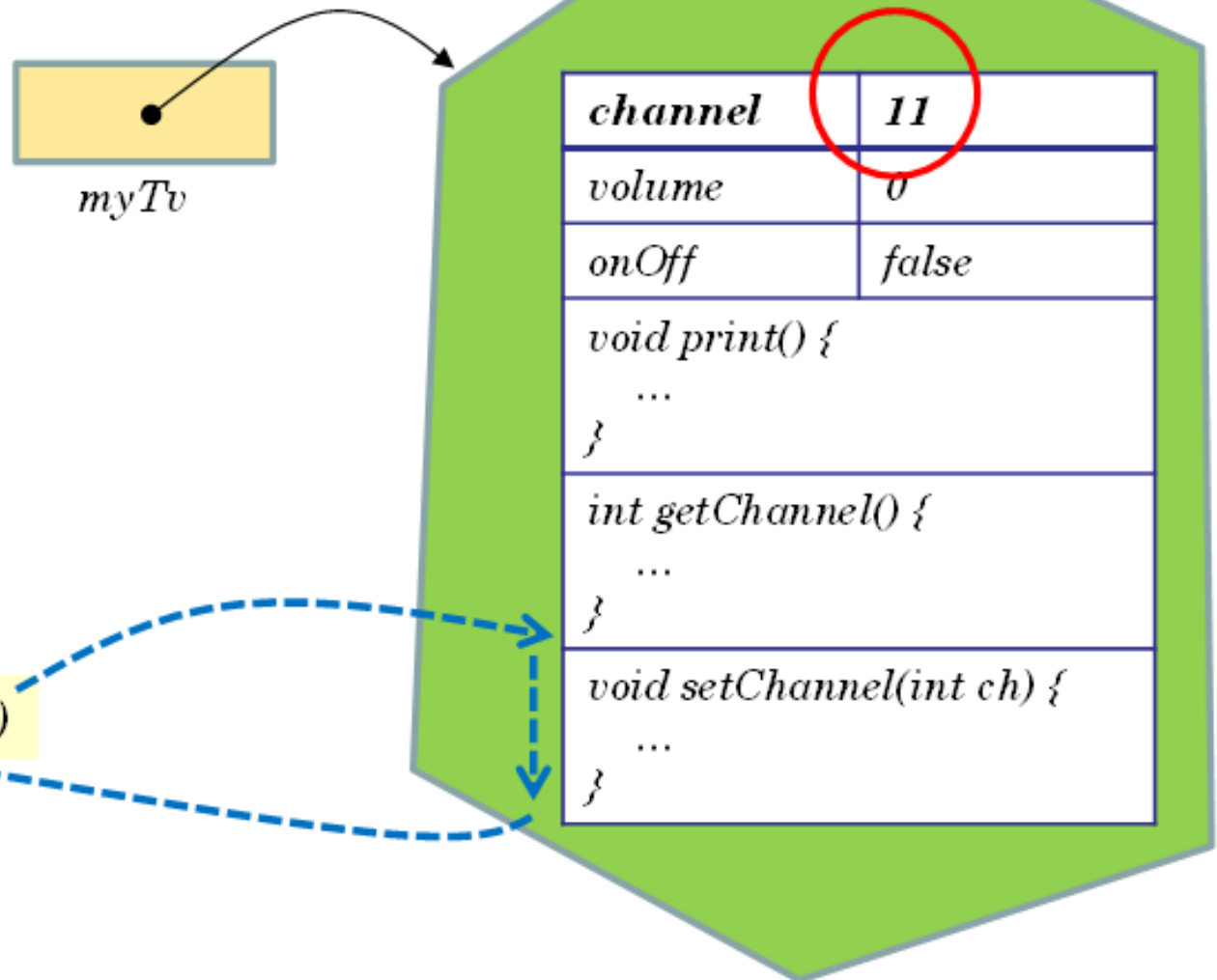
예제

```
public class TelevisionTest {  
    public static void main(String[] args) {  
        Television myTv = new Television();  
        myTv.setChannel(11);  
        int ch = myTv.getChannel();  
        System.out.println("현재 채널은 " + ch +  
"입니다.");  
    }  
}
```

현재 채널은 11입니다.



예제 설명





LAB#2

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

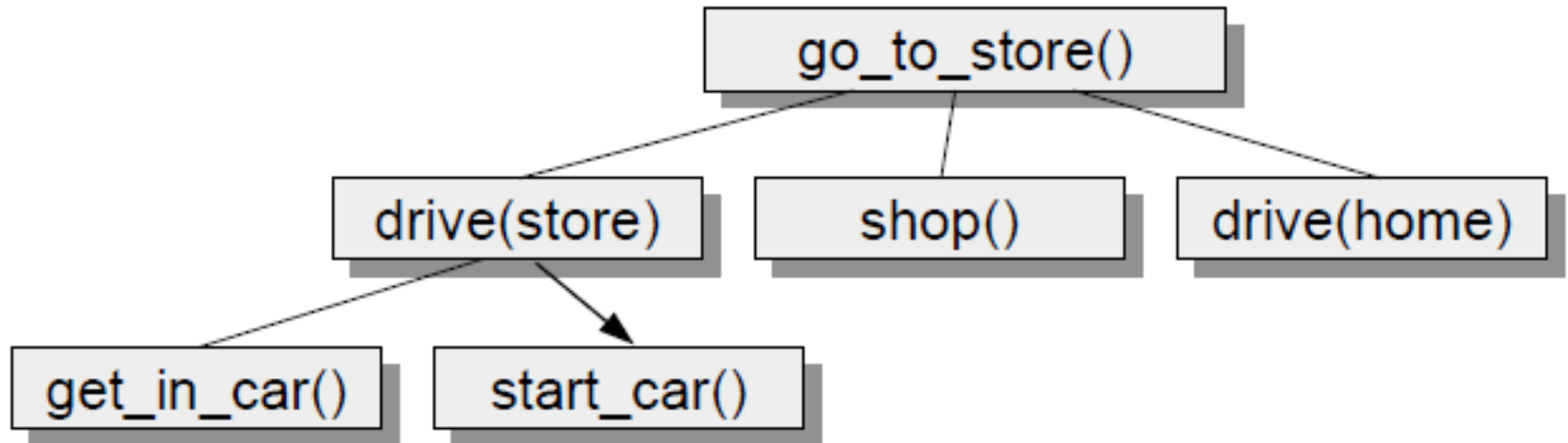
```
public class MathTest {  
    public static void main(String[] args) {  
        int sum;  
        Math obj = new Math();  
        sum = obj.add(2, 3);  
        System.out.println("2와 3의 합은 " + sum);  
        sum = obj.add(7, 8);  
        System.out.println("7와 8의 합은 " + sum);  
    }  
}
```

사칙연산 각각에 대해 클래스를 만들고, 메인 함수에서 사용해 보자.



절차 지향과 객체 지향

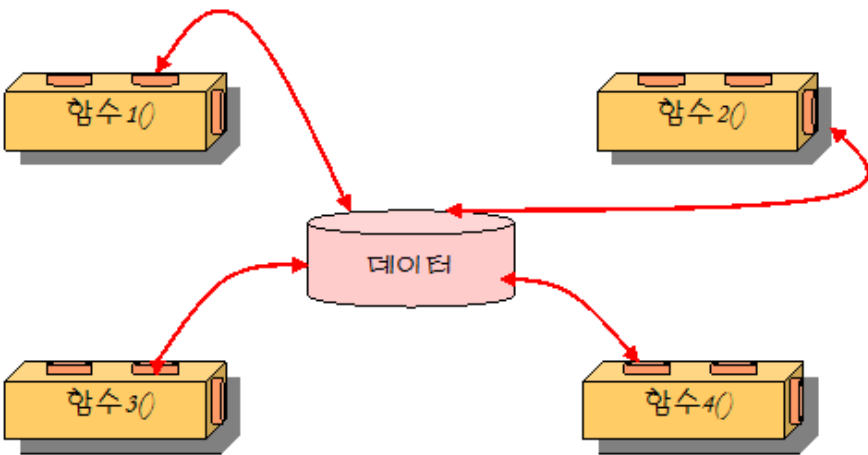
- 절차 지향 프로그래밍(procedural programming): 문제를 해결하는 절차를 중요하게 생각하는 방법



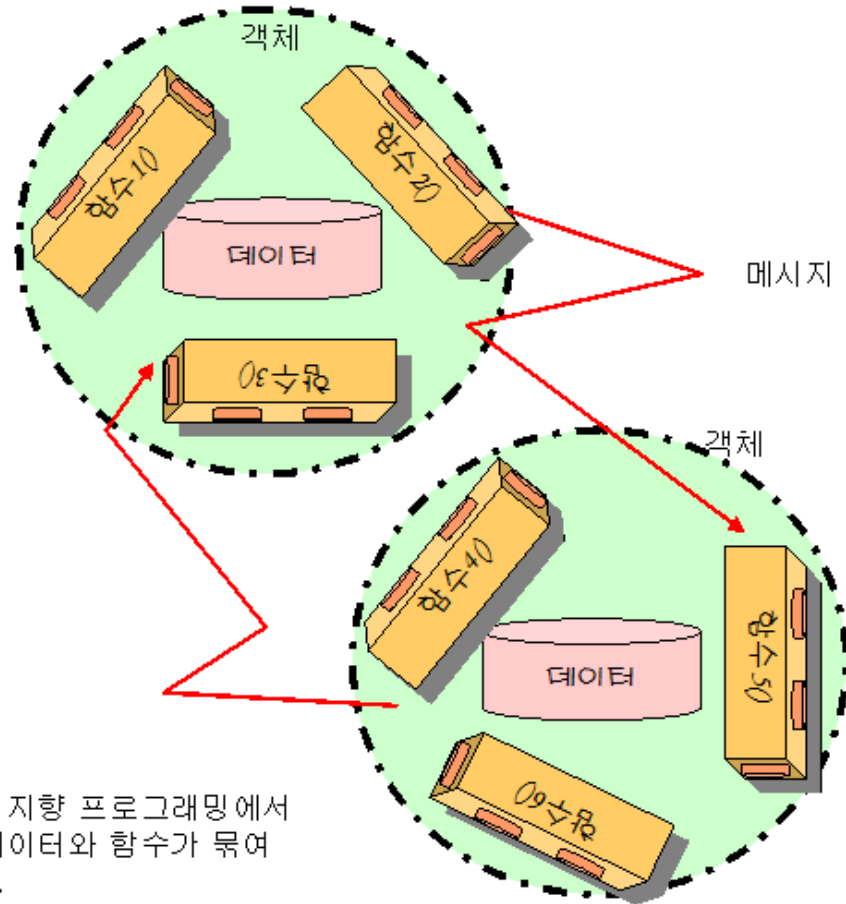
- 객체 지향 프로그래밍(Object-Oriented Programming): 데이터와 절차를 하나의 덩어리(객체)로 묶어서 생각하는 방법이다.



절차 지향과 객체 지향



절차 지향 프로그래밍에서는
데이터와 함수가 묶여
있지 않다.

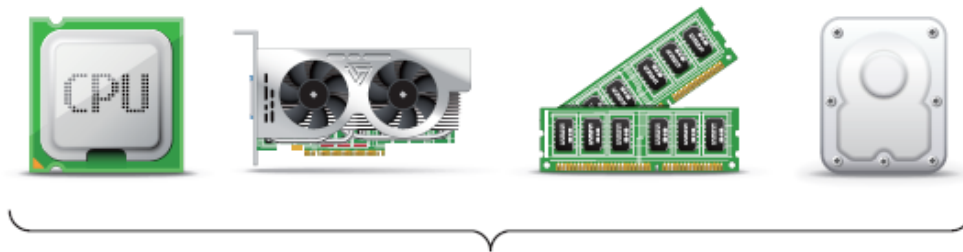


객체 지향 프로그래밍에서는
데이터와 함수가 묶여
있다.

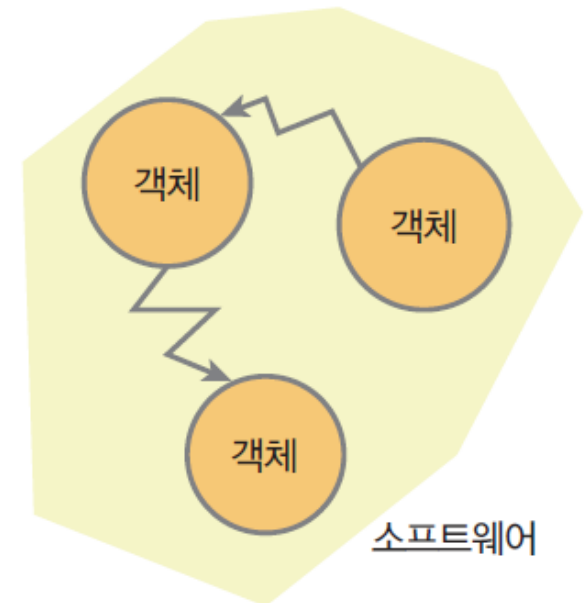


객체 지향 방법

- 객체 지향으로 소프트웨어를 작성하는 것은 컴퓨터 하드웨어 부품을 구입하여서 컴퓨터를 조립하는 것과 비슷하다.



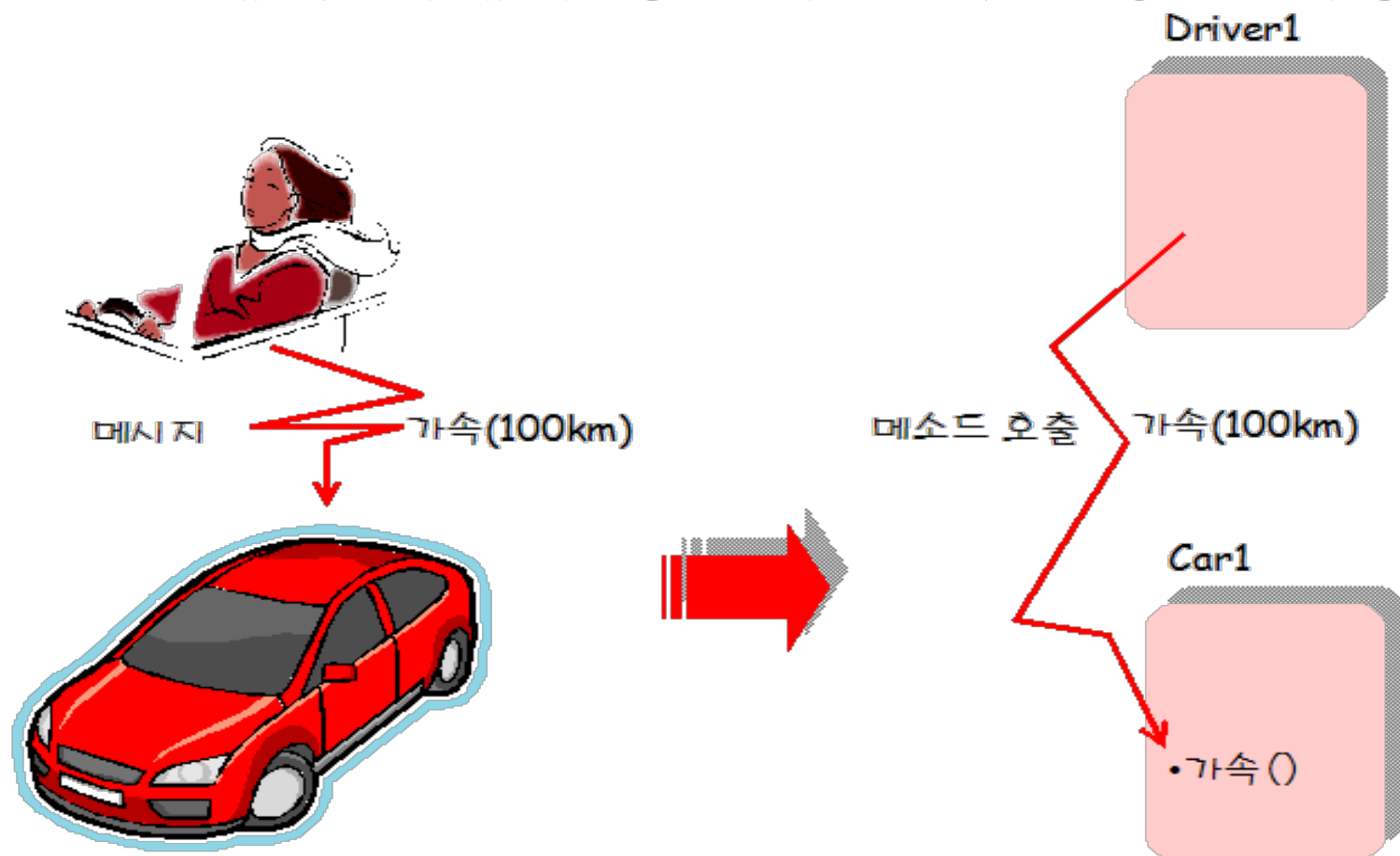
부품을 조립하여 제품을 만들듯이
객체를 조합하여 소프트웨어를 만든다





메시지

- 소프트웨어 객체는 메시지(message)를 통해 다른 소프트웨어 객체와 통신하고 서로 상호 작용한다.





중간 점검 및 LAB#3

1. 객체들은 _____전달을 통해서 서로 간에 상호 작용을 한다.
2. 객체 지향 프로그래밍은 _____들을 조합하여서 프로그램을 작성하는 기법이다.
3. 자동차 객체에서 생각할 수 있는 메시지와 매개 변수에 대하여 나열하여 보라.





Q & A

