

COMP217

Java Programming

Summer 2018

Day 7

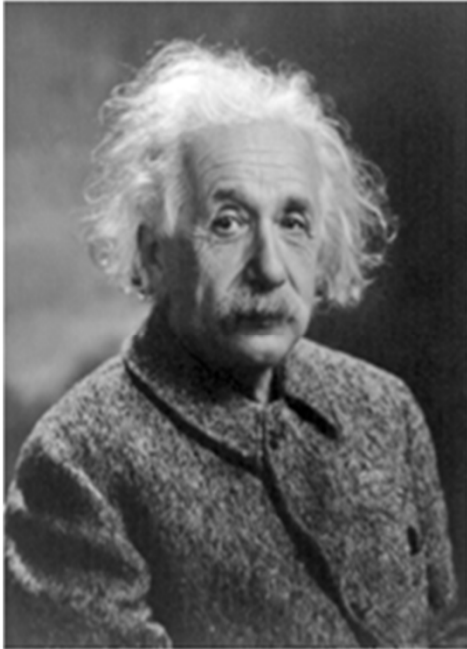
Class and Objects

Goals

- 이번 주에 배우게 될 내용 :
 - 객체지향이란?
 - 객체
 - 메시지
 - 클래스
 - 객체 지향의 장점
 - String 클래스
- 파워 자바 7장 (클래스와 객체)

실제 세계의 객체

human



car



tree

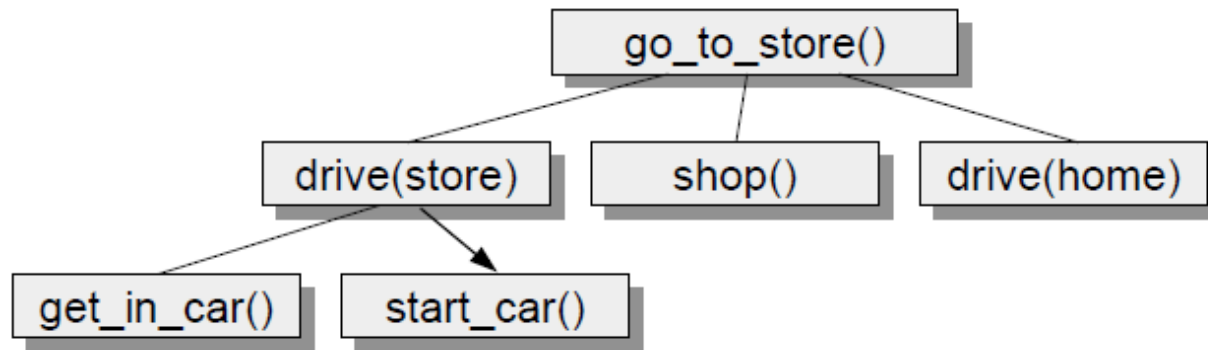


- 실제 세계는 객체들로 이루어진다.
- 객체(object)
 - 현실 세계의 사물이나 개념을 시스템에서 이용하기 위해
현실 세계를 자연스럽게 표현하여 손쉽게 이용할 수 있도록
만든 소프트웨어 모델

절차 지향과 객체 지향

- 절차 지향 프로그래밍(procedural programming)

- 문제를 해결하는 절차를 중요하게 생각하는 방법



- 객체 지향 프로그래밍(Object-Oriented Programming)

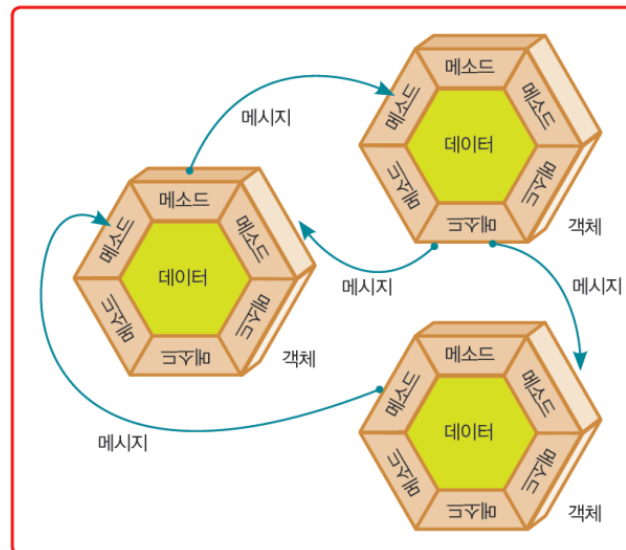
- 데이터와 절차를 하나의 덩어리(객체)로 묶어서 생각하는 방법
- 실제 세계를 모델링하여 소프트웨어를 개발하는 방법



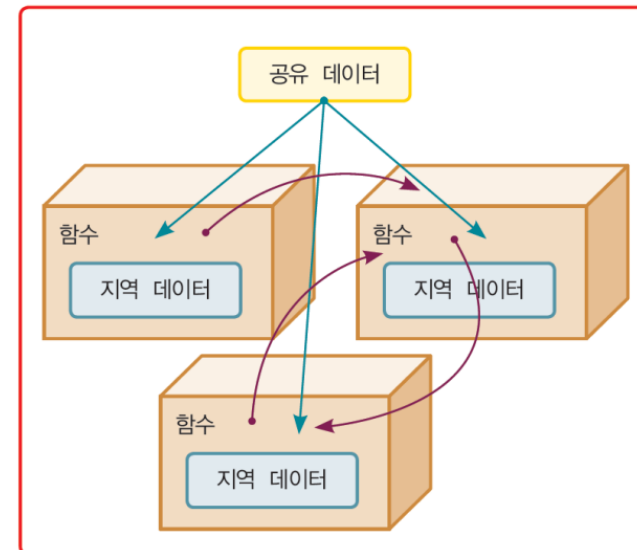
절차 지향과 객체 지향

- 객체지향 프로그래밍 언어
 - 원조는 시뮬라(simula)라는 프로그래밍 언어
 - 클래스라는 개념을 처음으로 도입
- 객체지향 프로그래밍
 - 클래스(class)를 생성하고
 - 클래스로부터 객체(object)를 만들어 객체 간의 상호작용을 이용하여 주어진 문제를 해결하는 프로그래밍 방식
 - 데이터인 필드와 절차인 메소드를 하나로 묶은 클래스 단위의 프로그램


객체지향 프로그래밍

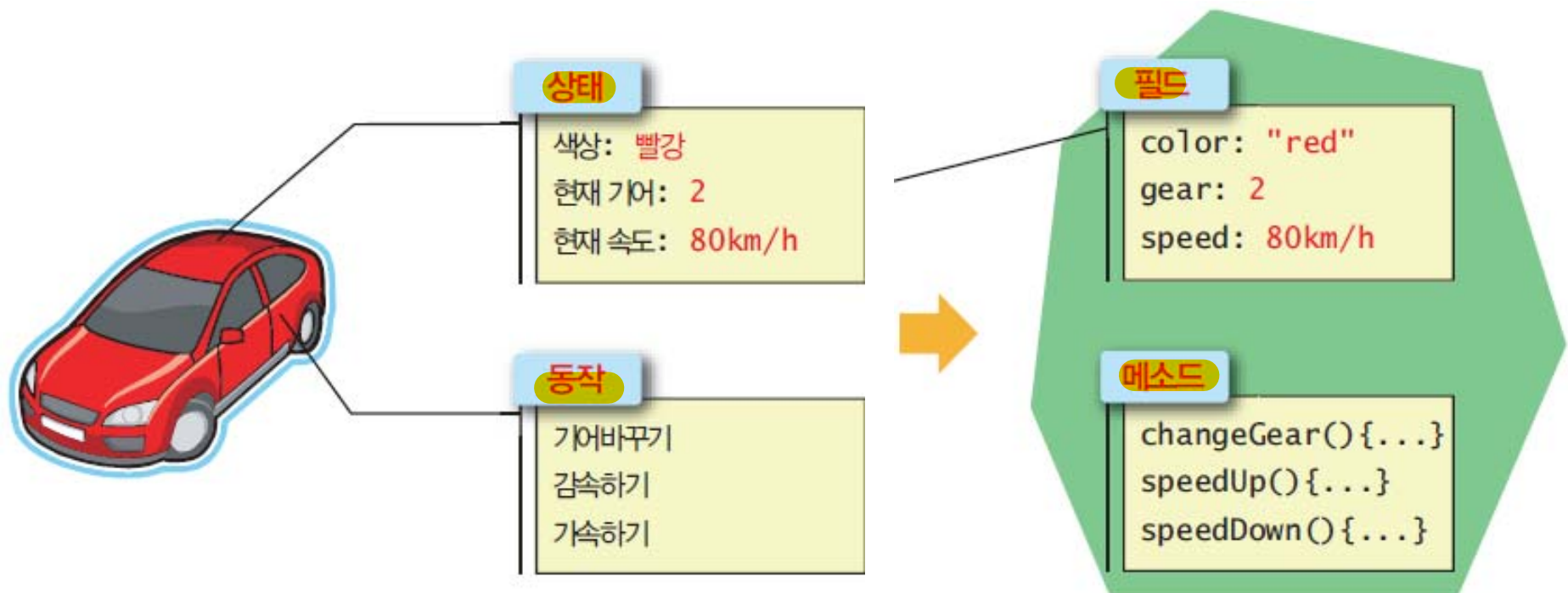


절차적 프로그래밍



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



추상화(Abstraction)

- 현실 세계의 사실에서 주어진 문제의 중요한 측면을 주목하여 설명하는 방식
- 실세계의 객체에서 불필요한 부분을 제거하여 필요한 부분만을 간결하고 이해하기 쉬운 클래스로 만드는 작업
- 속성과 행동으로 추상화를 실현




필요한 부분을 속성과 행동으로 구체화

```
public class Car {  
    private int maxSpeed;  
    public String brandName;  
    public int speed;  
    public static int prodNum;  
  
    //메소드 오버로딩  
    public int speedUp() {  
        return speed += 5;  
    }  
    public int speedUp(int speed) {  
        this.speed += speed;  
        return this.speed;  
    }  
    public int speedDown() {  
        return speed -= 5;  
    }  
    public int speedDown(int speed) {  
        return this.speed -= speed;  
    }  
  
    //setter와 getter  
    public int getMaxSpeed() {  
        return maxSpeed;  
    }  
    public void setMaxSpeed(int maxSpeed) {  
        this.maxSpeed = maxSpeed;  
    }  
}
```

중간 점검 문제

- 실제 세계의 객체에서 객체의 가능한 상태와 객체가 수행할 수 있는 동작을 정리하여 보자.

| 객체 | 상태  | 동작 |
|-----|--|----|
| 전구 | | |
| 라디오 | | |
| 강아지 | | |
| 자전거 | | |
| 사자 | | |

메시지

- 소프트웨어 객체는 메시지(message)를 통해 다른 소프트웨어 객체와 통신하고 서로 상호 작용한다.
- 메시지 전달 = 메소드 호출
 - 메시지에 대한 추가정보는 매개변수(parameter)형태로 전달

```
Car c = new Car( );
```

```
c.setSpeed(100);
```

메시지를 받는 객체


메시지 이름

메시지의 매개변수



그림 7-8 . 메시지 전달

중간 점검 문제

1. 객체들은 _____ 전달을 통해서 서로 간에 상호 작용을 한다. 
2. 자동차 객체에서 생각할 수 있는 메시지와 매개 변수에 대하여 나열하여 보라.

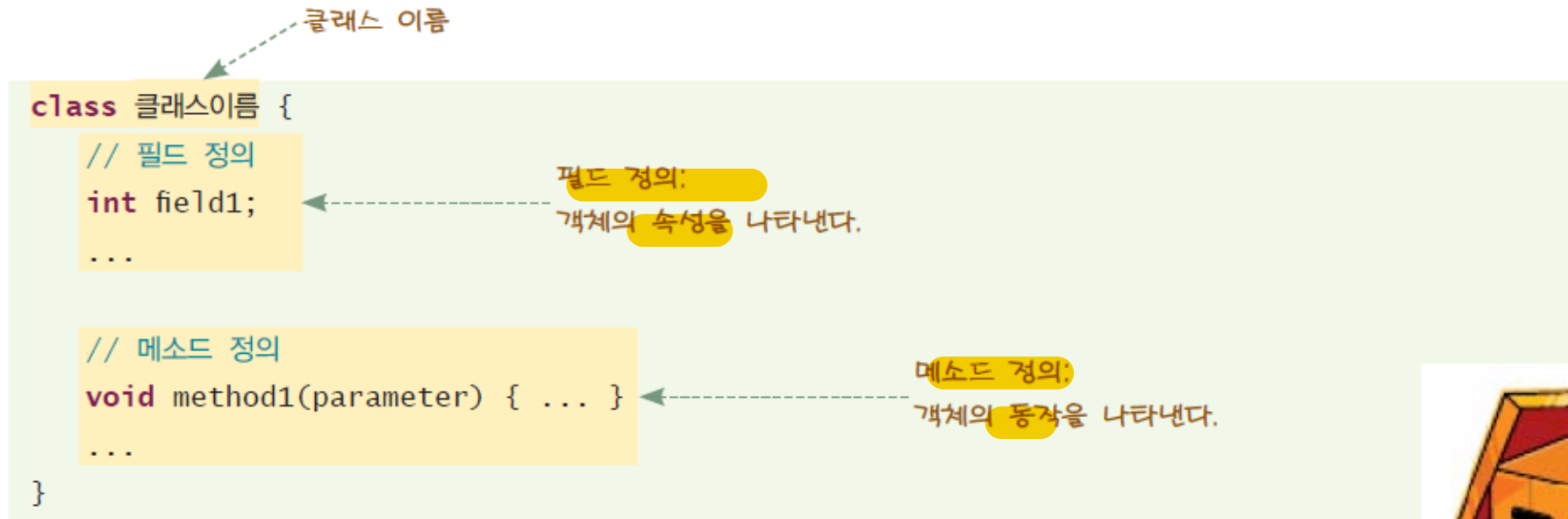
클래스

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



그림 7-9 . 객체를 클래스라는 설계도로 생성된다.

클래스의 구조와 예



```
class Box1 {  
    int width;  
    int length;  
    int height;  
}
```

필드 정의:
객체의 속성을 나타낸다.

```
class Box2 {  
    int width;  
    int length;  
    int height;  
    int calVolume()  
    {  
        return width*length*height;  
    }  
}
```

메소드 정의:
객체의 동작을 나타낸다.

클래스의 예: 자동차



- 클래스로부터 객체를 생성하려면?!

Car myCar = **new** Car();



객체의 생성

```
Car    myCar;           // ❶ 참조 변수를 선언  
myCar = new Car();      // ❷ 객체를 생성하고 ❸ 참조값을 myCar에 저장
```

- ❶ 참조 변수 선언 - Car타입의 객체를 참조할 수 있는 변수 myCar를 선언한다.



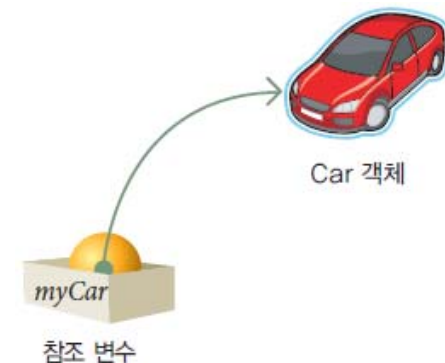
위의 문장으로 객체가 생성되는 것은 아님!!!
객체를 가리키는 참조값을 담을 수 있는 변수만 생성됨.

- ❷ 객체 생성 - new 연산자를 이용하여 객체를 생성하고 객체 참조값을 반환한다.



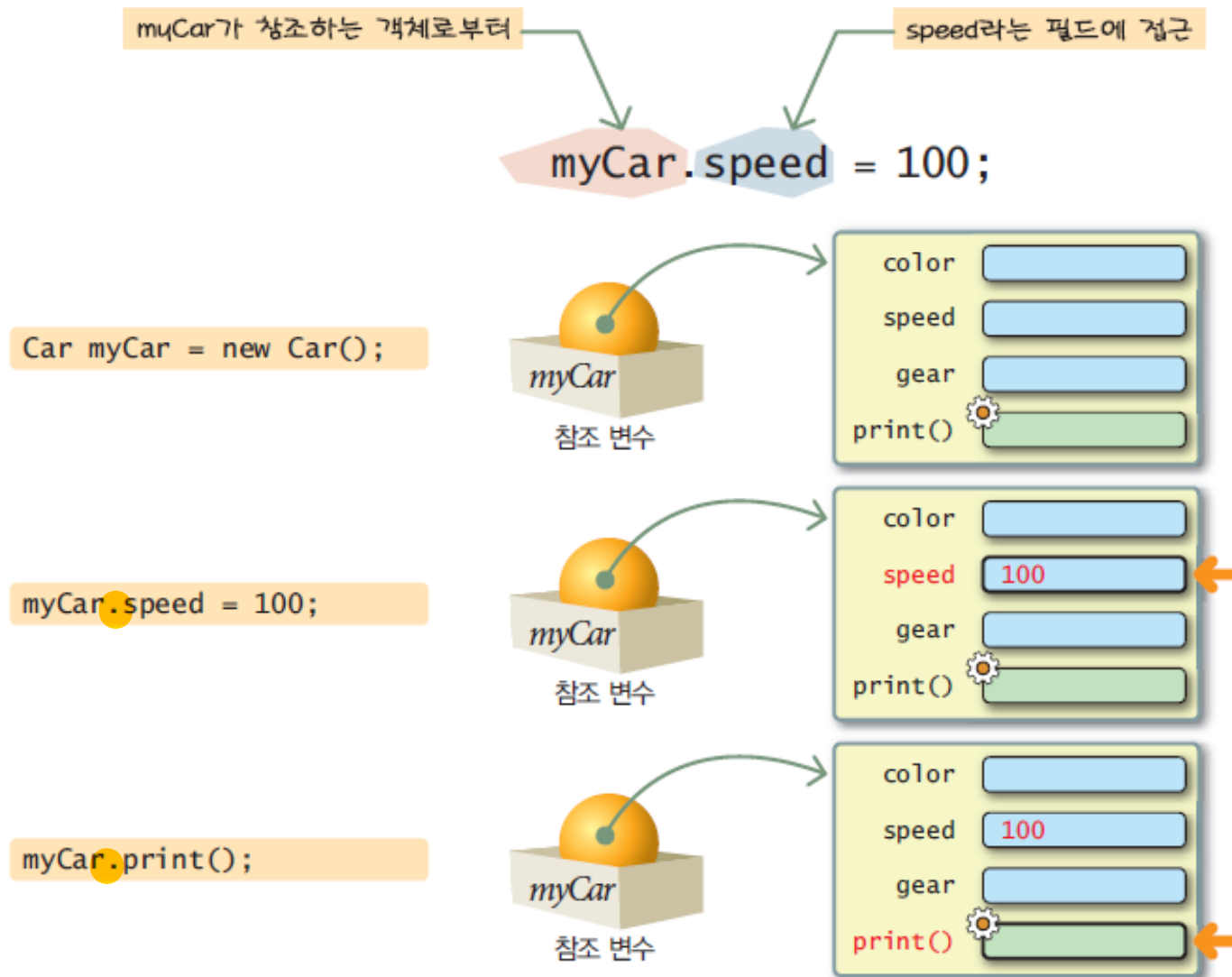
Car 객체

- ❸ 참조 변수와 객체의 연결 - 생성된 새로운 객체의 참조값을 myCar 라는 참조 변수에 대입한다.



객체의 필드와 메소드 접근

- 도트(.) 연산자 사용!



객체 생성 코드



참조 변수

| | |
|---------|----------------------------------|
| color | <input type="text" value="red"/> |
| speed | <input type="text" value="0"/> |
| gear | <input type="text" value="1"/> |
| print() | <input type="button" value="⚙"/> |



클래스



객체

```
01 class Car {
02     // 필드 정의
03     String color;    // 색상
04     int speed;       // 현재 속도
05     int gear;        // 현재 기어
06     void print() {
07         System.out.println("( " + color + ", " + speed + ", " + gear + " )");
08     }
09 }
10 class CarTest {
11     public static void main(String[] args) {
12         Car myCar = new Car(); ← 객체 생성
13         myCar.color = "red";
14         myCar.speed = 0; ← 객체의 필드 변경
15         myCar.gear = 1;
16         myCar.print(); ← 객체의 메소드 호출
17     }
18 }
```


객체를 하나 더 생성하는 코드



클래스

```
01 class Car {
02     // 필드 정의
03     String color;    // 색상
04     int speed;       // 현재 속도
05     int gear;        // 현재 기어
06     void print() {
07         System.out.println("( " + color + ", " + speed + ", " + gear + " )");
08     }
09 }
10 class CarTest {
11     public static void main(String[] args) {
12
13         Car myCar = new Car();    // 객체 생성
14         myCar.color = "red";      // 객체의 필드 변경
15         myCar.speed = 0;          // 객체의 필드 변경
16         myCar.gear = 1;           // 객체의 필드 변경
17         myCar.print();            // 객체의 메소드 호출
18
19         Car yourCar = new Car();   // 객체 생성
20         yourCar.color = "blue";    // 객체의 필드 변경
21         yourCar.speed = 60;        // 객체의 필드 변경
22         yourCar.gear = 3;          // 객체의 필드 변경
23         yourCar.print();           // 객체의 메소드 호출
24     }
25 }
```



객체



객체



참조 변수

| | |
|---------|----------------------------------|
| color | <input type="text" value="red"/> |
| speed | <input type="text" value="0"/> |
| gear | <input type="text" value="1"/> |
| print() | <input type="button" value="⚙"/> |



참조 변수

| | |
|---------|-----------------------------------|
| color | <input type="text" value="blue"/> |
| speed | <input type="text" value="60"/> |
| gear | <input type="text" value="3"/> |
| print() | <input type="button" value="⚙"/> |

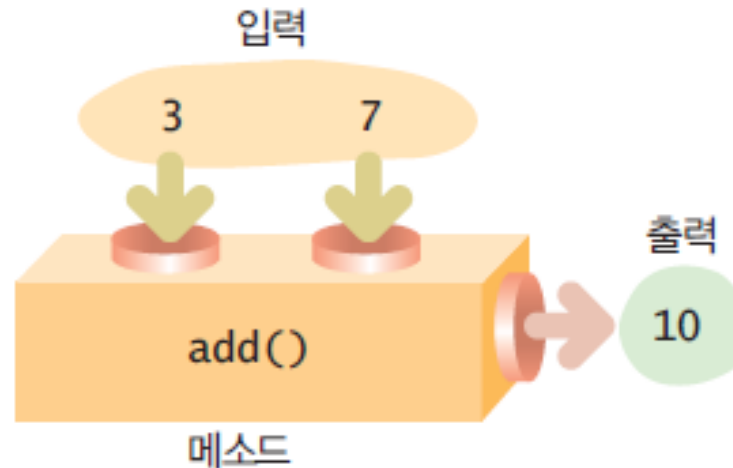
중간 점검 문제



1. 객체들을 만드는 설계도에 해당되는 것이 _____이다.
2. 같은 종류의 객체가 여러 개 생성될 때 각 객체의 필드와 메소드는 공유되는가? 아니면 각 객체마다 별도로 만들어지는가?
3. 클래스 선언 시에 클래스 안에 포함되는 것은 _____과 _____이다.
4. 객체의 멤버에 접근하는데 사용되는 연산자는 _____이다.
5. 각 객체마다 별도로 가지고 있는 것은 클래스의 _____이다.
6. 상품의 재고를 나타내는 클래스를 작성하여 보자. 클래스 안에 상품 번호, 재고수량이 필드로 저장되고 재고를 증가, 감소하는 메소드를 작성하여 보라.

메소드

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



접근 제어 지정자: public, private와 같은 접근 제어를 나타낸다.

반환형: 메소드로부터 반환되는 데이터의 타입

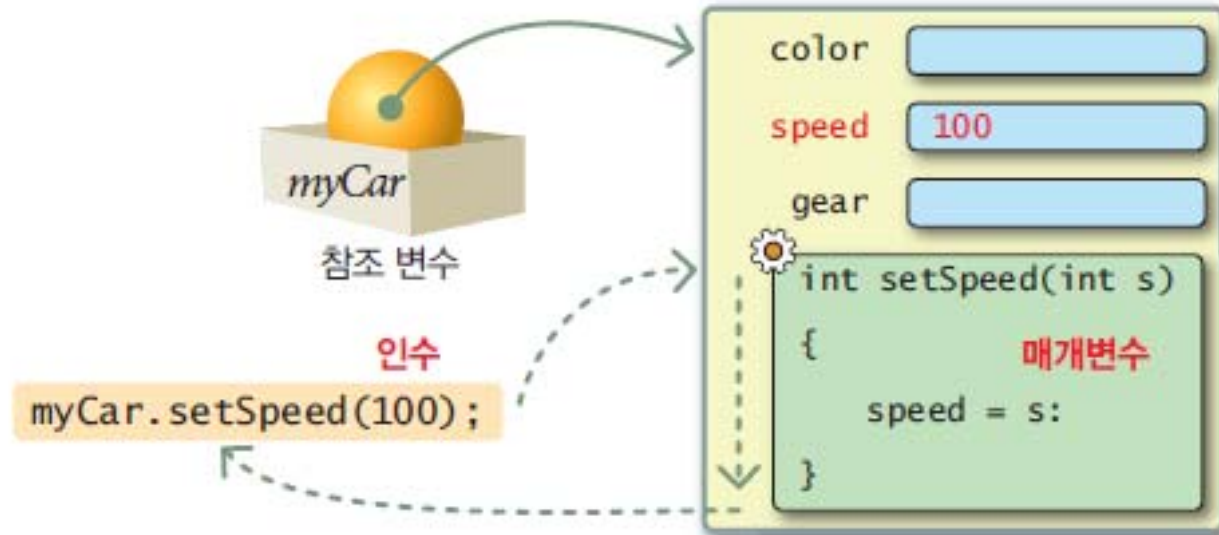
```
public int add(int x, int y)
{
    return x + y;
}
```

매개변수 목록: 메소드의 입력

메소드의 몸체로서 합을 계산한다.
메소드가 값을 반환할 때 return 사용

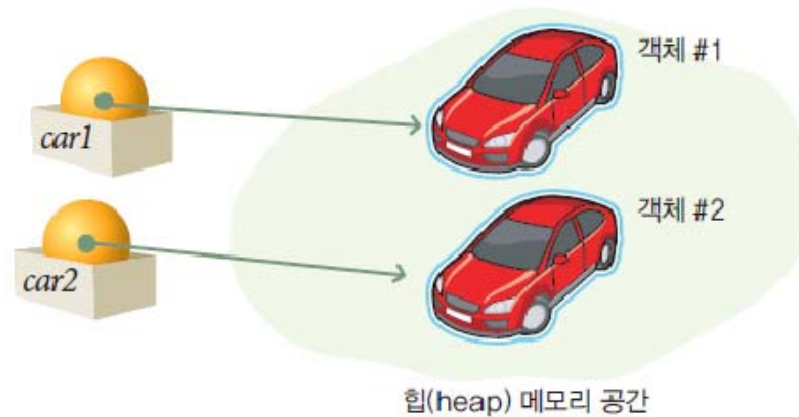
인수와 매개 변수

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

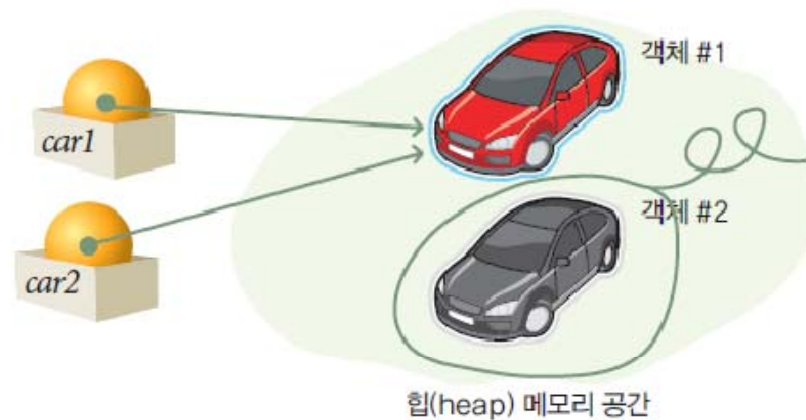


객체의 생성과 참조 값 복사

```
Car car1 = new Car(); // 첫 번째 객체  
Car car2 = new Car(); // 두 번째 객체
```



```
car2 = car1; // car1과 car2는 같은 객체를 가리킨다.  
// car2가 가리켰던 객체는 쓰레기 수집기에 의하여 수거된다.
```



객체의 소멸

```
car1 = null; // 객체 1은 아직도 활성화된 참조가 있기 때문에 소멸되지 않는다.
```



- 어떤 참조 변수도 객체를 참조하지 않을때, 그 객체는 가비지 콜렉션(garbage collection)의 대상
- **가비지 콜렉션**
 - 가비지 콜렉터가 프로그램이 사용하는 메모리 영역을 조사하여 더 이상 사용되지 않는 객체들을 수집하여 삭제하는 작업

중간 점검 문제

1. 기초 변수와 참조 변수의 차이점은 무엇인가?
2. 하나의 참조 변수가 다른 참조 변수로 대입되면 어떤 일이 발생하는가?
3. 객체를 소멸시키려면 어떻게 하면 되는가?

String 클래스

- 문자열은 String 이라는 클래스를 사용
 - 자바에서 문자열은 기초 자료형이 아님
- 객체의 생성
 - `String s = new String("Hello, World");`
 - `String s = "Hello, World";`
 - 원래 객체의 생성은 new 연산자를 사용해야 하지만 String 클래스는 new 연산자를 사용하지 않고도 객체 생성 가능
- 객체의 메서드 호출
 - `int size = s.length();` //size는 12가 된다.
 - 객체의 메서드를 사용하려면 도트 연산자(dot operator) 사용

String 클래스의 메소드

| | |
|---------|--|
| char | charAt(int index) 지정된 인덱스에 있는 문자를 반환한다. |
| int | compareTo(String anotherString) 사전적 순서로 문자열을 비교한다. 앞에 있으면 -1, 같으면 0, 뒤에 있으면 1이 반환된다. |
| String | concat(String str) 주어진 문자열을 현재의 문자열 뒤에 붙인다. |
| boolean | equals(Object anObject) 주어진 객체와 현재의 문자열을 비교한다. |
| boolean | equalsIgnoreCase(String anotherString) 대소문자를 무시하고 비교한다. |
| boolean | isEmpty() length()가 0이면 true를 반환한다. |
| int | length() 현재 문자열의 길이를 반환한다. |
| String | replace(char oldChar, char newChar) 주어진 문자열에서 oldChar를 newChar로 변경한, 새로운 문자열을 생성하여 반환한다. |
| String | substring(int beginIndex, int endIndex) 현재 문자열의 일부를 반환한다. |
| String | toLowerCase() 문자열의 문자들을 모두 소문자로 변경한다. |
| String | toUpperCase() 문자열의 문자들을 모두 대문자로 변경한다. |

예제

```
1 public class Test{
2     public static void main(String args []){
3         String s1 = "A barking dog";
4         String s2 = "A bc";
5         String s3 = " never Bites!";
6
7         System.out.println(s1.charAt(0));
8         System.out.println(s1.compareTo(s2));
9         System.out.println(s2.length());
10
11        String s4 = s1.concat(s3);
12        System.out.println(s4);
13        System.out.println(s1+s3);
14
15        String s5 = s1.replace('A', 'a');
16        System.out.println(s1.equals(s5));
17        System.out.println(s1.equalsIgnoreCase(s5));
18        System.out.println(s3.substring(1,6));
19        System.out.println(s4.toUpperCase());
20    }
21 }
```

```
A
-2
4
A barking dog never Bites!
A barking dog never Bites!
false
true
never
A BARKING DOG NEVER BITES!
```

숫자를 문자열로 변환

```
1 public class Test{  
2     public static void main(String args []){  
3         String add = "오백 : ";  
4         int x = 500;  
5         String result = add+x;  
6         System.out.println(result);  
7         System.out.println("덧셈 ? : "+1+5);  
8         System.out.println(1+5+" : 덧셈 결과");  
9         System.out.println("덧셈 : "+(1+5));  
10    }  
11 }
```

문자열을 숫자로 변환

- 문자열 "123"을 숫자 123으로 변환
 - `int i = Integer.parseInt("123");`
- 래퍼 클래스(wrapper class)를 사용한다.
 - `double d = Double.parseDouble("3.141592");`

| 기초 자료형 | 래퍼 클래스 |
|---------|-----------|
| byte | Byte |
| short | Short |
| int | Integer |
| long | Long |
| float | Float |
| double | Double |
| char | Character |
| boolean | Boolean |
| void | Void |



Integer 클래스

| 반환값 | 메소드 이름 | 설명 |
|--------|------------------------------------|-------------------------------|
| byte | <code>byteValue()</code> | int형을 byte형으로 변환한다. |
| double | <code>doubleValue()</code> | int형을 double형으로 변환한다. |
| float | <code>floatValue()</code> | int형을 float형으로 변환한다. |
| int | <code>parseInt(String s)</code> | 문자열을 int형으로 변환한다. |
| String | <code>toBinaryString(int i)</code> | int형의 정수를 2진수 형태의 문자열로 변환한다. |
| String | <code>toHexString(int i)</code> | int형의 정수를 16진수 형태의 문자열로 변환한다. |
| String | <code>toOctalString(int i)</code> | int형의 정수를 8진수 형태의 문자열로 변환한다. |

중간 점검 문제

1. 변수를 크게 두 가지로 나누면 **_기초자료형_** 변수와 **_참조자료형** 변수로 분류할 수 있다.
2. 객체를 생성하는 키워드는 **new** 이다.
3. 문자열(String)은 클래스 **_String_**의 객체이다.
4. 문자열의 길이를 반환하는 메소드는 이다.