

## ■ 생성자

- 인스턴스가 생성될 때마다 호출되는 '**인스턴스 초기화 메소드**'
- 인스턴스 변수의 초기화 또는 인스턴스 생성시 수행할 작업에 사용
- 몇가지 조건을 제외하고는 메소드와 같다.
- **모든 클래스에는 반드시 하나 이상의 생성자가 있어야 한다.**
- new 연산자에 의해 호출되어 객체의 초기화 담당

```
new 클래스();
```

```
Card c = new Card();
```

1. 연산자 new에 의해서 메모리(heap)에 Card클래스의 인스턴스가 생성된다.
2. 생성자 Card()가 호출되어 수행된다.
3. 연산자 new의 결과로, 생성된 Card인스턴스의 주소가 반환되어 참조변수 c에 저장된다.

## ■ 생성자의 기본 문법

- 생성자의 이름은 클래스의 이름과 같아야 한다.
- 생성자는 리턴값이 없다. (하지만 void를 쓰지 않는다.)

```
<modifiers> <class_name>(<argument_list>)  
{  
    [<statements>]  
}
```

## ■ 기본 생성자

- 매개변수가 없는 생성자
- 생성자를 선언하지 않은 경우 컴파일 시 자동 추가
- 클래스에 생성자가 하나도 없으면 컴파일러가 기본 생성자를 추가한다.  
(생성자가 하나라도 있으면 컴파일러는 기본 생성자를 추가하지 않는다.)

소스 파일(Car.java)

```
public class Car {  
  
}
```

→

바이트 코드 파일(Car.class)

```
public class Car {  
    public Car() { } //자동 추가  
}          기본 생성자
```

```
Car myCar = new Car();
```

기본 생성자

## ■ 기본 생성자 사용

ch08.Car1

```
String color;  
int door;  
  
public Car1() {  
    color = "빨강";  
    door = 2;  
}
```

ch08.Car1Main

```
Car1 car = new Car1();  
  
System.out.println(car.color);  
System.out.println(car.door);
```

빨강  
2

## ■ 매개변수가 있는 생성자

- 클래스에 생성자가 명시적으로 선언된 경우 반드시 선언된 생성자 사용

```
public class Car {  
    //생성자  
    Car(String model, String color, int maxSpeed) { ... }  
}
```

```
Car myCar = new Car("그랜저", "검정", 300);
```

## ■ 매개변수가 있는 생성자 사용

ch08.Car2

```
String color;  
int door;  
  
public Car2(String c, int d) {  
    color = c;  
    door = d;  
}
```

ch08.Car2Main

```
Car2 car = new Car2("빨강", 2);  
  
System.out.println(car.color);  
System.out.println(car.door);
```

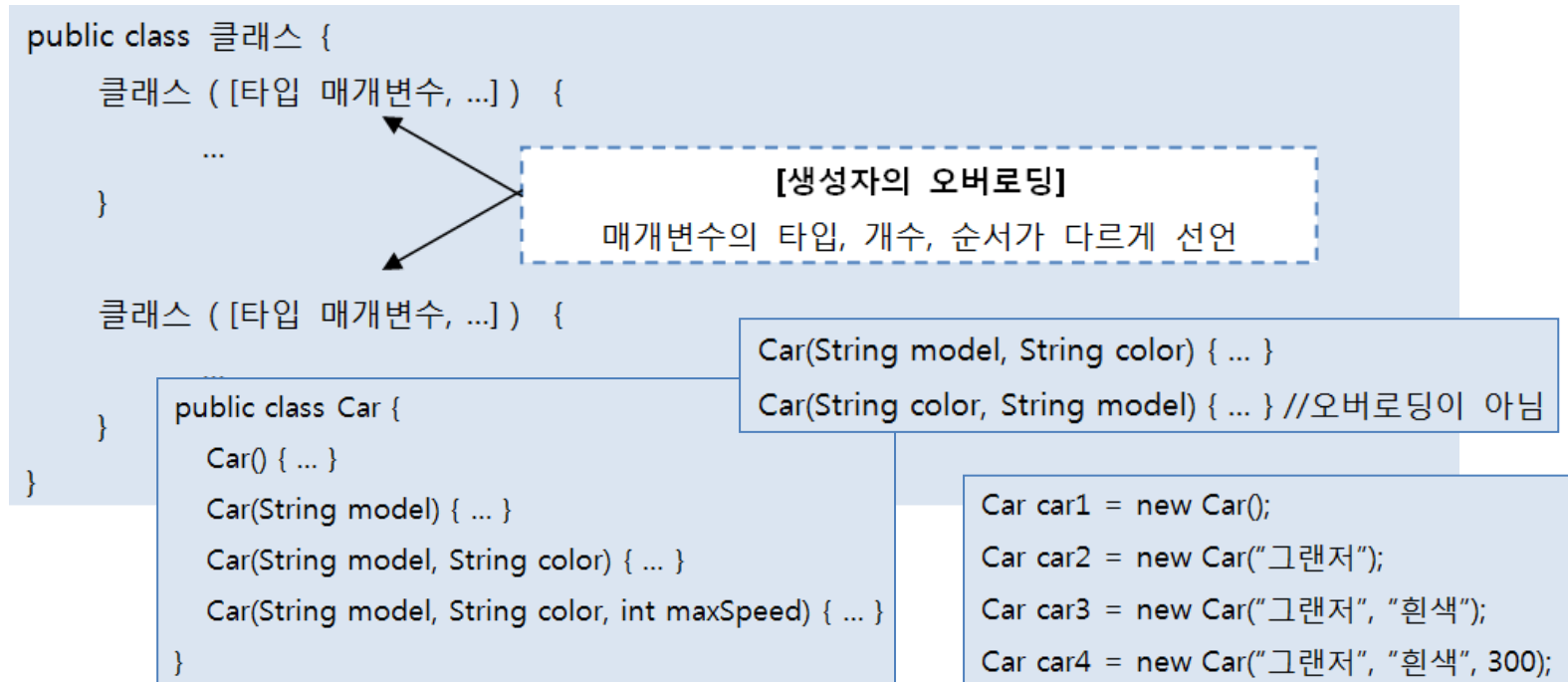
빨강  
2

## ■ 생성자를 여러가지 형태로 만드는 이유

- 객체 생성할 때 외부 값으로 객체를 초기화할 필요
- 외부 값이 어떤 타입으로 몇 개가 제공될 지 모름 - 생성자도 다양화

## ■ 생성자 오버로딩(Overloading)

- 매개변수의 타입, 개수, 순서가 다른 생성자 여러 개 선언



## ■ 여러가지 형태의 생성자 사용

ch08.Car3

```
String color;
int door;

public Car3() {
    color = "파랑";
    door = 4;
}

public Car3(String c) {
    color = c;
    door = 4;
}

public Car3(String c, int d) {
    color = c;
    door = d;
}
```



## ■ 연습문제 (ch08.연습문제01)

### ● 프로그램이 동작될 수 있도록 코드 작성하기

```
public class 연습문제01 {  
    float bottom;  
    float height;  
  
    // ① 기본 생성자 작성  
    // ② 매개변수 2개 생성자 작성  
    // ③ getArea 메소드 작성  
    // ④ bottom과 height의 setter 메소드 작성  
  
}
```

```
public class 연습문제01Main {  
    public static void main(String[] args) {  
        연습문제01 t1 = new 연습문제01(); // 기본 생성자  
        t1.setBottom(100.0f);  
        t1.setHeight(200.0f);  
        float tArea1 = t1.getArea();  
        System.out.println(tArea1);  
  
        연습문제01 t2 = new 연습문제01(20f, 50f); // 매개변수 2개 생성자  
        float tArea2 = t2.getArea();  
        System.out.println(tArea2);  
    }  
}
```

20000.0  
1000.0

## ■ 여러가지 형태의 생성자 사용

ch08.Car3Main

```
Car3 car = new Car3();  
System.out.println(car.color);  
System.out.println(car.door);  
  
Car3 car2 = new Car3("빨강");  
System.out.println(car2.color);  
System.out.println(car2.door);  
  
Car3 car3 = new Car3("노랑", 2);  
System.out.println(car3.color);  
System.out.println(car3.door);
```

파랑  
4  
빨강  
4  
노랑  
2

## ■ 생성자 필드 초기화

- 기본값으로 변수에 값을 할당하고 값을 바꿀 변수만 초기화 가능

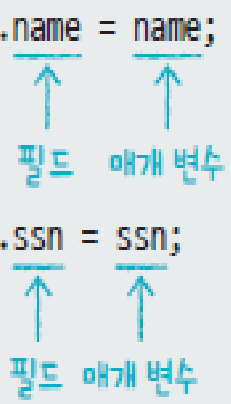
```
public class Korean {  
    //필드  
    String nation = "대한민국";  
    String name;  
    String ssn;  
  
    //생성자  
    public Korean(String n, String s) {  
        name = n;  
        ssn = s;  
    }  
}
```

```
Korean k1 = new Korean("박자바", "011225-1234567");  
Korean k2 = new Korean("김자바", "930525-0654321");
```

## ■ 생성자 필드 초기화

- 변수명은 상관없으나 가능한 한 필드의 이름과 동일하게 사용할 것을 권장
- 필드명과 매개변수의 이름이 같은 경우 this 키워드로 표현

```
public Korean(String name, String ssn) {  
    this.name = name;  
    this.ssn = ssn;  
}
```



## ■ 참조변수 this

ch08.Car4

```
String color = null;
int door = 0;

public Car4(String color, int door) {
    this.color = color;
    this.door = door;
}

public Car4(String color) { this(color, 4); }

public Car4() { }

public void setColor(String color) {
    this.color = color;
}

public String getColor() { return color; }

public void setDoor(int door) {
    this.door = door;
}

public int getDoor() { return door; }
```

## ■ 참조변수 this

ch08.Car4Main

```
Car4 car = new Car4();  
System.out.println(car.getColor());  
System.out.println(car.getDoor());  
  
Car4 car2 = new Car4("blue", 4);  
System.out.println(car2.getColor());  
System.out.println(car2.getDoor());
```

```
null  
0  
blue  
4
```

## ■ 생성자에서 다른 생성자 호출하기 – this()

- this() – 생성자, 같은 클래스의 다른 생성자를 호출할 때 사용
  - 초기화 내용을 한 생성자에 몰아 작성
  - 다른 생성자는 초기화 내용을 작성한 생성자를 this(...)로 호출
- 다른 생성자 호출은 생성자의 첫 문장에서만 가능

```
1 class Car {  
2     String color;  
3     String gearType;  
4     int door;  
5  
6     Car() {  
7         color = "white";  
8         gearType = "auto";  
9         door = 4;  
10    }  
11  
12    Car(String c, String g, int d) {  
13        color = c;  
14        gearType = g;  
15        door = d;  
16    }  
17  
18 }  
19
```

\* 코드의 재사용성을 높인 코드

```
Car() {  
    //Card("white","auto",4);  
    this("white","auto",4);  
}
```

## ■ 생성자에서 다른 생성자 호출하기

ch08.Car5

```
String color;  
int door;  
  
public Car5() {  
    this("파랑", 4); // 자신의 생성자 호출  
}  
  
public Car5(String c) {  
    this(c, 4);  
}  
  
public Car5(String c, int d) {  
    color = c;  
    door = d;  
}
```



## ■ 생성자에서 다른 생성자 호출하기

ch08.Car5Main

```
Car5 car = new Car5();  
System.out.println(car.color);  
System.out.println(car.door);  
  
Car5 car2 = new Car5("빨강");  
System.out.println(car2.color);  
System.out.println(car2.door);  
  
Car5 car3 = new Car5("노랑", 2);  
System.out.println(car3.color);  
System.out.println(car3.door);
```

파랑  
4  
빨강  
4  
노랑  
2