

Java의 정석

제 6 장

객체지향개념 I-3

- 1. 객체지향언어란?
- 2. 클래스와 객체

객체지향개념 1-1

- 3. 변수와 메서드
- 4. 메서드 오버로딩

객체지향개념 1-2

- 5. 생성자
- 6. 변수의 초기화

객체지향개념 1-3

5. 생성자

5.1 생성자란?

5.2 생성자의 조건

5.3 기본 생성자

5.4 매개변수가 있는 생성자

5.5 생성자에서 다른 생성자 호출하기 – this()

5.6 참조변수 this

5.7 생성자를 이용한 인스턴스의 복사

6. 변수의 초기화

6.1 변수의 초기화

6.2 멤버변수의 초기화

6.3 초기화 블록

6.4 멤버변수의 초기화 시기와 순서

5. 생성자

5.1 생성자(constructor)란?

▶ 생성자란?

- 인스턴스가 생성될 때마다 호출되는 ‘인스턴스 초기화 메서드’
- 인스턴스 변수의 초기화 또는 인스턴스 생성시 수행할 작업에 사용
- 몇가지 조건을 제외하고는 메서드와 같다.
- 모든 클래스에는 반드시 하나 이상의 생성자가 있어야 한다.

* 인스턴스 초기화 - 인스턴스 변수에 적절한 값을 저장하는 것.

```
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.

5.2 생성자의 조건

▶ 생성자의 조건

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

```
클래스이름 (타입 변수명, 타입 변수명, ... ) {  
    // 인스턴스 생성시 수행될 코드  
    // 주로 인스턴스 변수의 초기화 코드를 적는다.  
}
```

```
class Card {  
    ...  
    Card() { // 매개변수가 없는 생성자.  
        // 인스턴스 초기화 작업  
    }  
  
    Card(String kind, int number) { // 매개변수가 있는 생성자  
        // 인스턴스 초기화 작업  
    }  
}
```

5.3 기본 생성자(default constructor)

▶ 기본 생성자란?

- 매개변수가 없는 생성자
- 클래스에 생성자가 하나도 없으면 컴파일러가 기본 생성자를 추가한다.
(생성자가 하나라도 있으면 컴파일러는 기본 생성자를 추가하지 않는다.)

```
클래스이름 () { }
```

```
Card () { } // 컴파일러에 의해 추가된 Card클래스의 기본 생성자. 내용이 없다.
```

“모든 클래스에는 반드시

하나 이상의 생성자가 있어야 한다.”

5.3 기본 생성자(default constructor)

“모든 클래스에는 반드시 하나 이상의 생성자가 있어야 한다.”

[예제6-18]/ch6/ConstructorTest.java

```
class Data1 {  
    int value;  
}  
  
class Data2 {  
    int value;  
    Data2(int x) {    // 매개변수가 있는 생성자.  
        value = x;  
    }  
}  
  
class ConstructorTest {  
    public static void main(String[] args) {  
        Data1 d1 = new Data1();  
        Data2 d2 = new Data2();    // compile error발생  
    }  
}
```

```
class Data1 {  
    int value;  
    Data1() {} // 기본생성자  
}
```


5.4 매개변수가 있는 생성자

```
class Car {  
    String color;           // 색상  
    String gearType;        // 변속기 종류 - auto(자동), manual(수동)  
    int door;               // 문의 개수  
  
    Car() {} // 생성자  
    Car(String c, String g, int d) { // 생성자  
        color = c;  
        gearType = g;  
        door = d;  
    }  
}
```

```
Car c = new Car();  
c.color = "white";  
c.gearType = "auto";  
c.door = 4;
```



```
Car c = new Car("white", "auto", 4);
```

5.5 생성자에서 다른 생성자 호출하기 – this()

- ▶ this() – 생성자, 같은 클래스의 다른 생성자를 호출할 때 사용
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);  
}
```

```
Car() {  
    door = 5;  
    this("white","auto",4);  
}
```

5.6 참조변수 this

- ▶ this – 인스턴스 자신을 가리키는 참조변수. 인스턴스의 주소가 저장되어있음
모든 인스턴스 메서드에 지역변수로 숨겨진 채로 존재

```
1 class Car {  
2     String color;  
3     String gearType;  
4     int door;  
5  
6     Car() {  
7         //Card("white","auto",4);  
8         this("white","auto",4);  
9     }  
10
```

* 인스턴스변수와 지역변수를 구별하기
위해 참조변수 this사용

```
11 Car(String c, String g, int d){  
12     color = c;  
13     gearType = g;  
14     door = d;  
15 }  
16 }  
17
```

```
Car(String color, String gearType, int door){  
    this.color = color;  
    this.gearType = gearType;  
    this.door = door;  
}
```

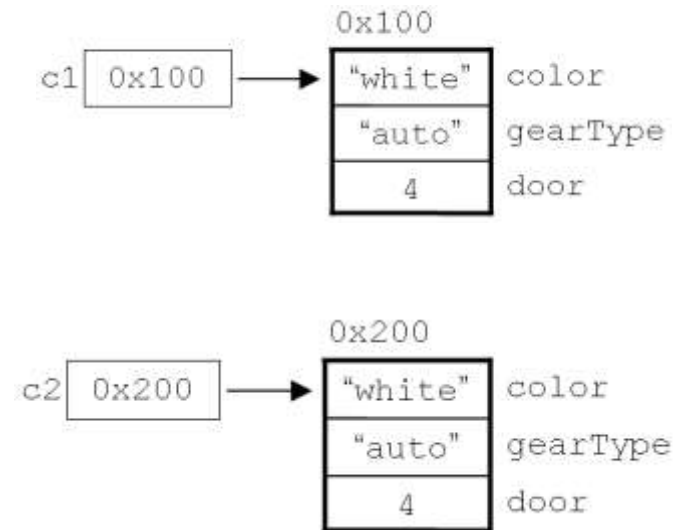
5.7 생성자를 이용한 인스턴스의 복사

- 인스턴스간의 차이는 인스턴스변수의 값 뿐 나머지는 동일하다.
- 생성자에서 참조변수를 매개변수로 받아서 인스턴스변수들의 값을 복사한다.
- 똑같은 속성값을 갖는 독립적인 인스턴스가 하나 더 만들어진다.

```

1 class Car {
2     String color;        // 색상
3     String gearType;     // 변속기 종류 - auto(자동), manual(수동)
4     int door;            // 문의 개수
5
6     Car() {
7         this("white", "auto", 4);
8     }
9
10    Car(String color, String gearType, int door) {
11        this.color = color;
12        this.gearType = gearType;
13        this.door = door;
14    }
15
16    Car(Car c) {           // 인스턴스의 복사를 위한 생성자.
17        color = c.color;
18        gearType = c.gearType;
19        door = c.door;
20    }
21 }
22
23 class CarTest3 {
24     public static void main(String[] args) {
25         Car c1 = new Car();
26         Car c2 = new Car(c1); // Car(Car c)를 호출
27     }
28 }

```



```

Car(Car c) {
    this(c.color, c.gearType, c.door);
}

```

6. 변수의 초기화

6.1 변수의 초기화

- 변수를 선언하고 처음으로 값을 저장하는 것
- 멤버변수(인스턴스변수, 클래스변수)와 배열은 각 타입의 기본값으로 자동초기화되므로 초기화를 생략할 수 있다.
- 지역변수는 사용전에 꼭!!! 초기화를 해주어야한다.

자료형	기본값
boolean	false
char	'\u0000'
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d 또는 0.0
참조형 변수	null

```
class InitTest {  
    int x;           // 인스턴스변수  
    int y = x;       // 인스턴스변수  
  
    void method1() {  
        int i;       // 지역변수  
        int j = i;    // 컴파일 에러!!! 지역변수를 초기화하지 않고 사용했음.  
    }  
}
```

6.1 변수의 초기화 - 예시(examples)

선언예	설 명
<pre>int i=10; int j=10;</pre>	int형 변수 i를 선언하고 10으로 초기화 한다. int형 변수 j를 선언하고 10으로 초기화 한다.
<pre>int i=10, j=10;</pre>	같은 타입의 변수는 콤마(,)를 사용해서 함께 선언하거나 초기화 할 수 있다.
<pre>int i=10, long j=0;</pre>	타입이 다른 변수는 함께 선언하거나 초기화할 수 없다.
<pre>int i=10; int j=i;</pre>	변수 i에 저장된 값으로 변수 j를 초기화 한다. 변수 j는 i의 값인 10으로 초기화 된다.
<pre>int j=i; int i=10;</pre>	변수 i가 선언되기 전에 i를 사용할 수 없다.

```
class Test
{
    int j = i;
    int i = 10; // 에러!!!
}
```

```
class Test
{
    int i = 10;
    int j = i; // OK
}
```

6.2 멤버변수의 초기화

▶ 멤버변수의 초기화 방법

1. 명시적 초기화(explicit initialization)

```
class Car {  
    int door = 4;           // 기본형 (primitive type) 변수의 초기화  
    Engine e = new Engine(); // 참조형 (reference type) 변수의 초기화  
  
    //...  
}
```

2. 생성자(constructor)

```
Car(String color, String gearType, int door){  
    this.color = color;  
    this.gearType = gearType;  
    this.door = door;  
}
```

3. 초기화 블록(initialization block)

- 인스턴스 초기화 블록 : { }
- 클래스 초기화 블록 : static { }

6.3 초기화 블록(initialization block)

- ▶ 클래스 초기화 블록 – 클래스변수의 복잡한 초기화에 사용되며 클래스가 로딩될 때 실행된다.
- ▶ 인스턴스 초기화 블록 – 생성자에서 공통적으로 수행되는 작업에 사용되며 인스턴스가 생성될 때 마다 (생성자보다 먼저) 실행된다.

```
class InitBlock {  
    static { /* 클래스 초기화블록 입니다. */ }  
  
    { /* 인스턴스 초기화블록 입니다. */ }  
  
    // ...  
}
```

```
1 class StaticBlockTest {  
2     static int[] arr = new int[10]; // 명시적 초기화  
3  
4     static { // 배열 arr을 1~10사이의 값으로 채운다.  
5         for(int i=0;i<arr.length;i++) {  
6             arr[i] = (int) (Math.random()*10) + 1;  
7         }  
8     }  
9     //...  
10 }
```

6.4 멤버변수의 초기화 시기와 순서

- ▶ 클래스변수 초기화 시점 : 클래스가 처음 로딩될 때 단 한번
- ▶ 인스턴스변수 초기화 시점 : 인스턴스가 생성될 때 마다

```

1 class InitTest {
2     static int cv = 1; // 명시적 초기화
3     int iv = 1;        // 명시적 초기화
4
5     static { cv = 2; } // 클래스 초기화 블록
6     { iv = 2; }        // 인스턴스 초기화 블록
7
8     InitTest() { // 생성자
9         iv = 3;
10    }
11 }

```

```
InitTest it = new InitTest();
```

클래스 초기화			인스턴스 초기화			
기본값	명시적 초기화	클래스 초기화블록	기본값	명시적 초기화	인스턴스 초기화블록	생성자
cv <input type="text" value="0"/>	cv <input type="text" value="1"/>	cv <input type="text" value="2"/>	cv <input type="text" value="2"/> iv <input type="text" value="0"/>	cv <input type="text" value="2"/> iv <input type="text" value="1"/>	cv <input type="text" value="2"/> iv <input type="text" value="2"/>	cv <input type="text" value="2"/> iv <input type="text" value="3"/>
1	2	3	4	5	6	7

6.4 멤버변수의 초기화 시기와 순서

- * 플래시 동영상 : Initialization.exe 또는 Initialization.swf
(java_jungsuk_src.zip의 flash폴더에 위치)

The screenshot shows the Java IDE with the following components:

- Console Window (MS-DOS):** Displays the command prompt with the following commands:

```
C:\Wj2sdk1.4.1\work>javac InitTest.java
C:\Wj2sdk1.4.1\work>java InitTest
```
- Code Editor (InitTest.java):** Shows the source code of the `InitTest` class:

```
1 class InitTest {
2     static int cv = 1;
3     int iv = 1;
4
5     static { cv = 2; }
6     { iv = 2; }
7
8     InitTest () {
9         iv = 3;
10    }
11
12    public static void main(String args[]) {
13        InitTest i = new InitTest();
14        System.out.println(cv);
15        System.out.println(i.iv);
16    }
17 }
```
- Method Area:** Shows the `InitTest` class object with a static variable `cv` set to 2.
- Call Stack:** Shows the `main` method in the `main` class, with a local variable `i` pointing to the `InitTest` instance.
- Heap:** Shows the `InitTest` instance object with a variable `iv` set to 3.

At the bottom, a status bar indicates: "10. cv의 값을 화면에 출력한다."

6.4 멤버변수의 초기화 시기와 순서 - 예제설명

```

1 class Product {
2     static int count = 0;    // 생성된 인스턴스의 수를 저장하기 위한 변수
3     int serialNo;           // 인스턴스 고유의 번호
4
5     { // 인스턴스 초기화 블록 : 모든 생성자에서 공통적으로 수행될 코드
6         ++count;
7         serialNo = count;
8     }
9
10    public Product() {}
11 }
12
13 class ProductTest {
14     public static void main(String args[]) {
15         Product p1 = new Product();
16         Product p2 = new Product();
17         Product p3 = new Product();
18
19         System.out.println("p1의 제품번호(serial no)는 " + p1.serialNo);
20         System.out.println("p2의 제품번호(serial no)는 " + p2.serialNo);
21         System.out.println("p3의 제품번호(serial no)는 " + p3.serialNo);
22         System.out.println("생산된 제품의 수는 모두 "+Product.count+"개 입니다.");
23     }
24 }

```

Diagram illustrating the state of the `Product` class and its instances during execution:

- Static Variable:** `count` is represented by a box containing the value 0.
- Instance 1 (p1):** Address `0x100`. Its `serialNo` field (represented by a box) contains the value `0x100`.
- Instance 2 (p2):** Address `0x200`. Its `serialNo` field (represented by a box) contains the value `0x200`.
- Instance 3 (p3):** Address `0x300`. Its `serialNo` field (represented by a box) contains the value `0x300`.