Java

제 9 강

객체지향개념 I-3 & 클래스 예제

1. 객체지향언어란?
2. 클래스와 객체

→ 객체지향개념 I-1

3. 변수와 메서드
4. 메서드 오버로딩

→ 객체지향개념 I-2

5. 생성자
6. 변수의 초기화

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

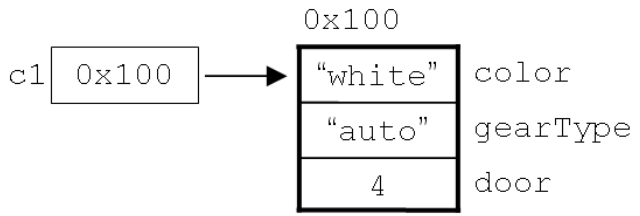
* 인스턴스변수와 지역변수를 구별하기 위해 참조변수 this사용

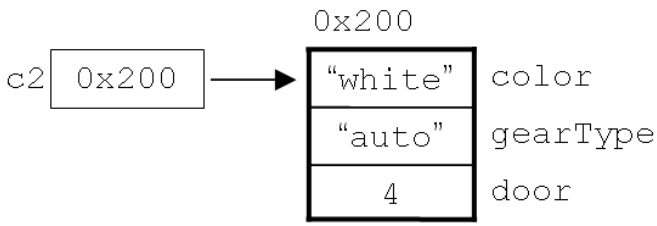
```
Car(String color, String gearType, int door){
    this.color = color;
    this.gearType = gearType;
    this.door = door;
}
```

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);
}
```

5.8 생성자 예제

```
1  package javacore.ch06;
2
3  public class TestVelocity {
4
5      public static void main(String[] args) {
6
7          VelocityUtils bike1 = new VelocityUtils(110); // speed (현재속도이며 초기값 0), maxSpeed ( 최고속도 )
8          VelocityUtils bike2 - new VelocityUtils(140);
9
10         bike1.speedUp(50); // 현재속도를 증가하는 함수( 단 0보다 작거나 최대속도보다는 커질 수 없음 )
11         System.out.println("bike1의 속도: " + bike1.showSpeed() + "km/h");
12         System.out.println("bike1의 최고속도: " + bike1.showMaxSpeed() + "km/h");
13
14         bike2.speedUp(70);
15         System.out.println("bike2의 속도: " + bike2.showSpeed() + "km/h");        // 현재속도를 보여줌
16         System.out.println("bike2의 최고속도: " + bike2.showMaxSpeed() + "km/h"); // 최고속도를 보여줌
17
18     }
19
20 }
```

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
int i=10; int j=10;	int형 변수 i를 선언하고 10으로 초기화 한다. int형 변수 j를 선언하고 10으로 초기화 한다.
int i=10, j=10;	같은 타입의 변수는 콤마(,)를 사용해서 함께 선언하거나 초기화 할 수 있다.
int i=10, long j=0;	타입이 다른 변수는 함께 선언하거나 초기화할 수 없다.
int i=10; int j=i;	변수 i에 저장된 값으로 변수 j를 초기화 한다. 변수 j는 i의 값인 10으로 초기화 된다.
int j=i; int i=10;	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 블럭(initialization block)

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

6.3 초기화 블럭(initialization block)

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

```
class InitBlock {
    static { /* 클래스 초기화블럭 입니다. */ }

    { /* 인스턴스 초기화블럭 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 블럭
6      {   iv = 2; }           // 인스턴스 초기화 블럭
7
8      InitTest() { // 생성자
9          iv = 3;
10     }
11 }
```

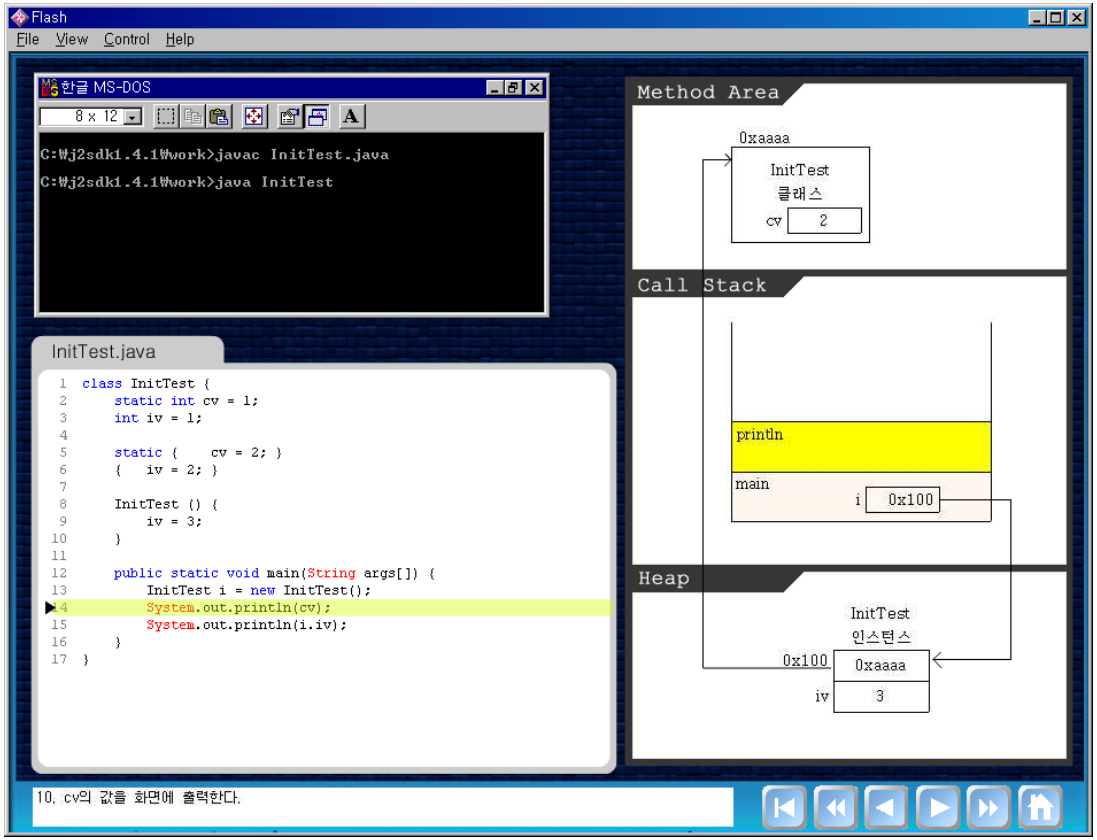
```
InitTest it = new InitTest();
```

클래스 초기화			인스턴스 초기화			
기본값	명시적 초기화	클래스 초기화블럭	기본값	명시적 초기화	인스턴스 초기화블럭	생성자
cv 0	cv 1	cv 2	cv 2 iv 0	cv 2 iv 1	cv 2 iv 2	cv 2 iv 3
1	2	3	4	5	6	7

6.4 멤버변수의 초기화 시기와 순서

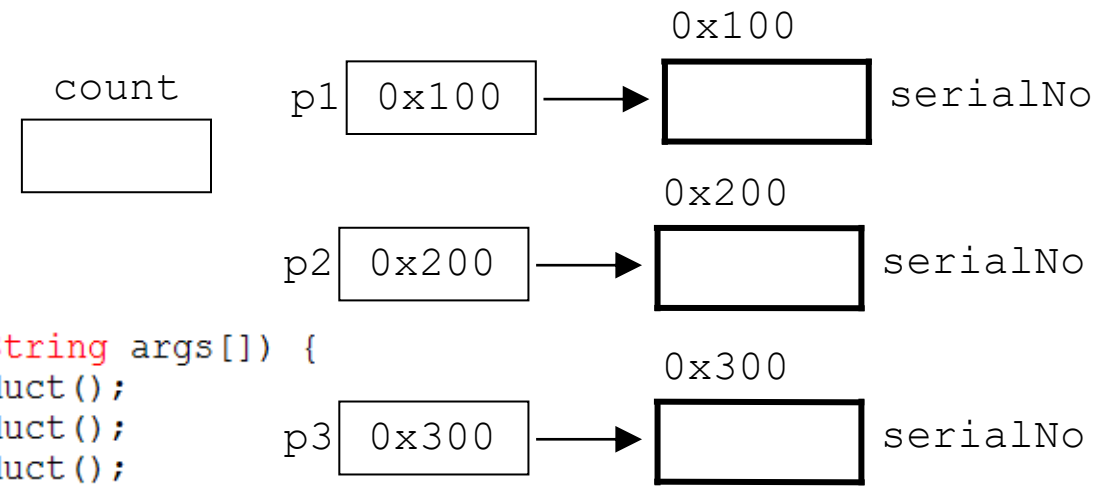
* 플래시 동영상 : Initialization.exe 또는 Initialization.swf

(java_jungsuk_src.zip의 flash폴더에 위치)



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 블럭 : 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 }
```



7. 클래스 예제

9강 객체지향개념 I & 클래스 예제

```
5  public class Machine2 {
6
7      public static void main(String[] args){
8
9          Product p = new Product();
10         Scanner in = new Scanner(System.in);
11
12         while(true){
13             System.out.print("아이스크림 자판기 시작할까요?(1: 예, 2:아니오) ");
14             int kind = in.nextInt();
15             if(kind == 2) {
16                 System.out.print("아이스크림 자판기를 종료합니다 Good Bye!");
17                 break;
18             }
19
20             System.out.println("Welcome to 아이스크림 자판기!!!");
21             System.out.print("아이스크림 자판기에 넣을 금액을 입력하세요: ");
22             p.enterMoney(in.nextInt());
23
24             System.out.print("선택(딸기:3000, 2:바닐라3500, 3:초코4000, 4:블루베리5000, 5:없음0): ");
25             p.buy(in.nextInt());
26             p.printMoney();
27         }
28     }
```

Product 클래스

```
1  package com.hk.app;
2
3  public class Product {
4      //자판기 제품 가격
5       final int RED = 3000;
6       final int YELLOW = 3500;
7       final int BROWN = 4000;
8       final int VIOLET = 5000;
9
10      //투입된 금액
11      int inMoney;
12
13      public void enterMoney(int money){
14       inMoney = money;
15      }
16
```

```
17      //구매 메서드
18      public void buy(int kind){
19
20       switch(kind){
21       case 1:
22        if(compute(RED)==true) {     System.out.println("1번 새콤달콤 딸기맛 아이스크림~");
23        }
24       break;
25       case 2:
26        if(compute(YELLOW)==true) {    System.out.println("2번 바닐라맛 아이스크림입니다!");
27        }
28       break;
29       case 3:
30        if(compute(BROWN)==true) {    System.out.println("3번 초코쿠키가 듬뿍 들어간 초코맛 아이스크림 야미~");
31        }
32       break;
33       case 4:
34        if(compute(VIOLET)==true) {    System.out.println("4번 몸에 좋은 블루베리 있는 아이스크림이 쏙~");
35        }
36       break;
37       default:
38        System.out.println("잘못 선택하셨습니다.");
39       }
40      }
```

```
42      //자판기 금액에서 차감 메서드
43      public boolean compute(int money){
44       if(inMoney < money)   {
45        System.out.println("금액이 부족합니다. 돈을 더 넣어주세요");
46        return false;
47       }
48       else {
49        inMoney -= money;
50        return true;
51       }
52      }
```

```
55      public void printMoney(){
56       int temp = inMoney;
57       int m1000 = 0;
58       int m500 = 0;
59       int m100 = 0;
60       int m50 = 0;
61       int m10 = 0;
62       m1000 = temp / 1000;
63       temp %= 1000;
64       m500 = temp / 500;
65       temp %= 500;
66       m100 = temp / 100;
67       temp %= 100;
68       m50 = temp / 50;
69       temp %= 50;
70       m10 = temp / 10;
71       temp %= 10;
72       System.out.println("거스름돈 총:"+inMoney+"원");
73       System.out.println("1000원: "+m1000);
74       System.out.println("500원: "+m500);
75       System.out.println("100원: "+m100);
76       System.out.println("50원: "+m50);
77       System.out.println("10원: "+m10);
78       System.out.println("입니다.");
79      }
```

감사합니다.