



제10장

생성자와 변수의 초기화

5. this() – 생성자에서 다른 생성자 호출

- **this()** – 생성자 호출, 같은 클래스 내에서 다른 생성자를 호출할 때 사용함.
다른 생성자 호출은 생성자의 첫 문장에서 사용해야 함.
this와 개념이 틀리다.

```
public class Student {  
    String name;  
    int age;  
  
    public Student() {  
        this.name = "이영준";  
        this.age = 55;  
    }  
  
    public Student(String name, int age) {  
        this.name = name;  
        this.age = age;  
    }  
}
```

```
public Student() {  
    this("이영준", 55);  
}
```

```
public Student() {  
    this.name = "손호준";  
    this("이영준", 55);  
}
```

또 다른, 생성자 호출함

this()는 앞서 수행한 값들을 되돌리기 때문에 항상 첫 문장에 쓰게 바람직하다. JDK1.8부터는 예외를 발생 시켜주고 있다.

6. this - 참조변수

- **this - 인스턴스 자신을 가리키는 참조변수.** 인스턴스의 주소가 저장되어있음
모든 인스턴스 메서드에 지역변수로 숨겨진 채로 존재함.
new라는 연산자가 힙에 인스턴스를 할당할 때, 비로소 활성화가 이루어짐.

(static에서는 사용할 수 없다. main()역시 사용불가)

```
public class Student {  
    String name;  
    int age;  
  
    public Student() {  
        this.name = "손호준";  
    }  
  
    public Student(String name,int age) {  
        this.name = name;  
        this.age = age;  
    }  
}
```

this() : 또 다른 생성자 호출

this : 참조변수와 같은 역할

1. 인스턴스 변수와 지역 변수를 구별하기 위해 참조변수 this 사용
2. this를 안 붙이면 지역변수로 인식함.(변수의 모호성이 발생)

7. 생성자를 이용한 인스턴스 복제

- 인스턴스간의 차이는 인스턴스 변수의 값 뿐 나머지는 동일하다.(붕어빵은 독립적)
- 생성자에서 참조변수를 매개변수로 받아서 인스턴스 변수들의 값을 복제.
- 같은 인스턴스 변수 값을 갖는 독립적인 인스턴스가 하나 더 만들어짐.

```
public class Student {  
    String name;  
    int age;  
  
    public Student() {  
        this("이영준", 55);  
    }  
  
    public Student(String name, int age) {  
        this.name = name;  
        this.age = age;  
    }  
    //인스턴스 복제를 위한 생성자.  
    public Student(Student s) {  
        this.name = s.name;  
        this.age = s.age;  
    }  
}
```

```
Student s1 = new Student();  
Student s2 = new Student(s1);
```

생성자 호출

```
public Student(Student s) {  
    this("이영준", 55);  
}
```

8. 변수의 초기화

▣ 변수를 선언하고 최초로 값을 저장하는 것

- 앞서 강의했지만, 멤버변수(인스턴스 변수, 정적 변수)는 new에 의해 각 타입의 기본값으로 자동 초기화가 되므로, 초기화를 생략 가능하다.
- 하지만, 지역변수(메서드 내) 사용 하기 전에 반드시 초기화를 해야 한다.

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

원래, 모든 변수는 무조건 초기화가 원칙이나, 자바에서는 클래스내의 멤버변수로 존재하는 변수들은 new연산자가 자동 초기화를 해주어 프로그래머의 불필요한 작업을 줄여준다.

9. 멤버변수의 초기화

▣ 멤버변수의 초기화 방법(3가지)

1. 명시적 초기화(explicit initialization)

```
class A {  
    int age = 55;  
    String name = "신은혁";  
}
```

2. 생성자(constructor)

```
public Student(String name, int age) {  
    this.name = name;  
    this.age = age;  
}
```

3. 초기화 블록(initialization block)

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

10. 초기화 블록

■ 클래스 초기화 블록

- 생성자 이전에 클래스 변수들의 복잡한 초기화에 사용된다.
- 클래스가 로딩될 때, **단 한번 실행**된다.

■ 인스턴스 초기화 블록

- 생성자에서 공통적으로 수행될 꺼라고 생각되는 것 중, 복잡한 실행코드를 작성한다.
- 인스턴스가 생성될 때마다, (**생성자보다 먼저**) 실행된다.

```
public class Student {  
  
    static double[] darr = new double[10];  
    int[] iarr = new int[10];  
    //정적 초기화 블록  
    static {  
        for(int i=0; i<darr.length; i++) {  
            darr[i] = Math.random() * 100;  
        }  
    }  
    //인스턴스 초기화 블록  
    {  
        for(int i=0; i<darr.length; i++) {  
            this.iarr[i] = (int)(Math.random() * 100);  
        }  
    }  
}
```

11. 멤버변수의 초기화 시기와 순서

- ▣ 클래스 변수 초기화 시점 : **클래스가 처음 로딩될 때, 단 한번**
- ▣ 인스턴스 변수 초기화 시점 : **인스턴스가 생성될 때 마다**

```
//명시적 초기화
static String name = "진창일";
int age = 28;

//초기화 블록
static { name = "김영삼"; }
{ this.age = 35; }

//생성자
public Student(String name, int age) {
    this.age = age;
}
```

초기화 순서

▣ 클래스 변수 초기화

1. 기본값
2. 명시적 초기화
3. 클래스 초기화 블록

▣ 인스턴스 변수 초기화

1. 기본값
2. 명시적 초기화
3. 인스턴스 초기화 블록
4. 생성자

12. 멤버변수의 초기화 시기와 순서-예제

```
public class Student {  
  
    String name;  
    int age;  
    static int i = 0;  
  
    //인스턴스 초기화 블록  
    {  
        ++i;  
        this.age = i;  
    }  
    //기본 생성자  
    public Student() {  
    }  
}
```

```
public class StudentExample {  
  
    public static void main(String[] args) {  
  
        Student s1 = new Student();  
        Student s2 = new Student();  
        Student s3 = new Student();  
  
        System.out.println(s1.age);  
        System.out.println(s2.age);  
        System.out.println(s3.age);  
        System.out.println(Student.i);  
    }  
}
```

Console 33
<terminated> StudentExamp
1
2
3
3

13. 싱글톤(singleton)

■ 하나의 어플리케이션에서 오르지 단, 하나만 생성되는 객체(인스턴스)

■ 싱글톤을 만드는 방법

```
public class User {  
    //외부에서 접근 못하도록 private불임.  
    private static User singleton = new User();  
  
    //기본 생성자  
    private User() {  
    }  
  
    //외부에서 호출할 수 있도록 getInstance()제공  
    public static User getInstance() {  
        return User.singleton;  
    }  
}
```

■ 싱글톤 순서

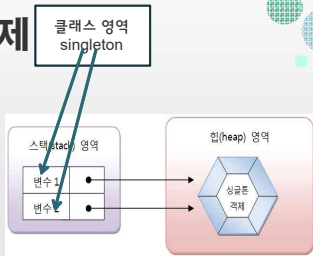
1. 해당 클래스 자신의 타입으로 정적 멤버 선언과 생성
2. 해당 클래스를 외부에서 new연산자로 인스턴스 생성 못하게 생성자를 private로 막기
3. 정적 멤버의 주소를 넘겨주기(공유) 위해서 외부에서 호출할 수 있는 getInstance()제공함

■ 싱글톤을 사용하는 이유

- 단, 하나의 인스턴만 생성을 원하고자 할 때 사용함.
ex) 사용자 환경설정, 커백션 풀, 사용자 정보로딩 등

14. 싱글톤(singleton) - 예제

```
public class UserEx {  
  
    public static void main(String[] args) {  
  
        User user1 = new User();  
        User user2 = new User();  
        User user3 = User.getInstance();  
        User user4 = User.getInstance();  
  
    }  
}
```



Console

<terminated> UserExample [Java Application] C:\Program Files\Java\jdk1.8.0_131\

sec07_exam_singleton.User@15db9742
sec07_exam_singleton.User@15db9742
같은 Singleton 객체입니다.

싱글톤은 하나의 객체의 주소를 공유하는 개념이기
때문에 위와 같이 주소가 동일하게 나옴을 명심하자.

감사합니다.

