

Chapter

06

클래스



06-3. 생성자

혼자 공부하는 자바(개정판) (신용권 저)

❖ 목차

- 시작하기 전에
- 기본 생성자
- 생성자 선언
- 필드 초기화
- 생성자 오버로딩
- 다른 생성자 호출 : this()
- 키워드로 끝내는 핵심 포인트



시작하기 전에

[핵심 키워드] : 기본 생성자, 생성자 선언, 매개 변수, 객체 초기화, 오버로딩, this()

[핵심 포인트]

생성자는 new 연산자로 호출되는 중괄호{} 블록이다.
객체 생성 시 초기화를 담당한다.

❖ 생성자 (constructor)

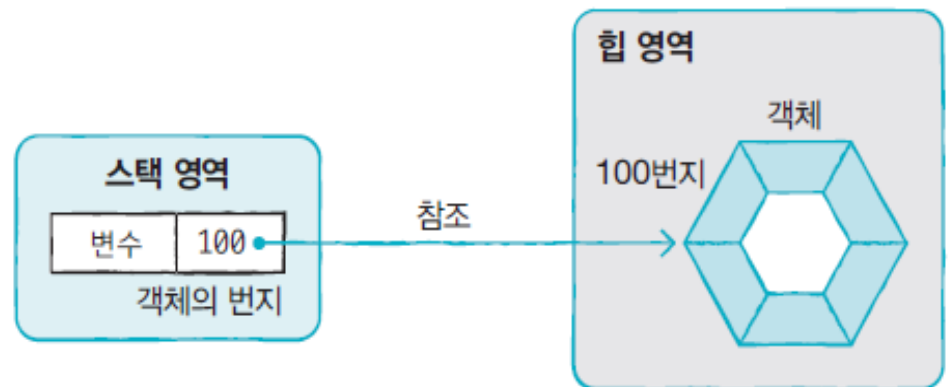
- 클래스로부터 new 연산자로 객체를 생성할 때 호출되어 객체의 초기화를 담당

❖ 객체 초기화

- 필드를 초기화하거나 메소드를 호출해,
객체를 사용할 준비를 하는 것

❖ 생성자가 성공적으로 실행

- 힙 영역에 객체 생성되고 객체 번지가 ;



기본 생성자

❖ 기본 생성자 (default constructor)

- 클래스 내부에 **생성자 선언 생략할 경우** 바이트 코드에 자동 추가

```
[public] 클래스() { }
```

- 클래스에 생성자 선언하지 않아도
new 생성자()로 객체 생성 가능

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

↑
기본 생성자

소스 파일(Car.java)

```
public class Car {  
  
}
```

컴파일

바이트 코드 파일(Car.class)

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

↑
기본 생성자



생성자 선언

❖ 생성자 선언

```
클래스( 매개변수선언, ... ) {  
    //객체의 초기화 코드  
}
```

} 생성자 블록

- 매개 변수 선언은 생략할 수도 있고 여러 개 선언할 수도 있음

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

- 클래스에 생성자가 명시적으로 선언되었을 경우 반드시 선언된 생성자 호출하여 객체 생성

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

필드 초기화

❖ 생성자의 필드 초기화

```
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");
```

필드 초기화

- 매개 변수 이름 은 필드 이름과 유사하거나 동일한 것 사용 권장
- 필드와 매개 변수 이름 완전히 동일할 경우 this.필드로 표현

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

가 , 가
name = name;
name
this

↑ ↑
필드 매개 변수
↑ ↑
필드 매개 변수



생성자 오버로딩

❖ 생성자 오버로딩 (overloading)

- 매개 변수를 달리하는 생성자 여러 개 선언 -> !
- 외부에서 제공되는 다양한 데이터를 사용하여 객체 화하기 위해

```
public class 클래스 {  
    클래스 ( [타입 매개변수, ...] ) {  
        ...  
    }  
  
    클래스 ( [타입 매개변수, ...] ) {  
        ...  
    }  
}
```

[생성자의 오버로딩]

매개 변수의 타입, 개수, 순서가 다르게 선언

가
ex. add(int a, long b) add(long b, int a)

ex. add(int a, int b) add(int b, int a) (가)



생성자 오버로딩

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

```
Car car1 = new Car();  
Car car2 = new Car("그랜저");  
Car car3 = new Car("그랜저", "흰색");  
Car car4 = new Car("그랜저", "흰색", 300);
```

- 매개 변수의 타입, 개수, 선언된 순서 같은 경우, 매개 변수 이름만 바꾸는 것은 생성자 오버로딩이 아니다

```
Car(String model, String color) { ... }  
Car(String color, String model) { ... } //오버로딩이 아님
```



다른 생성자 호출: this()

❖ this() 코드

- 생성자에서 다른 생성자 호출
- 필드 초기화 내용을 한 생성자에만 집중 작성하고 나머지 생성자는 초기화 내용 가진 생성자로 호출

· 생성자 수에 따라 초기화 내용도 코드 반복하게 되거

```
클래스( [매개변수, ...] ) {  
    this( 매개변수, ..., 값, ... ); ← 클래스의 다른 생성자 호출  
    실행문;  
}
```

- 생성자 첫 줄에서만 허용



다른 생성자 호출: this()

```
Car(String model) {  
    this.model = model;  
    this.color = "은색";  
    this.maxSpeed = 250;  
}
```

} 중복 코드

```
Car(String model, String color) {  
    this.model = model;  
    this.color = color;  
    this.maxSpeed = 250;  
}
```

} 중복 코드

```
Car(String model, String color, int maxSpeed) {  
    this.model = model;  
    this.color = color;  
    this.maxSpeed = maxSpeed;  
}
```

} 중복 코드

```
Car(String model) {  
    this(model, "은색", 250);  
}
```

```
Car(String model, String color) {  
    this(model, color, 250);  
}
```

```
Car(String model, String color, int maxSpeed) {  
    this.model = model;  
    this.color = color;  
    this.maxSpeed = maxSpeed;  
}
```

} 공통 실행코드



키워드로 끝내는 핵심 포인트

- **기본 생성자** : 클래스 선언 시 컴파일러에 의해 자동으로 추가되는 생성자
- **생성자 선언** : 생성자를 명시적으로 선언 가능. 생성자를 선언하면 기본 생성자는 생성되지 않음
- **매개 변수** : 생성자 호출 시 값을 전달받기 위해 선언되는 변수
- **객체 초기화** : 생성자 내부에서 필드값 초기화하거나 메소드 호출해서 사용 준비를 하는 것
- **오버로딩** : 매개 변수 달리하는 생성자를 여러 개 선언
- **this()** : 객체 자신의 또다른 생성자를 호출할 때 사용





Thank You!