03-07 [Java]

소유자	종수 김
태그	

[생성자가 필요한 이유]

생성자

- 객체를 생성하는 시점에 특정 작업을 진행

▼ 생성자가 없는 경우

```
package construct;

public class MemberInitMain1 {
    public static void main(String[] args) {
        MemberInit member1 = new MemberInit();
        member1.name = "user1";
        member1.age = 15;
        member1.grade = 90;

        MemberInit member2 = new MemberInit();
        member2.name = "user2";
        member2.age = 30;
        member2.grade = 80;

        MemberInit[] members = {member1, member2};
        for (MemberInit m : members) {
            System.out.println("이름 : " + m.name + " 나이 :
        }
    }
}
```

▼ 생성자가 없이 초기화 해주는 경우

```
package construct;

public class MemberInitMain2 {
    public static void main(String[] args) {
        MemberInit member1 = initMember("user1", 15, 90);

        MemberInit member2 = initMember("user2", 30, 80);

        MemberInit[] members = {member1, member2};
        for (MemberInit m : members) {
            System.out.println("이름 : " + m.name + " 나이 :
        }
    }
    public static MemberInit initMember(String name, int a
        MemberInit member = new MemberInit();
        member.name = name;
        member.age = age;
        member.grade = grade;

        return member;
    }
}
```

위의 코드를 보면 개선할 수 있는 사항은 속성과 기능을 한 곳에 두는 것.

▼ 생성자

```
package construct;

public class MemberInit {
    String name;
    int age;
    int grade;

    public MemberInit() {

    }
```

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


package construct;

public class MemberInitMain3 {
    public static void main(String[] args) {
        MemberInit member1 = new MemberInit("user1", 15, 9

        MemberInit member2 = new MemberInit("user2", 30, 8

        MemberInit[] members = {member1, member2};
        for (MemberInit m : members) {
            System.out.println("이름 : " + m.name + " 나이 :
        }
    }
}
```

'객체를 생성하는 시점'에 초기값을 생성하거나, 특정 작업을 실행하는데 사용

this

Q. 객체에서 멤버 변수와, 지역 변수의 네이밍이 같다면 둘을 어떻게 구분 ?

A. this 키워드를 통해 구분 가능. this.변수이름 이라면 멤버 변수에 접근, 일반적인 변수이름 이라면 지역 변수(범위상 가까운 스코프)를 의미.

Q. 만약 멤버 변수와 메서드 내에 지역 변수가 있다면 우선 순위는 ?

A. 지역 변수

- this는 생략 가능하나, 변수를 찾을 때 가까운(스코프) 변수를 우선해서 찾으며, 가까운 스코프가 없을 시 멤버 변수.

생성자 - 도입

객체를 생성하고 바로 초기 값을 할당해야 하는 경우.
객체를 생성하자마자 즉시 필요한 기능을 좀 더 편리하게 수행할 수 있도록 '생성자'라는 기

능을 제공

```
package construct;

public class MemberConstruct {
    String name;
    int age;
    int grade;

    MemberConstruct(String name, int age, int grade) {
        this.name = name;
        this.age = age;
        this.grade = grade;
    }
}
```

- 생성자는 메서드와 비슷하지만, 차이가 존재
  - 생성자의 이름은 클래스의 이름과 동일하다. 따라서 첫 글자도 대문자.
  - 생성자는 반환 타입이 없다.

생성자 호출
생성자는 인스턴스를 생성하고 나서 바로 호출 new 명령어 다음에 생성자 이름과 매개변수.

생성자 장점

- 좋은 프로그램은 자유도 높은 프로그램이 아닌 적절한 제약이 있는 프로그램.

1. 중복 호출 제거
   - 생성자가 없다면, 객체를 생성한 후 값을 세팅하기 위해 중복 호출이 필요
2. 제약 - 생성자 호출 필수
   - 생성자가 없는 경우 만약 값을 세팅하는 호출을 빠트리면 값이 할당되지 않음.
   이는 시스템에 문제가 발생할 수 있음.
3. 개발자가 의도한 대로 - 필수값 입력을 보장할 수 있다.
   - 개발자가 설계 당시 의도한대로 객체를 생성할 때 직접 정의한 생성자가 있다면 반드시 그 중 하나를 무조건 호출해야함. 이는 설계 당시 '초기 값이 없는 객체는 만들 수 없다.'와 같은 요구사항을 무조건 지키게 만들 수 있음.

생성자 - 기본 생성자

사용자가 직접 생성자를 정의하지 않으면, 컴파일러가 기본(default) 생성자를 자동으로 생성.
- 매개 변수가 존재하지 않음.

사용자 정의 생성자가 존재하면 컴파일러는 기본 생성자를 생성하지 않음.
기본 생성자를 개발자가 직접 정의해줘야함.

why?

생성자를 사용하지 않는 경우가 있는데, 이 때 기본 생성자를 자동으로 생성해주지 않으면 개발자가 직접 정의해줘야함. ⇒ 귀찮음 방지.

생성자 - 오버로딩

생성자도 메서드와 마찬가지로 매개변수(시그니처)만 다르게 오버로딩이 가능.
객체를 생성할 때, 무조건 직접 정의한 생성자 중 한개는 무조건 호출해야함.

```
    MemberConstruct(String name, int age, int grade) {
        this.name = name;
        this.age = age;
        this.grade = grade;
    }

    MemberConstruct(int age, int grade) {
        this("unnamed", age, grade);
    }
```

Tip : 오버로딩 된 생성자 내부에서 생성자를 호출해 중복을 제거할 수도 있다.

- this()는 생성자 코드의 첫줄에만 작성할 수 있다.