[Java] ??/????? this()? ??

??1

```
class Super {
    Super() {
        System.out.print('A');
    }
    Super(char x) {
        System.out.print(x);
    }
}

class Sub extends Super {
    Sub() {
        super();              // 호출되면 'A' 출력
        System.out.print('B');
    }
    Sub(char x) {
        this();               // Sub() 생성자 호출 → 'A' + 'B' 출력
        System.out.print(x); // 이후 'D' 출력
    }
}

class Test {
    public static void main(String[] args) {
        Super s1 = new Super('C'); // Super(char) 생성자 → 출력: C
        Super s2 = new Sub('D');   // Sub(char) → this() → super() → A + B + D
    }
}
// 최종출력 CABD
```

- `this()` → `Sub()` 생성자 호출
- `Sub()` 안에 `super()` 호출됨 → `Super()` 생성자 실행 → `'A'` 출력
- `Sub()`에서 `'B'` 출력

- `Sub()` 끝나면 다시 돌아와서 `'D'` 출력

자바에서 this()는 같은 클래스 내부의 다른 생성자를 호출할 때 사용된다.

정확히 말하면 생성자 안에서 this();를 쓰면 그 클래스(Sub)의 '매개변수가 없는 생성자'를 호출한다.

현재 실행 중이던 생성자는 잠시 멈추고, 호출된 생성자의 실행이 먼저 진행된다.

즉, 제어권(control)이 this()가 호출한 다른 생성자로 넘어가는 것이다.

```
Sub(char x) {
    this();            // 여기서 Sub() 생성자를 먼저 호출
    System.out.print(x);  // 그 다음에 'D' 출력
}
```

1. 이 때 this()는 같은 클래스 안에서만 쓸 수 있다.
2. 또한 this()는 항상 현재 클래스의 다른 생성자를 호출한다.
3. 매개변수가 맞아야 호출할 수 있다. (아래는 예시)

```
this();        // → Sub()
this('X');     // → Sub(char x)
this(1, 2);    // → Sub(int a, int b)
```

??2

```
class Parent {
    int x = 100;      // Parent 클래스의 인스턴스 변수 x

    Parent() {
        this(500);    // Parent(int x) 호출
    }

    Parent(int x) {
        this.x = x;   // Parent의 x에 값 저장
    }

    int getX() {
        return this.x;  // Parent의 x 반환
    }
```

```
}

class Child extends Parent {
    int x = 2000;     // Child 클래스의 인스턴스 변수 x

    Child() {
        this(5000);   // Child(int x) 호출
    }

    Child(int x) {
        this.x = x;   // Child의 x에 값 저장
    }

    public static void main(String[] args) {
        Child obj = new Child();
        System.out.println(obj.getX()); // Parent의 getX() 호출
    }
}
// 최종출력 500
```

1. `Child()` 호출
2. 내부에서 `this(5000)` → `Child(int x)` 호출
3. 자바가 자동으로 `super()` 삽입 → `Parent()` 호출
4. `Parent()` 안에서 `this(500)` → `Parent(int x)` 호출
5. `Parent`의 `x = 500` 설정
6. 이후 `Child(int x)` 실행 → `Child`의 `x = 5000` 설정
7. `obj.getX()` → `Parent`의 `x = 500` 반환

this(5000)은 같은 클래스인 Child의 생성자 중 , int를 매개로 받는 생성자를 호출한다 .

```
Child() {
    this(5000);   // 같은 클래스의 생성자 Child(int x) 호출
}
```

여기서 설정되는 건 Child 클래스의 x 필드 (int x = 2000) 이고 , this.x = x;에 의해 Child의 x는 5000으로 바 뀐다 .

그럼 **Parent**의 생성자는 언제 호출되는가**?**

항상 서브클래스 생성자가 호출되기 전에 슈퍼클래스 생성자가 먼저 호출되어야 한다 .

그래서 Child(int x)가 실행되기 전에 , 자동으로 super()가 먼저 붙는다 .

즉 , 자바가 내부적으로 다음처럼 처리한다 .

```
Child(int x) {
    super();        // 자동으로 삽입됨!
    this.x = x;     // Child의 x = 5000
}
```

그리고 Parent()는 다음처럼 생겼다 .

```
Parent() {
    this(500);    // Parent(int x) 호출
}
Parent(int x) {
    this.x = x;   // Parent의 x = 500
}
```

- Parent의 x = 500
- Child의 x = 5000

◻◻ obj.getX()는 왜 500이 나오는가?

getX()는 Parent 클래스에 정의되어 있다 .

```
int getX() {
    return this.x;
}
```

하지만 중요한 건 **getX()**는 **Parent**의 **x**를 반환한다는 뜻이 아니라, **"지금 참조 중인 인스턴스의 x 필드 중, Parent 클래스에서 접근 가능한 걸 반환한다"**는 의미이다.

→ 여기서 this는 Child 인스턴스를 가리키지만 , getX()는 Parent 클래스 안에 있어서 Parent의 x를 가리킨다 .

◻◻ super()는 부모 클래스의 '기본 생성자'를 호출한다.

`Child(int x)`는 이렇게 생겼다 .

```
Child(int x) {
    this.x = x;
```

```
}
```

그런데 이 생성자의 첫 줄에 명시적으로 super(...)가 없으면, 자바는 자동으로 super();를 삽입한다.
내부적으로는 이렇게 되는 것과 같다.

```
Child(int x) {
    super();      // 자동으로 삽입됨
    this.x = x;   // Child의 x 필드
}
```

→ 이 `super()`는 `Parent()`를 호출한다. 왜냐하면 `Parent` 클래스에 기본 생성자 `Parent()`가 존재하기 때문이다.

Parent()에서의 this(500)

```
Parent() {
    this(500);    // 같은 클래스의 다른 생성자 호출
}
```

이 this(500)은 마찬가지로 같은 Parent()클래스 내에 있는 매개변수가 있는 생성자를 호출한다.

```
Parent(int x) {
    this.x = x;
}
```

그래서 결국 아래와 같은 흐름으로 코드가 동작한다.

```
super();             // → Parent()
  → this(500);       // → Parent(int x)
     → this.x = 500; // → Parent의 x 필드에 500 저장
```

Revision #9
Created 20 June 2025 04:56:22 by Dain
Updated 22 June 2025 07:36:15 by Dain