

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

??1

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

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

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

// ????? CABD
```

- `this()` → `Sub()` ??? ??
- `Sub()` ?? `super()` ??? → `Super()` ??? ?? → `'A'` ??
- `Sub()` ?? `'B'` ??

- Sub() 메소드 호출 시 'D' 출력

이제 this()를 호출하여 Sub() 메소드를 호출하고, 'D'를 출력합니다.

이제 Sub() 메소드 내부에서 this()를 호출하여 Sub() 메소드를 호출하고, 'D'를 출력합니다.

이제 Sub() 메소드 내부에서 this()를 호출하여 Sub() 메소드를 호출하고, 'D'를 출력합니다.

이제 Sub() 메소드 내부에서 this()를 호출하여 Sub() 메소드를 호출하고, 'D'를 출력합니다.

```
Sub(char x) {
    this();          // Sub() 메소드 호출
    System.out.print(x); // 'D' 출력
}
```

1. 이 now this()를 호출하여 Sub() 메소드를 호출하고, 'D'를 출력합니다.
2. 이 now this()를 호출하여 Sub() 메소드를 호출하고, 'D'를 출력합니다.
3. 이 now this()를 호출하여 Sub() 메소드를 호출하고, 'D'를 출력합니다.

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

??2

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

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

    Parent(int x) {
        this.x = x; // Parent의 x를 x로 설정
    }

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

```

}

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

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

    Child(int x) {
        this.x = x;    // Child의 x를 5000으로 설정
    }

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

```

1. Child() 호출
2. this(5000) → Child(int x) 호출
3. super() → Parent() 호출
4. Parent() 호출 → this(500) → Parent(int x) 호출
5. Parent의 x = 500 설정
6. Child(int x) 호출 → Child의 x = 5000 설정
7. obj.getX() → Parent의 x = 500 반환

this(5000)은 Child 클래스의 x를 5000으로 설정하는 것, int x = 2000은 Child 클래스의 x를 2000으로 설정하는 것.

```

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

```

Child 클래스의 x는 2000 (int x = 2000) 이고, this.x = x;은 Child 클래스의 x를 5000으로 설정하는 것.

Parent 클래스의 getX() 메서드?

Parent 클래스의 getX() 메서드는 Child 클래스의 getX() 메서드를 호출하는 것.

Child(int x) 调用 , 调用 super() 调用 。

调用 , 调用 调用 调用 。

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

调用 Parent() 调用 调用 。

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

- Parent x = 500
- Child x = 5000

调用 obj.getX() 调用 500 调用?

getX() 调用 Parent 调用 调用 调用 。

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

调用 调用 调用 **getX()** 调用 **Parent** 调用 x 调用 调用 调用 调用, "调用 调用 调用 调用 x 调用 , **Parent** 调用 调用 调用 调用 调用" 调用 调用。

→ 调用 this 调用 Child 调用 调用 调用 , getX() 调用 Parent 调用 调用 调用 Parent 调用 x 调用 调用 调用 。

调用 **super()** 调用 调用 调用 '调用 调用' 调用 调用。

Child(int x) 调用 调用 调用 。

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

```
}
```

이것이 바로 `super(...)`이 호출되는 방법입니다. `super();`는 `super()`를 호출하는 것과 동일합니다.

```
Child(int x) {  
    super();    // 부모 클래스 호출  
    this.x = x; // Child의 x 값  
}
```

→ `super()`는 `Parent()`를 호출합니다. `Parent`는 `Parent` 클래스를 호출하는 것입니다.

`Parent()`는 `this(500)`

```
Parent() {  
    this(500); // 500 값을 this에 전달  
}
```

`this(500)`는 `Parent()`를 호출하는 것과 동일합니다.

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

이것이 바로 `this`가 사용되는 방법입니다.

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

Revision #9

Created 20 June 2025 04:56:22 by Dain

Updated 22 June 2025 07:36:15 by Dain