



JAVA

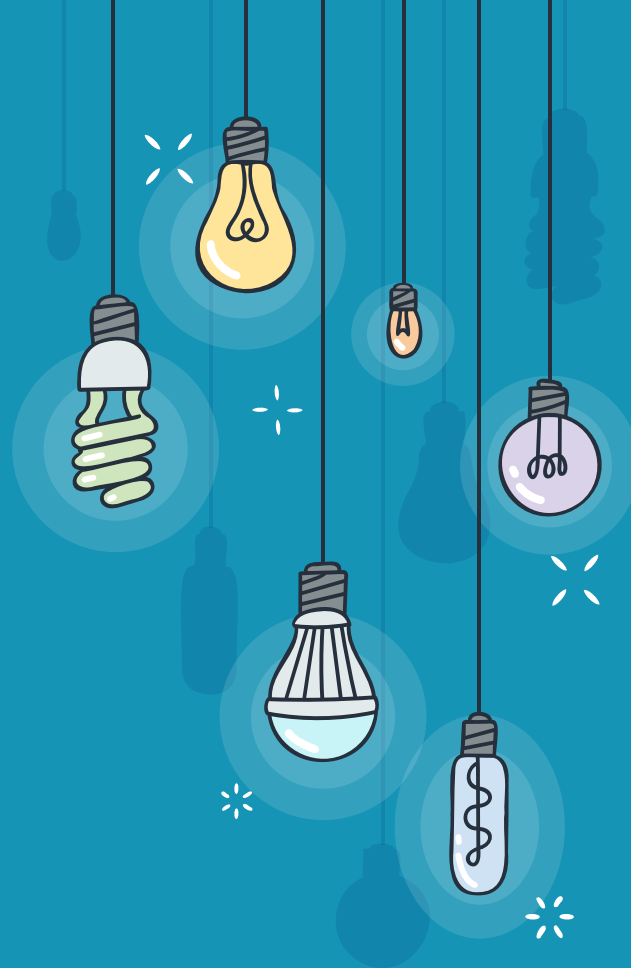
1. 객체지향언어

2. 클래스와 객체

3. 변수

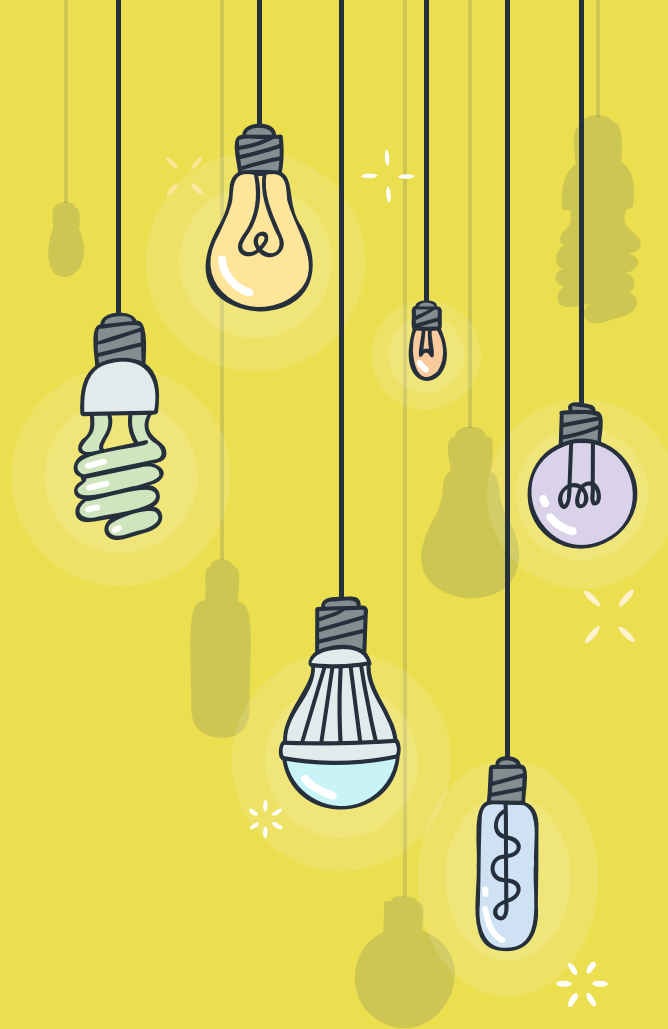
4. 메서드

5. 메서드 오버로딩



1

상속



* 상속(INHERITANCE)

+ 상속

× 부모 클래스의 모든 멤버를 자식 클래스에게 물려줌

◆ 생성자 제외

× 장점

◆ 기존의 클래스를 재사용

◆ 중복 코드 줄임

◆ 부모 클래스의 수정으로 모든 자식 클래스도 수정되는 효과로
유지보수 시간 줄임

* 상속

+ 클래스 상속

- × 자식 클래스 선언 시 extends 뒤에 부모 클래스 기술

```
class Child extends Parent {  
    ...  
}
```

- × 단 하나의 부모 클래스만 상속 가능
- × 자식 객체 생성시 부모 객체가 생성된 후 자식 객체가 생성됨

* OBJECT 클래스

+ Object 클래스

- × 상속이 없는 클래스는 자동으로 Object클래스를 상속 받음
- × 모든 클래스는 Object클래스에 정의된 11개의 메소드를 자동 상속 받음
 - ◆ toString(), equals(Object obj), hashCode(), ...

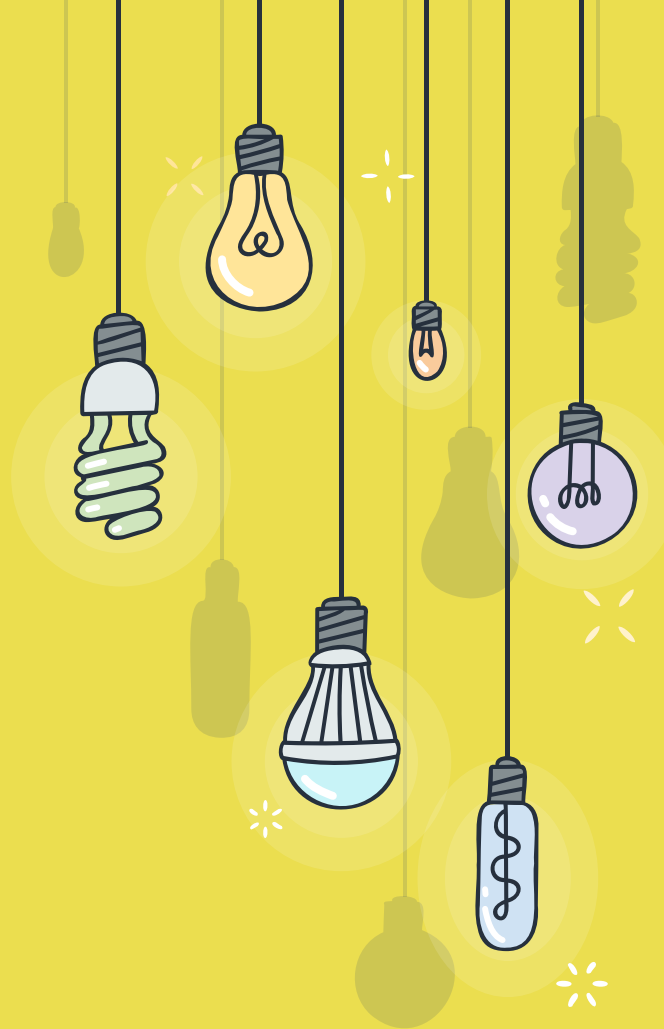
```
class Parent {  
    ...  
}  
  
class Child extends Parents {  
    ...  
}
```

=

```
class Parent extends Object {  
    ...  
}  
  
class Child extends Parents {  
    ...  
}
```

2

오버라이딩



* 오버라이딩(OVERRIDING)

+ 오버라이딩

- × 자식 클래스 에서 부모의 메소드를 재정의(수정)하여 사용
- × 자식 객체에서 메소드를 호출하면 재정의된 메서드가 호출됨

+ 오버라이딩 조건

- × 선언부가 같아야 함
- × 접근제어자는 최소 부모와 같거나 더 넓어야 함
- × 새로운 예외를 throws 할 수 없음

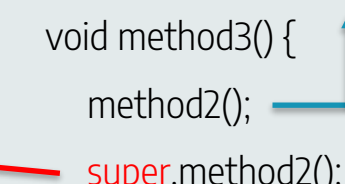
* SUPER 키워드

+ super 키워드

- × 오버라이딩한 메소드는 항상 자식 메소드가 호출됨
- × 부모의 메소드를 호출하려면 super 키워드를 붙여 사용

```
class Parent {  
    void method1() { ... }  
    void method2() { ... }  
}
```

```
class Child extends Parent {  
    void method2() { ... }  
    void method3() {  
        method2();  
        super.method2();  
    }  
}
```



* THIS 와 SUPER

+ this

× 인스턴스 자신을 가리키는 참조변수

+ super

× 부모의 멤버를 가리키는 참조변수

```
class Parent {  
    int var = 1;  
}  
class Child extends Parent {  
    int var = 10;  
    void method() {  
        int var = 100;  
        System.out.println(x);  
        System.out.println(this.x);  
        System.out.println(super.x);  
    }  
}
```

* SUPER() - 조상의 생성자 호출

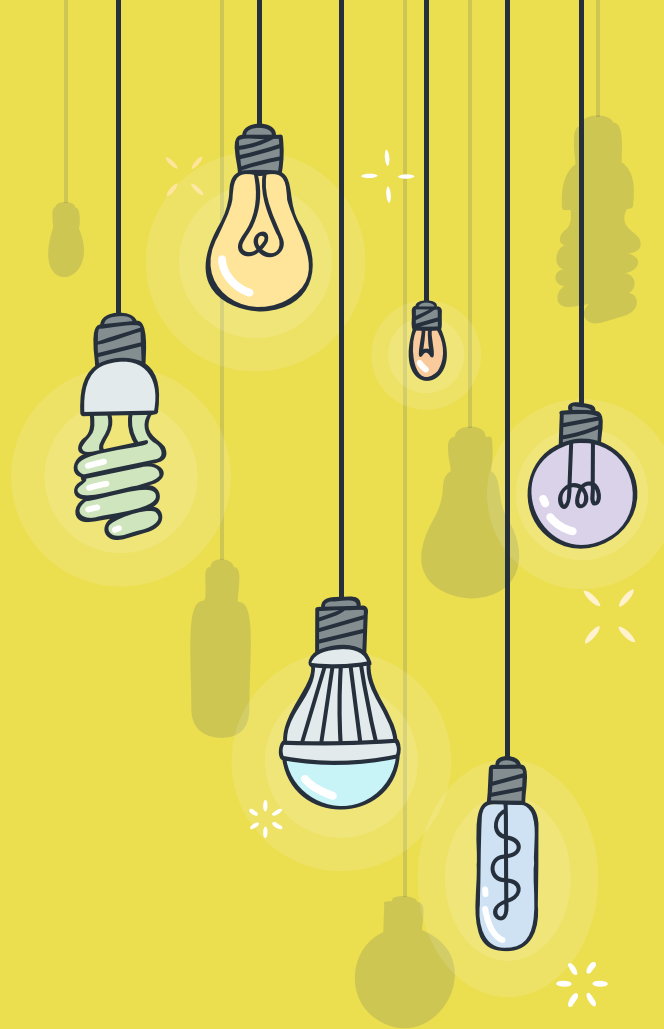
+ super()

- × 부모 클래스에 생성자가 정의되어 있고 매개변수를 받을 때
 - ◆ 자식 클래스를 객체 생성하면 우선 부모 클래스의 객체가 만들어 지면서 이때 부모의 생성자가 호출됨
- × 자식 생성자에 super()로 부모 생성자를 호출하여 인자값을 넘겨줌
- × super()는 자식 생성자 반드시 첫 줄에 기술

```
class Parent {  
    int var;  
    public Parent(int var) {  
        this.var = var;  
    }  
}  
class Child extends Parent {  
    public Child() {  
        super(10);  
    }  
}
```

3

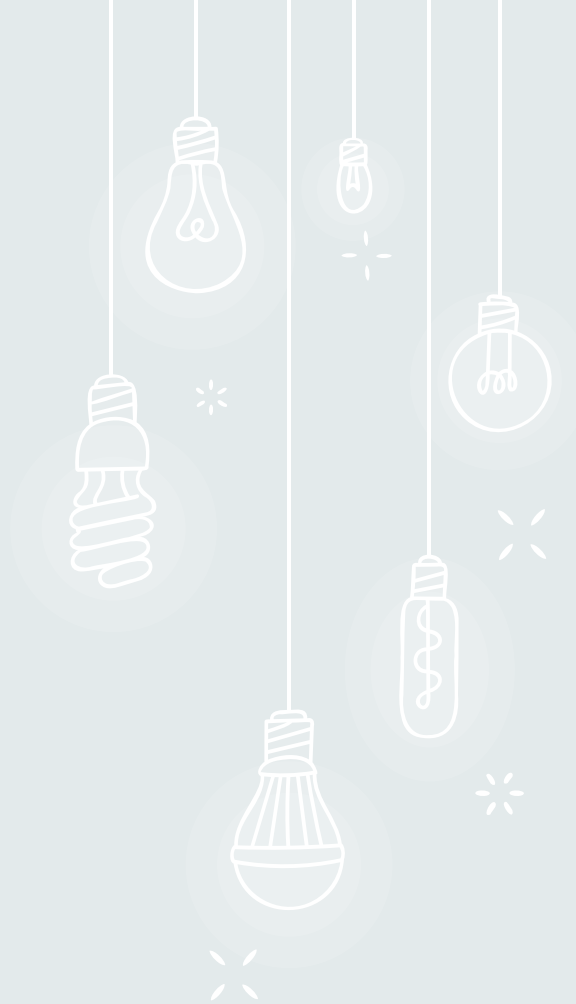
다형성



* 다형성

+ 다형성

- × 하나의 객체가 다양한 형태 또는 타입을 가질 수 있는 능력
- × 클래스 타입변환과 메소드 재정의를 이용하여 다형성 구현



THANKS!

+ Any questions?

