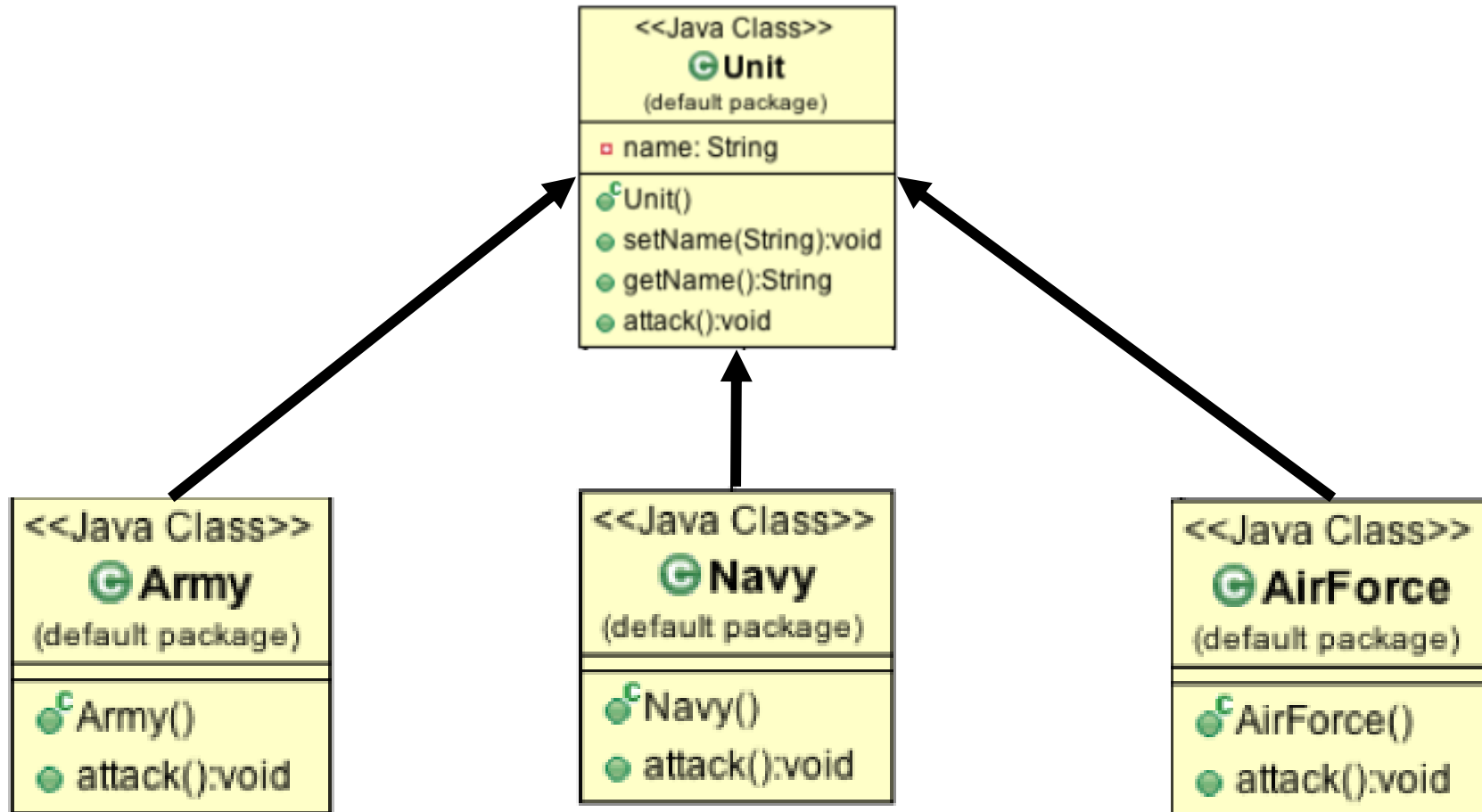


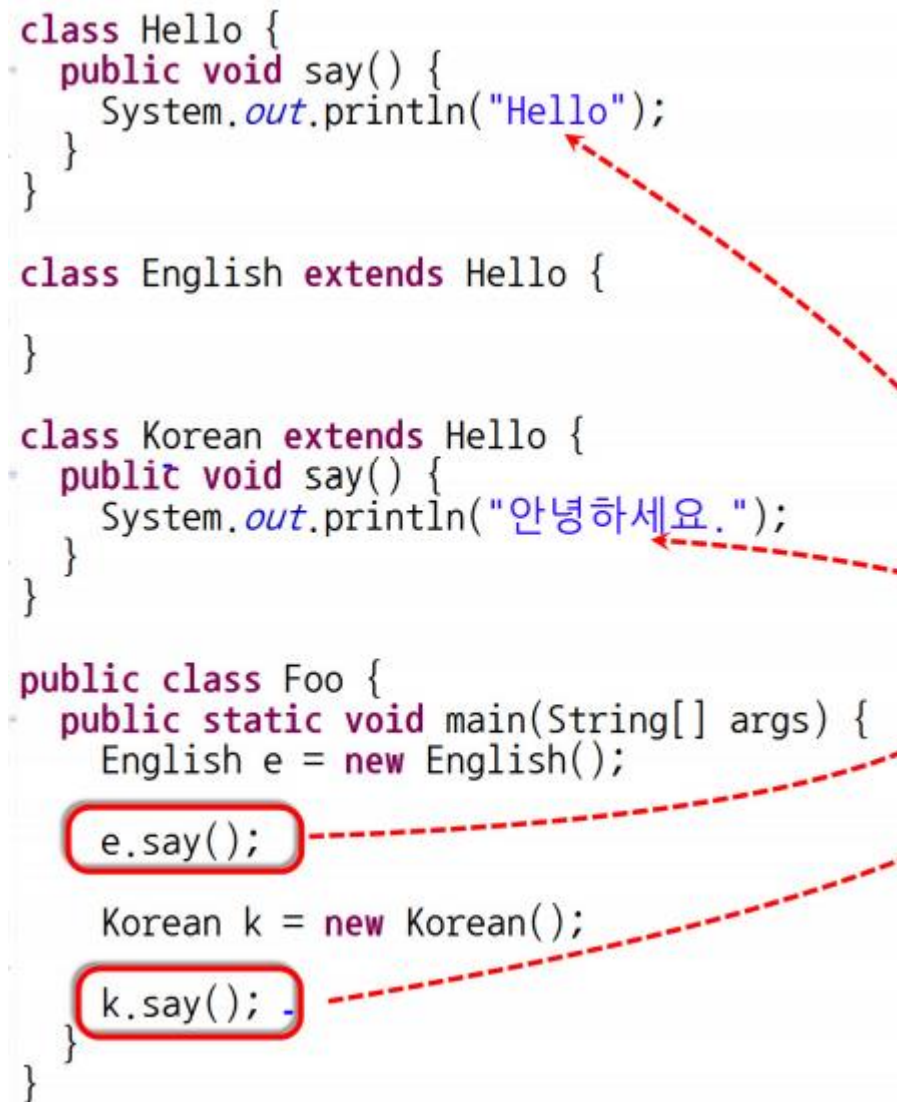
# Override

# 1. 다형성 구현



## 2. Override

```
class Hello {  
    public void say() {  
        System.out.println("Hello");  
    }  
}  
  
class English extends Hello {  
}  
  
class Korean extends Hello {  
    public void say() {  
        System.out.println("안녕하세요.");  
    }  
}  
  
public class Foo {  
    public static void main(String[] args) {  
        English e = new English();  
        e.say();  
        Korean k = new Korean();  
        k.say();  
    }  
}
```



### 3. super(1/4)

```
1 class Hello {  
2     public void say() {  
3         System.out.println("Hello");  
4     }  
5 }  
6  
7 class Korean extends Hello {  
8     public void say() {  
9         System.out.println("안녕하세요.");  
10    }  
11  
12    public void sayHello() {  
13        super.say();  
14        this.say();  
15    }  
16 }
```

The diagram illustrates the execution flow of the `sayHello()` method in the `Korean` class. A red dotted arrow originates from the `super.say();` call on line 13 and points to the `say()` method of the `Hello` class on line 2. A blue dotted arrow originates from the `this.say();` call on line 14 and points to the `say()` method of the `Korean` class on line 8.

### 3. super(2/4)

```
1 class Hello {  
2     public void say() {  
3         System.out.println("Hello");  
4     }  
5 }  
6  
7 class Korean extends Hello {  
8     public void say() {  
9         System.out.println("안녕하세요.");  
10    }  
11 }  
12  
13 public class Foo {  
14     public static void main(String[] args) {  
15         Korean k = new Korean();  
16         k.say();  
17     }  
18 }
```

The diagram illustrates the execution of the `k.say();` statement in the `main` method of the `Foo` class. Two arrows originate from this call:

- A blue dashed arrow points to the `say()` method in the `Hello` class, representing the superclass method.
- A red dotted arrow points to the `say()` method in the `Korean` class, representing the subclass method that overrides the superclass method.

### 3. super(3/4)

```
1 class Hello {  
2     public void say() {  
3         System.out.println("Hello");  
4     }  
5 }  
6  
7 class Korean extends Hello {  
8     public void say() {  
9         System.out.println("Hello");  
0         System.out.println("안녕하세요.");  
1     }  
2 }  
3  
4 public class Foo {  
5     public static void main(String[] args) {  
6         Korean k = new Korean();  
7         k.say();  
8     }  
9 }
```

The diagram illustrates the relationship between the `say()` methods in the `Hello` and `Korean` classes. A red dotted arrow points from the `say()` method in the `Korean` class to the `say()` method in the `Hello` class, indicating that `Korean` inherits the `say()` method from `Hello`. A blue dashed arrow points from the right towards the `say()` method in the `Hello` class.

### 3. super(4/4)

```
1 class Hello {  
2     public void say() {  
3         System.out.println("Hello");  
4     }  
5 }  
6  
7 class Korean extends Hello {  
8     public void say() {  
9         super.say();  
10        System.out.println("안녕하세요.");  
11    }  
12 }  
13  
14 public class Foo {  
15     public static void main(String[] args) {  
16         Korean k = new Korean();  
17         k.say();  
18     }  
19 }
```

The diagram illustrates the inheritance of the `say()` method. A red box highlights the `say()` method in the `Hello` class (lines 2-4). A red dotted arrow points from the `super.say();` call in the `Korean` class (line 9) to the `say()` method in the `Hello` class. A blue dashed arrow points from the `super.say();` call to the `say()` method in the `Korean` class, indicating the method call within the subclass.

## 4. 생성자가 있는 부모 클래스

```
1 class Hello {  
2     public Hello(String msg) {  
3         System.out.println(msg);  
4     }  
5 }  
6  
7 class Korean extends Hello {  
8  
9 }
```

```
1 class Hello {  
2     public Hello(String msg) {  
3         System.out.println(msg);  
4     }  
5 }  
6  
7 class Korean extends Hello {  
8     // 부모와 동일한 파라미터를  
9     // 받도록 생성자를 정의하고,  
10    // 전달받은 파라미터를 부모에게  
11    // 재전달 한다.  
12    public Korean(String msg) {  
13        super(msg);  
14    }  
15 }
```