

Java Study Skill Up Team Review


7.5 final class, final method

- final field 는 기본적으로 초기값 설정 이후에 값을 변경할 수 없다고 배웠다.

☹️ 그럼 final class, final method 는 어떻게 제어가 될까? 이는 상속과 관련이 있었다.

[final class]

- 먼저 class 에 final 키워드를 선언하면 말 그대로 최종적인 클래스로 상속이 불가능한 클래스가 된다.
- 그 말은 final 키워드로 선언된 클래스는 부모 클래스가 될 수 없다는 말이다.

 FinalParent.java

```
package com.chapter.ch07.example.ex03;

final class FinalParent {
}
```

 Children.java

```
package com.chapter.ch07.example.ex03;

public class Children extends FinalParent {
}
```

☹️ 오류 발생

- Cannot inherit from final 'com.chapter.ch07.example.ex03.FinalParent'

☹️ 오류 내용


- 부모 클래스가 final 로 선언되어 있어서 해당 클래스를 상속할 수 없다.

[final method]

☹️ 그럼 method 가 final 이면 호출이 불가능할까?


✗ 그건 아니다.

- method 에 final 키워드를 선언하면 최종적인 메서드가 되는 것은 맞지만,
- 호출은 가능하되 오버라이딩을 할 수 없는 메서드가 된다.

 ParentFinalMethod.java

```
package com.chapter.ch07.example.ex03;

public class ParentFinalMethod {
    final void finalMethod() {
        System.out.println("final method");
    }
}
```

 ParentFinalMethod.java

```
package com.chapter.ch07.example.ex03;

public class Children extends ParentFinalMethod {

}
```

 Main.java

```
package com.chapter.ch07.example.ex03;

public class Main {
    public static void main(String[] args) {
        Children children = new Children();
        children.finalMethod();
    }
}
```


- 즉 이처럼 상속 받고 ParentFinalMethod 의 finalMethod 를 호출하면 "final method" 가 잘 출력이 된다.
- 하지만

 Children.java

```
package com.chapter.ch07.example.ex03;

public class Children extends ParentFinalMethod {
    @Override
    void finalMethod() {    // ✕ 오류 발생
        System.out.println("overriding");
    }
}
```

- 이처럼 오버라이딩을 시도하면 오류가 발생한다.

 오류 발생

- 'finalMethod()' cannot override 'finalMethod()' in 'com.chapter.ch07.example.ex03.ParentFinalMethod'; overridden method is final

오류 내용

- final 로 선언되었기 때문에 자식 클래스에서 변경하거나 재정의(override)할 수 없다.

[Protected 접근 제한자]

- protected 는 상속과 관련이 있다.
- 같은 패키지의 경우 default 처럼 접근이 가능하다.
- 다른 패키지의 경우 자식 클래스만 접근이 가능하다.

ProtectedClass.java

```
package com.chapter.ch07.example.ex04;

public class ProtectedClass {
    void protectedMethod() {
        System.out.println("protected method");
    }
}
```

Children.java (같은 패키지)

```
package com.chapter.ch07.example.ex04;

public class Children {
    // 같은 패키지 ○ 접근 가능
    void childrenMethod() {
        ProtectedClass protectedClass = new ProtectedClass();
        protectedClass.protectedMethod();
    }
}
```

- 같은 패키지는 protected 키워드로 선언한 method 에 접근이 가능하다.
- 하지만

Children.java (다른 패키지)

```
package com.chapter.ch07.example.ex03;

import com.chapter.ch07.example.ex04.ProtectedClass;

public class Children {
    // 다른 패키지 ✕ 접근 불가능
    void method () {
        ProtectedClass protectedClass = new ProtectedClass();
    }
}
```

```
        protectedClass.protectedMethod();
    }
}
```

- 다른 패키지의 경우 오류가 발생한다.

오류 발생

- 'protectedMethod()' is not public in 'com.chapter.ch07.example.ex04.ProtectedClass'. Cannot be accessed from outside package

오류 내용

- 'protected' 접근 제어자는 같은 패키지 내에서는 접근할 수 있으나, 패키지 외부에서는 접근할 수 없도록 제한된 접근 제어자이다.

해결 방안

- 생성자에서 super() 로 호출하여 상속을 통해 사용이 가능하다.

Children.java (다른 패키지)

```
package com.chapter.ch07.example.ex03;

import com.chapter.ch07.example.ex04.ProtectedClass;

public class Children extends ProtectedClass{
    // 다른 패키지
    public Children() {
        super();    // 자식 생성자에서 super() 호출
        this.protectedMethod(); // ⬤ 접근 가능
        method();
    }

    public void method () {
        this.protectedMethod(); // ⬤ 접근 가능
    }
}
```

- 하지만 X x = new X(); 처럼 직접 객체를 생성해서 사용하는 것은 불가능 하다.

7.7 타입 변환

[자동 타입 변환]

- 자동적으로 타입 변환이 일어나는 것

main.java

```

package com.chapter.ch07.example.ex05;

public class main {
    public static void main(String[] args) {
        Cat cat = new Cat();
        Dog dog = new Dog();

        Animals animals1 = cat;
        Animals animals2 = dog;

        System.out.println(animals1 == cat);
        System.out.println(animals2 == dog);

        Animals animals3 = new Cat();    // ○ Dog 도 가능
        Cat cat1 = new Animals();         // ✕ 컴파일 에러: 상위를 하위에 대입하는 것을
불가능함
        Cat cat2 = dog;                   // ✕ 컴파일 에러: 상속 관계가 아님

        animals1.catMethod();    // Animals 부모 클래스에 있는 메서드
        animals1.catMethod2();   // Animals 부모 클래스에 있는 오버라이딩 된 메서드
        animals1.catMethod3();   // ✕ Animals 부모 클래스에 없음. 컴파일 에러 : 부모 타
입으로 자동 타입 변환된 이후에는 부모 클래스에 선언된 필드와 메서드만 접근이 가능하다.
    }
}

```

- 즉 Animals 타입과 Cat, Dog 는 각각 다른 타입으로 분류되기 때문에 대입이 불가능하다.
- 하지만 상속 관계에 있을 때 자식은 부모를 상속 받아 동일하게 취급될 수 있다.
- 그렇기 때문에 변수 타입은 달라도 상속 관계에서 상위 타입 객체에 하위 타입 객체가 대입될 수 있다. (반대 불가능)
- 이때 자동 타입 변환이 일어나며 동일한 객체를 참조하는 것이 가능해 진다.
- 단, 상속 관계에 있지 않는 경우에는 당연히 타입 불일치 오류가 발생한다.

오류 발생

- 타입(Type) 불일치(Type Mismatch) 오류

오류 내용

- 'Cat' 타입의 객체가 필요한데 'Dog' 타입의 객체가 제공되었다.

[강제 타입 변환]

- 자식 타입은 부모 타입으로 자동 변환되지만, 반대로 부모 타입은 자식 타입으로 자동 변환되지 않는다.

```

package com.chapter.ch07.example.ex06;
public class Main {
    public static void main(String[] args) {
        Parent parent = new Child();
    }
}

```

```
parent.field1 = "test";
parent.method1();
parent.method2();

// ✕ 불가능
// parent.field2 = "test";
// parent.method3();

// field2와 method3를 사용하기 위해서 강제 타입 변환으로 복원
Child child = (Child) parent;
child.field2 = "test";
child.method3();
    }
}
```