default 메서드

디폴트(default) 메서드

- 인터페이스에 선언되는 메서드는 모두 추상(abstract)메서드
- 하지만, JDK 1.8 버전에서 디폴트(default) 메서드가 추가됨!
- default 메서드는 구현부(바디{})가 있는 일반 메서드
- 단, 접근제어자는 public만 사용가능 하며 생략 가능

```
interface A {
    default void method() {
        System.out.println("구현부가 있는 일반 메서드");
    };
}

class B implements A {
    // default 메서드는 오버라이딩 불필요!
}
```

디폴트 메서드가 추가된 이유

- 시스템이 오래되 많은 요구사항이 발생하여 새롭게 반영하고 싶은 기능들이 생김!(최초 개발 시에는 interface가 적절 했음)
- 새로운 기능을 추가하려면 **인터페이스 수정**은 물론, **구현클래스 전부 오버라이딩** 필요! (즉, 수정할 포인트가 너무 많다)
- default 메서드는 일반 메서드 이므로 **interface에만 추가**하면 구현클래스들 전부 수정할 필요 없음!

애초에 클래스, 추상클래스??

- ❖ 클래스
  - **통일성을 강제**할 수 없다 (오버라이딩이 선택사항)
  - 부모클래스의 **인스턴스 생성**이 가능하다 (버그 발생 가능성)
- ❖ 추상 클래스
  - **다중 상속**이 불가능하다
- default 메서드가 없었을 때 해결방안?

```
interface A {}
abstract class B {}

class C extends B implements A {}
```

interface의 다중 구현은?

- 인터페이스 **다중 구현이 가능한 이유**는 메서드의 구현부가 없기 때문에 동작이 **충돌할 일이 없기 때문**이었다! 하지만 `default` 메서드가 추가되면서 **충돌 가능성이 발생!**
- 충돌 회피 규칙
  - 조상클래스의 메서드와 인터페이스의 `default` 메서드가 충돌하면 **조상 클래스의 메서드가 높은 우선순위**를 갖는다.
  - 인터페이스끼리 충돌하는 메서드가 있는 경우(`default`가 아니더라도) **자식클래스는 반드시 조상의 메서드를 재정의**(`Override`) 해야한다.

충돌 회피 규칙

- **슈퍼클래스 vs 인터페이스**
  (SuperClass win)

```java
class A {
    public void method() { System.out.println("클래스 A");}
}
interface B {
    default void method() { System.out.println("인터페이스 B"); }
}

class C extends A implements B {}

public class Main {
    public static void main(String[] args) {
        C c = new C();
        c.method();
    }
}
```

```
클래스 A
```

- **인터페이스 vs 인터페이스**
  (반드시 Override)

```java
interface A {
    default void method() { System.out.println("인터페이스 A"); }
}
interface B {
    default void method() { System.out.println("인터페이스 B"); }
}

class C implements A, B {
    @Override
    public void method() {
        System.out.println("구현 클래스 C");
    }
}

public class Main {
    public static void main(String[] args) {
        C c = new C();
        c.method();
    }
}
```

```
구현 클래스 C
```