Chapter 03.코틀린 고급 - 05.확장 함수

확장 함수

1. 확장 함수란

- 코틀린은 클래스를 상속하거나 데코레이터 패턴과 같은 디자인 패턴을 사용하지 않고도 클래스를 확장할 수 있는 기능을 제공한다
- 예를들어 일반적으로 수정할 수 없는 코틀린의 표준 라이브러리에 기능을 추가하기 위해 확장을 사용할 수 있다

```
MyStringExtensions.kt


fun String.first(): Char {
    return this[0]
}

fun String.addFirst(char: Char): String {
    return char + this.substring(0)
}

fun main() {
    println("ABCD".first())   // 출력 : A

    println("ABCD".addFirst('Z'))   // 출력 : ZABCD
}
```

- 확장 함수 내부의 this는 확장의 대상이 되는 객체의 참조이다 (이런한 것을 receiver 혹은 수신자 객체라고 부른다)

---

2. 자바로 변환된 확장 함수

- 확장 함수는 실제론 대상 객체를 수정하지 않는다 자바 내부적으로 static 메서드를 만든다
- 첫번째 인자로 확장 대상 객체를 사용한다

```
import org.jetbrains.annotations.NotNull;

public final class Java_MyStringExtensionsKt {

  public static final char first(@NotNull String $this) {
    return $this.charAt(0);
  }
}
```

---

3. 멤버와 중복될 경우

- 확장하려는 클래스에 동일한 명칭의 함수가 존재할 경우 클래스의 멤버 함수가 우선된다

```
class MyExample {

    fun printMessage() = println("클래스 출력")
}

fun MyExample.printMessage() = println("클래스 출력")

fun main() {
    MyExample().printMessage()
    // 클래스 출력
}
```

- 함수의 시그니처가 다른 경우는 문제 없이 확장 기능을 사용할 수 있다

```
class MyExample {

    fun printMessage() = println("클래스 출력")
}

fun MyExample.printMessage(message: String) = println(message)

fun main() {
    MyExample().printMessage("확장 출력")
    // 확장 출력
}
```

---

4. 널 가능성이 있는 클래스에 대한 확장

- null인 경우 내부에서 `this == null` 과 같은 형태로 null 검사를 수행할 수 있다

```
class MyExample {
  /* ... */
}

fun MyExample?.printNullOrNotNull() {
    if (this == null) println("널인 경우에만 출력")
    else println("널이 아닌 경우에만 출력")
}

fun main() {
    var myExample: MyExample? = null
    myExample.printNullOrNotNull()
    // 널인 경우에만 출력

    myExample = MyExample()
    myExample.printNullOrNotNull()
    // 널이 아닌 경우에만 출력
}
```

- 이렇게 처리하면 널 안정성 체크 필요없이 호출할 수 있다