Chapter 02.코틀린 기초 - 10.열거형

열거형

1. eum 클래스 사용

- 코틀린은 서로 연관된 상수들의 집합을 `enum class` 를 사용해서 정의할 수 있다
- 결제 상태를 나타내는 `PaymentStatus` enum 클래스

```
enum class PaymentStatus {
    UNPAID, PAID, FAILED, REFUNDED
}
```

- enum 클래스도 클래스이므로 생성자와 프로퍼티를 정의할 수 있다

```
enum class PaymentStatus(val label: String) {
    UNPAID("미지급"),
    PAID("지급완료"),
    FAILED("지급실패"),
    REFUNDED("환불")
}
```

- 정의된 프로퍼티를 사용하는 방법은 아래와 같다

```
println(PaymentStatus.PAID.label)
// 지급완료
```

- 정의된 상수 목록 뒤에 함수를 정의할 경우 `;` 세미콜론을 붙여야한다
- 결제 가능여부를 확인하는 `isPayable` 함수를 구현

```
enum class PaymentStatus(val label: String) {
    UNPAID("미지급"),
    PAID("지급완료"),
    FAILED("지급실패"),
    REFUNDED("환불");

    fun isPayable(): Boolean = false

}
```

- 각각의 결제 상태에 따라 `isPayable` 상태를 다르게 구현하려한다
- abstract 함수를 가질 수 있고 각각의 상수는 익명 클래스형태로 abstract 함수를 구현할 수 있다

```
enum class PaymentStatus(val label: String) {
    UNPAID("미지급") {
        override fun isPayable() = true
    },
    PAID("지급완료") {
        override fun isPayable() = false
    },
    FAILED("지급실패") {
        override fun isPayable() = false
    },
    REFUNDED("환불") {
        override fun isPayable() = false
    };

    abstract fun isPayable(): Boolean

}

fun main() {
    if (PaymentStatus.UNPAID.isPayable()) {
```

```
        println("결제 가능 상태")
    }
    // 결제 가능 상태
}
```

- enum 클래스에서 인터페이스를 구현할 수 있다

```
enum class PaymentStatus(val label: String) : Payable {
    UNPAID("미지급") {
        override fun isPayable() = true
    },
    PAID("지급완료") {
        override fun isPayable() = false
    },
    FAILED("지급실패") {
        override fun isPayable() = false
    },
    REFUNDED("환불") {
        override fun isPayable() = false
    };

}

interface Payable {
    fun isPayable(): Boolean
}
```

- `valueOf(value:String) : String`를 사용해서 enum 클래스를 생성할 수 있다

```
val paymentStatus = PaymentStatus.valueOf("PAID")
println(paymentStatus.label)
// 지급완료
```

- enum 클래스의 동등성 비교는 `==` 를 사용한다

```
if (paymentStatus == PaymentStatus.PAID) {
  println("결제 완료 상태")
}
// 결제 완료 상태
```

- enum 클래스의 상수를 나열하려면 `values() : Array<EnumClass>` 를 사용한다

```
for (status in PaymentStatus.values()) {
  println("[$status](${status.label})")
}
// [UNPAID](미지급)
// [PAID](지급완료)
// [FAILED](지급실패)
// [REFUNDED](환불)
```

- 상수는 제공하는 2개의 프로퍼티를 사용해 이름을 얻거나 순서를 얻을 수 있다

```
val name: String
val ordinal: Int
```

- 아래와 같이 사용한다

```
for (status in PaymentStatus.values()) {
  println("[${status.name}](${status.label}) : ${status.ordinal}")
}
// [UNPAID](미지급) : 0
// [PAID](지급완료) : 1
// [FAILED](지급실패) : 2
// [REFUNDED](환불) : 3
```