```
import Calendar


enum class Day {
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY
}

fun main() {
    val today = Day.FRIDAY

    //enum meghivása
    val enumday = Calendar()
    for (days in Calendar.Day2.entries){
        println("${days.name} : ${days.message()}")
    }
    println()

    when (today) {
        Day.MONDAY -> println("Hétfő van!")
        Day.FRIDAY -> println("Péntek van, majdnem hétvége!")
        Day.SATURDAY, Day.SUNDAY -> println("Hétvége!")
        else -> println("Munkanap.")
    }
}
```

main.kt

```
class Calendar {
    enum class Day2 {
        MONDAY {
            override fun message() = "Hétfő van!"
        },
        TUESDAY {
            override fun message() = "Kedd van!"
        },
        WEDNESDAY {
            override fun message() = "Szerda van!"
        },
        THURSDAY {
            override fun message() = "Csütörtök van!"
        },
        FRIDAY {
            override fun message() = "Péntek van, majdnem hétvége!"
        },
        SATURDAY {
            override fun message() = "Szombat van, pihenés!"
        },
        SUNDAY {
            override fun message() = "Vasárnap van, nyugi nap!"
        };
```

```
        abstract fun message(): String
    }
}
```

enumDay2.kt

```
enum class Animal(val sound: String) {
    DOG("Woof"),
    CAT("Meow"),
    COW("Moo");

    fun speak() {
        println("The $name says $sound")
    }
}

fun main() {
    Animal.DOG.speak() // The DOG says Woof
}
```

enum2.kt

---

1. **Mit csinál?**
2. **Mi benne a jó gyakorlat, mi nem szükséges?**
3. **Hogyan lehetne egyszerűsíteni, optimalizálni?**

🔍 1. Általános áttekintés

Fő elemek:

- `enum class Day`: egy sima felsorolás a napokkal.
- `class Calendar` → `enum class Day2`: egy beágyazott enum, minden naphoz saját `message()` függvénnyel.
- `enum class Animal`: minden állatnak saját hangja, van egy `speak()` függvénye.
- Két `main()` függvény is van (ez problémás, lásd később).

🧩 2. Részletes elemzés

✅ `enum class Day`

```
enum class Day {
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY
}
```

Ez egy egyszerű enum – nincs benne metódus, csak felsorolás. `when` ágon keresztül jól használható.

⚠️ `val enumday = Calendar()`

```
val enumday = Calendar()
```

Ez **nem szükséges**. Az `enum` osztály (`Day2`) statikus, tehát **nem kell példányosítani a `Calendar` osztályt** ahhoz, hogy hozzáférj az `enum`-hoz.

🟡 Ez itt felesleges:

```
val enumday = Calendar()
```

✅ `Calendar.Day2.entries` használata

```
for (days in Calendar.Day2.entries){
    println("${days.name} : ${days.message()}")
}
```

Ez teljesen helyes, és a `Day2` minden elemét kiírja a hozzá tartozó üzenettel. Jó gyakorlat!

✅ `when (today)`

```
val today = Day.FRIDAY
when (today) {
    Day.MONDAY -> println("Hétfő van!")
    Day.FRIDAY -> println("Péntek van, majdnem hétvége!")
    Day.SATURDAY, Day.SUNDAY -> println("Hétvége!")
    else -> println("Munkanap.")
}
```

Tökéletes példa az `enum` használatára egy `when`-nel.

⚠️ Két `main()` függvényed van!

```
fun main() {
    ...
}

fun main() {
    Animal.DOG.speak()
}
```

Ez **nem engedélyezett Kotlinban**! Csak **egy `main()` függvényed lehet** egy fájlban. Ha több `main()`-re van szükséged (pl. különböző példákhoz), akkor:

- vagy külön fájlba írod őket,
- vagy más néven nevezed meg (`fun animalMain()`),
- vagy összevonod őket egy `main()`-be.

✅ `enum class Animal` és `speak()`

```
enum class Animal(val sound: String) {
    DOG("Woof"),
    CAT("Meow"),
    COW("Moo");

    fun speak() {
        println("The $name says $sound")
    }
}
```

Ez egy szuper példa: enum értékekhez tartozik egy `val`, és egy metódus (`speak`). A `name` a beépített enum név.

🔧 3. Javasolt tisztább verzió

```
enum class Day {
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY
}

class Calendar {
    enum class Day2 {
        MONDAY { override fun message() = "Hétfő van!" },
        TUESDAY { override fun message() = "Kedd van!" },
        WEDNESDAY { override fun message() = "Szerda van!" },
        THURSDAY { override fun message() = "Csütörtök van!" },
        FRIDAY { override fun message() = "Péntek van, majdnem hétvége!" },
        SATURDAY { override fun message() = "Szombat van, pihenés!" },
        SUNDAY { override fun message() = "Vasárnap van, nyugi nap!" };

        abstract fun message(): String
    }
}

enum class Animal(val sound: String) {
    DOG("Woof"), CAT("Meow"), COW("Moo");

    fun speak() {
        println("The $name says $sound")
    }
}

fun main() {
    // 1. Enum kiírása üzenetekkel
    for (day in Calendar.Day2.entries) {
        println("${day.name}: ${day.message()}")
    }

    println()

    // 2. Simább when-ág példa
```

```
    val today = Day.FRIDAY
    when (today) {
        Day.MONDAY -> println("Hétfő van!")
        Day.FRIDAY -> println("Péntek van, majdnem hétvége!")
        Day.SATURDAY, Day.SUNDAY -> println("Hétvége!")
        else -> println("Munkanap.")
    }

    println()

    // 3. Állathang példa
    Animal.DOG.speak()
}
```

✅ Összefoglalás

Dolog	Megjegyzés
`enum` példányosítása	**Nem kell**, közvetlenül az osztályon keresztül elérhető
`Calendar()` példányosítás	**Felesleges** itt
`main()` duplán	**Nem megengedett**, csak egy `main` lehet egy fájlban
`entries`, `message()`	**Jó példa** az enum metódusaira
`enum` osztály `speak()`	**Szintén remek használat** egyedi adattal és metódussal