```
import filterMap.FilterPeople
import filterMap.FilterResult



fun main() {
    val persons = FilterPeople()
    val peopleResult :FilterResult = persons.filterFun()

    println("All Names : ${peopleResult.names}")
    println("Kiírva Zárojel nélkül : ${peopleResult.names.joinToString()}")
    println()
    println("Most egymás alá : ${peopleResult.names.joinToString("\n")}")
    println()//uj sor
    println("olderThan25 : ${peopleResult.olderThan25}")
    println()//uj sor
    println("isAnyoneOlderThan30 : ${peopleResult.isAnyoneOlderThan30}")
    println()
}
```

```
package filterMap

class FilterPeople {
    fun filterFun(): FilterResult {
        val people:List<Person> = listOf(
            Person("Alice",25),
            Person("Bob",30),
            Person("Charlie",22),
            Person("David",35),
            Person("Eva",28),
        )
        val olderThan25 :List<Person> = people.filter { it.age>25 }// a 25 év
alatiakat kiszürjük eltávolitjuk
        val names :List<String> = people.map { it.name }//nevek kiirasa
        val isAnyoneOlderThan30=people.any { it.age>30 }
        return FilterResult(
            olderThan25 = olderThan25,
            names = names,
            isAnyoneOlderThan30 = isAnyoneOlderThan30
        )
    }
}
```

```
package filterMap

data class FilterResult(
val olderThan25: List<Person>,
val names: List<String>,
val isAnyoneOlderThan30: Boolean
)
```

```
package filterMap

data class Person(val name :String, val age :Int)
```

Nézzük át lépésről lépésre a kódod felépítését és működését, hogy tiszta legyen, mit csinál:

---

1. A `Person` adatosztály (data class)

```
package filterMap

data class Person(val name: String, val age: Int)
```

- Ez egy egyszerű adattípus, amely egy személy nevét (`name`) és életkorát (`age`) tárolja.

---

2. A `FilterResult` adatosztály

```
package filterMap

data class FilterResult(
    val olderThan25: List<Person>,
    val names: List<String>,
    val isAnyoneOlderThan30: Boolean
)
```

- Ez a data class összefogja a `filterFun()` eredményeit három mezőben:
  - `olderThan25`: azok a személyek, akik idősebbek 25 évnél.
  - `names`: minden személy neveinek listája.
  - `isAnyoneOlderThan30`: igaz/hamis érték, hogy van-e valaki idősebb 30-nál.

---

3. A `FilterPeople` osztály és a `filterFun` függvény

```
package filterMap

class FilterPeople {
    fun filterFun(): FilterResult {
        val people: List<Person> = listOf(
            Person("Alice", 25),
            Person("Bob", 30),
            Person("Charlie", 22),
            Person("David", 35),
            Person("Eva", 28),
        )
        val olderThan25: List<Person> = people.filter { it.age > 25 }
        val names: List<String> = people.map { it.name }
        val isAnyoneOlderThan30 = people.any { it.age > 30 }
        return FilterResult(
```

```
            olderThan25 = olderThan25,
            names = names,
            isAnyoneOlderThan30 = isAnyoneOlderThan30
        )
    }
}
```

- A `people` listában 5 `Person` van.
- `filter`-rel kiszűröd azokat, akiknek az életkora nagyobb mint 25.
- `map`-pel kigyűjtöd az összes nevet.
- `any`-vel ellenőrzöd, hogy van-e 30 évnél idősebb személy.
- Ezeket az eredményeket egy `FilterResult` objektumba csomagolod és visszaadod.

---

4. A `main()` függvény, ahol az eredmény kiírásra kerül

```
import filterMap.FilterPeople
import filterMap.FilterResult

fun main() {
    val persons = FilterPeople()
    val peopleResult: FilterResult = persons.filterFun()

    println("All Names : ${peopleResult.names}")
    println("Kiírva Zárojel nélkül : ${peopleResult.names.joinToString()}")
    println()
    println("Most egymás alá : ${peopleResult.names.joinToString("\n")}")
    println() // új sor
    println("olderThan25 : ${peopleResult.olderThan25}")
    println() // új sor
    println("isAnyoneOlderThan30 : ${peopleResult.isAnyoneOlderThan30}")
}
```

- Létrehozol egy `FilterPeople` objektumot.
- Meghívod a `filterFun()`-t, ami egy `FilterResult`-ot ad vissza.
- Kiírod a nevek listáját több formában (az első sorban sima lista, a másodikban `joinToString()`-gel zárójelek nélkül, majd egyenként sorba tördelve).
- Kiírod az idősebb mint 25 éveseket (ez a személyek listája lesz).
- Kiírod, hogy van-e valaki 30 évnél idősebb.

Elemzés, megjegyzések

- A kód jól strukturált, a funkciók világosan szét vannak választva.
- A `filter` használata egyszerű és érthető, jól szűri az adatokat.
- A `map`-pel csak a nevek listáját kéred le, ami szintén helyes.
- A `any` használata hatékony módja annak, hogy megtudd, létezik-e olyan elem, ami megfelel a feltételnek.
- A `joinToString()` nagyon jól használható listák szép megjelenítésére.

Tippek esetleg:

- Ha a `olderThan25`-ből csak a neveket szeretnéd, szintén lehet `map`-pelni, pl.:

```
val olderThan25Names = olderThan25.map { it.name }
```

- A kiírásnál a `olderThan25` listát úgy, ahogy van, a Kotlin alapból a `toString()`-et hívja meg rajta, ami elég sok infót tartalmaz (pl. `Person(name=David, age=35)`). Ha csak neveket akarsz, használd a fenti `map`-et.
- A kód magyar kommentekkel van ellátva, ami segíti az átláthatóságot.

Szóval összefoglalva:

Ez a program létrehoz egy személylistát, kiszűri belőle azokat, akik 25 évnél idősebbek, kigyűjti az összes nevet, és ellenőrzi, hogy van-e valaki 30 évnél idősebb. Az eredményeket egy `FilterResult` objektumba rendezi, majd a `main` kiírja azokat.
