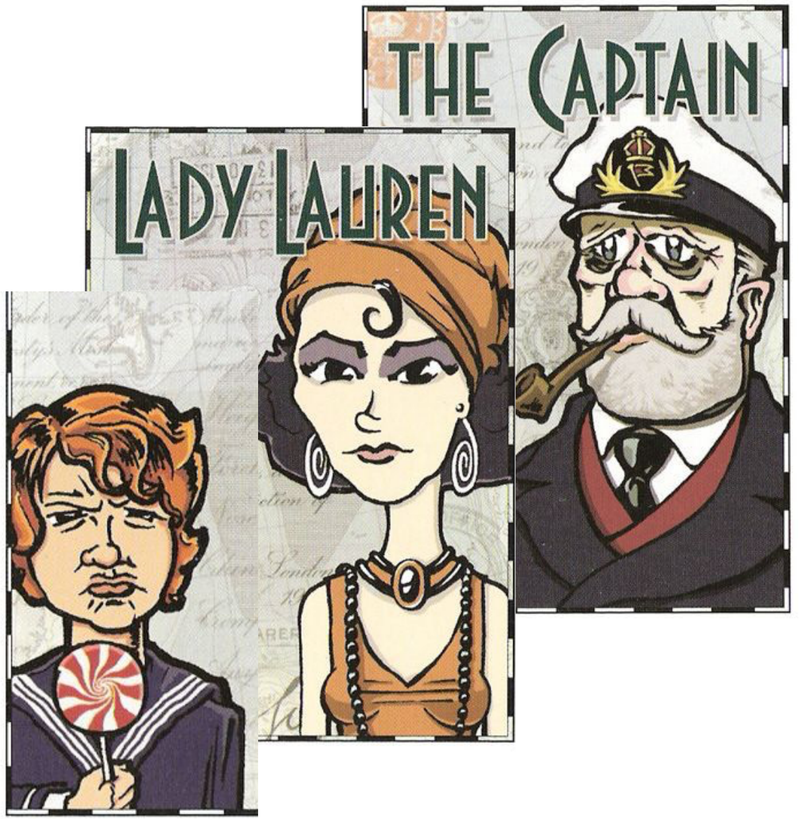
Operations quiz

The class `Hero`

```
data class Hero(
        val name: String,
        val age: Int,
        val gender: Gender?
)
enum class Gender { MALE, FEMALE }
```

Heroes

```
val heroes = listOf(
        Hero("The Captain", 60, MALE),
        Hero("Frenchy", 42, MALE),
        Hero("The Kid", 9, null),
        Hero("Lady Lauren", 29, FEMALE),
        Hero("First Mate", 29, MALE),
        Hero("Sir Stephen", 37, MALE))
```



#1. Find the result of the expression

```
val heroes = listOf(
        Hero("The Captain", 60, MALE),
        Hero("Frenchy", 42, MALE),
        Hero("The Kid", 9, null),
        Hero("Lady Lauren", 29, FEMALE),
        Hero("First Mate", 29, MALE),
        Hero("Sir Stephen", 37, MALE))
```

```
heroes.last().name
```

first, last, firstOrNull, lastOrNull

```
val heroes = listOf(
        Hero("The Captain", 60, MALE),
        Hero("Frenchy", 42, MALE),
        Hero("The Kid", 9, null),
        Hero("Lady Lauren", 29, FEMALE),
        Hero("First Mate", 29, MALE),
        Hero("Sir Stephen", 37, MALE))
```

```
heroes.last().name
```

Sir Stephen

#2. Find the result

```
val heroes = listOf(
        Hero("The Captain", 60, MALE),
        Hero("Frenchy", 42, MALE),
        Hero("The Kid", 9, null),
        Hero("Lady Lauren", 29, FEMALE),
        Hero("First Mate", 29, MALE),
        Hero("Sir Stephen", 37, MALE))
```

```
heroes.firstOrNull { it.age == 30 }?.name
```

first { … }, last { … }, firstOrNull { … }, lastOrNull { … }

```
val heroes = listOf(
        Hero("The Captain", 60, MALE),
        Hero("Frenchy", 42, MALE),
        Hero("The Kid", 9, null),
        Hero("Lady Lauren", 29, FEMALE),
        Hero("First Mate", 29, MALE),
        Hero("Sir Stephen", 37, MALE))
```

```
heroes.firstOrNull { it.age == 30 }?.name
```

null

```
heroes.first { it.age == 30 }.name
                        // NoSuchElementException
```

#3. Find the result

```
val heroes = listOf(
        Hero("The Captain", 60, MALE),
        Hero("Frenchy", 42, MALE),
        Hero("The Kid", 9, null),
        Hero("Lady Lauren", 29, FEMALE),
        Hero("First Mate", 29, MALE),
        Hero("Sir Stephen", 37, MALE))
```

```
heroes.map { it.age }.distinct().size
```

`distinct`: finding only distinct elements

```
val heroes = listOf(
        Hero("The Captain", 60, MALE),
        Hero("Frenchy", 42, MALE),
        Hero("The Kid", 9, null),
        Hero("Lady Lauren", 29, FEMALE),
        Hero("First Mate", 29, MALE),
        Hero("Sir Stephen", 37, MALE))
```

```
heroes.map { it.age }                    // [60, 42, 9, 29, 29, 37]
heroes.map { it.age }.distinct()         // [60, 42, 9, 29, 37]
heroes.map { it.age }.distinct().size    5
```

#4. Find the result

```
val heroes = listOf(
        Hero("The Captain", 60, MALE),
        Hero("Frenchy", 42, MALE),
        Hero("The Kid", 9, null),
        Hero("Lady Lauren", 29, FEMALE),
        Hero("First Mate", 29, MALE),
        Hero("Sir Stephen", 37, MALE))
```

```
heroes.filter { it.age < 30 }.size
```

Filtering the list contents

```
val heroes = listOf(
        Hero("The Captain", 60, MALE),
        Hero("Frenchy", 42, MALE),
        Hero("The Kid", 9, null),
        Hero("Lady Lauren", 29, FEMALE),
        Hero("First Mate", 29, MALE),
        Hero("Sir Stephen", 37, MALE))
```

```
heroes.filter { it.age < 30 }.size
```

3

#5. Find the result

```
val heroes = listOf(
        Hero("The Captain", 60, MALE),
        Hero("Frenchy", 42, MALE),
        Hero("The Kid", 9, null),
        Hero("Lady Lauren", 29, FEMALE),
        Hero("First Mate", 29, MALE),
        Hero("Sir Stephen", 37, MALE))
```

```
val (youngest, oldest) = heroes.partition { it.age < 30 }

oldest.size
```

`partition`: two collections as a result

```
val heroes = listOf(
        Hero("The Captain", 60, MALE),
        Hero("Frenchy", 42, MALE),
        Hero("The Kid", 9, null),
        Hero("Lady Lauren", 29, FEMALE),
        Hero("First Mate", 29, MALE),
        Hero("Sir Stephen", 37, MALE))
```

```
val (youngest, oldest) = heroes.partition { it.age < 30 }

oldest.size
```

3

#6. Find the result

```
val heroes = listOf(
        Hero("The Captain", 60, MALE),
        Hero("Frenchy", 42, MALE),
        Hero("The Kid", 9, null),
        Hero("Lady Lauren", 29, FEMALE),
        Hero("First Mate", 29, MALE),
        Hero("Sir Stephen", 37, MALE))
```

```
heroes.maxBy { it.age }?.name
```

Finding the maximum

```
val heroes = listOf(
        Hero("The Captain", 60, MALE),
        Hero("Frenchy", 42, MALE),
        Hero("The Kid", 9, null),
        Hero("Lady Lauren", 29, FEMALE),
        Hero("First Mate", 29, MALE),
        Hero("Sir Stephen", 37, MALE))
```

```
heroes.maxBy { it.age }?.name
```

The Captain

#7. Find the result

```
val heroes = listOf(
        Hero("The Captain", 60, MALE),
        Hero("Frenchy", 42, MALE),
        Hero("The Kid", 9, null),
        Hero("Lady Lauren", 29, FEMALE),
        Hero("First Mate", 29, MALE),
        Hero("Sir Stephen", 37, MALE))
```

```
heroes.all { it.age < 50 }
```

Predicates

```
val heroes = listOf(
        Hero("The Captain", 60, MALE),
        Hero("Frenchy", 42, MALE),
        Hero("The Kid", 9, null),
        Hero("Lady Lauren", 29, FEMALE),
        Hero("First Mate", 29, MALE),
        Hero("Sir Stephen", 37, MALE))
```

```
heroes.all { it.age < 50 }
```

false

#8. Find the result

```
val heroes = listOf(
        Hero("The Captain", 60, MALE),
        Hero("Frenchy", 42, MALE),
        Hero("The Kid", 9, null),
        Hero("Lady Lauren", 29, FEMALE),
        Hero("First Mate", 29, MALE),
        Hero("Sir Stephen", 37, MALE))
```

```
heroes.any { it.gender == FEMALE }
```

Predicates

```
val heroes = listOf(
        Hero("The Captain", 60, MALE),
        Hero("Frenchy", 42, MALE),
        Hero("The Kid", 9, null),
        Hero("Lady Lauren", 29, FEMALE),
        Hero("First Mate", 29, MALE),
        Hero("Sir Stephen", 37, MALE))
```

```
heroes.any { it.gender == FEMALE }
```

```
true
```