Member references

Member references

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

people.maxBy { it.age }
```

Convert lambda to reference

```
people.maxBy(Person::age)
```

class

member

You can store lambda in a variable

```
val isEven: (Int) -> Boolean =
        { i: Int -> i % 2 == 0 }
```

…but you can’t store a function in a variable

```
fun isEven(i: Int): Boolean = i % 2 == 0

val predicate = isEven
```

Compiler error

Use function reference instead

```
fun isEven(i: Int): Boolean = i % 2 == 0

val predicate = ::isEven

val predicate = { i: Int -> isEven(i) }
```

Member references

```
val action = { person: Person, message: String ->
    sendEmail(person, message)
}
```

```
val action = ::sendEmail
```

Passing function reference as argument

```
fun isEven(i: Int): Boolean = i % 2 == 0

val list = listOf(1, 2, 3, 4)
list.any(::isEven)                    true
list.filter(::isEven)                 [2, 4]
```

Bound & non-bound references

```
class Person(val name: String, val age: Int) {
    fun isOlder(ageLimit: Int) = age > ageLimit
}

val agePredicate = Person::isOlder

val alice = Person("Alice", 29)
agePredicate(alice, 21)          // true
```

regular non-bound reference

Bound & non-bound references

```
class Person(val name: String, val age: Int) {
    fun isOlder(ageLimit: Int) = age > ageLimit
}


val agePredicate: (Person, Int) -> Boolean =
                                Person::isOlder
val alice = Person("Alice", 29)
agePredicate(alice, 21)          // true
```

Bound & non-bound references

```
class Person(val name: String, val age: Int) {
    fun isOlder(ageLimit: Int) = age > ageLimit
}


val agePredicate: (Person, Int) -> Boolean =
                                Person::isOlder

val alice = Person("Alice", 29)
agePredicate(alice, 21)         // true
```

Non-bound reference: the corresponding lambda

```
class Person(val name: String, val age: Int) {
    fun isOlder(ageLimit: Int) = age > ageLimit
}

val agePredicate: (Person, Int) -> Boolean =
                                { person, ageLimit ->
                                  person.isOlder(ageLimit) }

val alice = Person("Alice", 29)
agePredicate(alice, 21)          // true
```

Bound reference

```
class Person(val name: String, val age: Int) {
    fun isOlder(ageLimit: Int) = age > ageLimit
}


val alice = Person("Alice", 29)
val agePredicate = alice::isOlder
agePredicate(21)                    // true
```

Bound reference

```
class Person(val name: String, val age: Int) {
    fun isOlder(ageLimit: Int) = age > ageLimit
}


val alice = Person("Alice", 29)
val agePredicate: (Int) -> Boolean = alice::isOlder
agePredicate(21)                    // true
```

Bound reference

```
class Person(val name: String, val age: Int) {
    fun isOlder(ageLimit: Int) = age > ageLimit
}


val alice = Person("Alice", 29)
val agePredicate: (Int) -> Boolean = alice::isOlder
agePredicate(21)                    // true
```

Bound reference: the corresponding lambda

```
class Person(val name: String, val age: Int) {
    fun isOlder(ageLimit: Int) = age > ageLimit
}


val alice = Person("Alice", 29)

val agePredicate: (Int) -> Boolean =
                        { ageLimit -> alice.isOlder(ageLimit) }

agePredicate(21)                    // true
```

Bound to `this` reference

```
class Person(val name: String, val age: Int) {
    fun isOlder(ageLimit: Int) = age > ageLimit

    fun getAgePredicate() = this::isOlder
}
```

`this` can be omitted

Bound to `this` reference

```
class Person(val name: String, val age: Int) {
    fun isOlder(ageLimit: Int) = age > ageLimit

    fun getAgePredicate() = ::isOlder
}
```

What is the type of `::isOlder` here?

```
class Person(val name: String, val age: Int) {
    fun isOlder(ageLimit: Int) = age > ageLimit

    fun getAgePredicate() = ::isOlder
}
```

1. `(Person, Int) -> Boolean`
2. `(Int) -> Boolean`

What is the type of `::isOlder` here?

```
class Person(val name: String, val age: Int) {
    fun isOlder(ageLimit: Int) = age > ageLimit

    fun getAgePredicate() = ::isOlder
}
```

1. `(Person, Int) -> Boolean`
2. `(Int) -> Boolean`

What is the type of `::isOlder` here?

```
class Person(val name: String, val age: Int) {
    fun isOlder(ageLimit: Int) = age > ageLimit

    fun getAgePredicate() = this::isOlder
}
```

1. `(Person, Int) -> Boolean`
2. `(Int) -> Boolean`

Is `::isEven` a bound reference?

```
fun isEven(i: Int): Boolean = i % 2 == 0


val list = listOf(1, 2, 3, 4)
list.any(::isEven)
list.filter(::isEven)
```

1. yes
2. no

Is `::isEven` a bound reference?

```
fun isEven(i: Int): Boolean = i % 2 == 0


val list = listOf(1, 2, 3, 4)
list.any(::isEven)
list.filter(::isEven)
```

reference to
top-level function

1. yes
2. no