Class modifiers

Class modifiers

`enum, data, inner, sealed`

`enum class`

`enum class`

represents enumeration

```
import Color.*

enum class Color {
    BLUE, ORANGE, RED
}

fun getDescription(color: Color) =
    when (color) {
        BLUE -> "cold"
        ORANGE -> "mild"
        RED -> "hot"
    }
```

Importing enum constants

```kotlin
package mypackage

enum class Color {
    BLUE, ORANGE, RED
}

fun getDescription(color: Color) =
    when (color) {
        Color.BLUE -> "cold"
        Color.ORANGE -> "mild"
        Color.RED -> "hot"
    }
```

Importing enum constants

```
package mypackage

import mypackage.Color.*

enum class Color {
    BLUE, ORANGE, RED
}

fun getDescription(color: Color) =
    when (color) {
        BLUE -> "cold"
        ORANGE -> "mild"
        RED -> "hot"
    }
```

enum class with properties

```
enum class Color(
        val r: Int, val g: Int, val b: Int
) {
    BLUE(0, 0, 255), ORANGE(255, 165, 0), RED(255, 0, 0);

    fun rgb() = (r * 256 + g) * 256 + b
}


println(BLUE.r)             // 0
println(BLUE.rgb())         // 255
```

`data class`

`data` modifier

Generates useful methods:

`equals`, `hashCode`, `copy`, `toString`, and some others

data modifier

```kotlin
data class Contact(val name: String, val address: String)

contact.copy(address = "new address")
```

Equals & reference equality

```
val set1 = setOf(1, 2, 3)
val set2 = setOf(1, 2, 3)
```

calls equals

```
set1 == set2
```

true

checks reference equality

```
set1 === set2
```

false

What will be printed?

```
class Foo(val first: Int, val second: Int)
data class Bar(val first: Int, val second: Int)

val f1 = Foo(1, 3)
val f2 = Foo(1, 3)
println(f1 == f2)

val b1 = Bar(1, 3)
val b2 = Bar(1, 3)
println(b1 == b2)
```

1. true true
2. true false
3. false true
4. false false

?

What will be printed?

```
class Foo(val first: Int, val second: Int)
data class Bar(val first: Int, val second: Int)

val f1 = Foo(1, 2)
val f2 = Foo(1, 2)
println(f1 == f2)

val b1 = Bar(1, 2)
val b2 = Bar(1, 2)
println(b1 == b2)
```

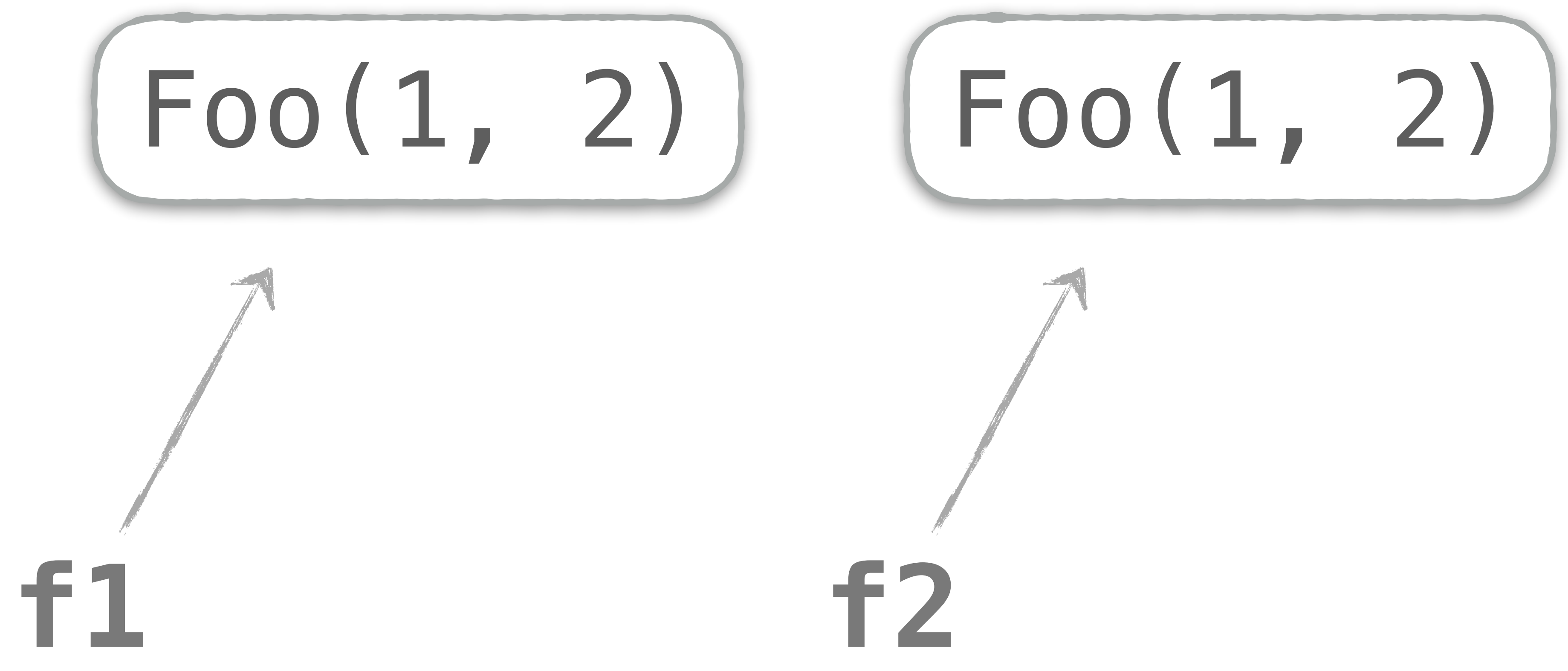
1. true true
2. true false
3. [✓ false true]
4. false false

Default `equals` checks reference equality

```kotlin
class Foo(val first: Int, val second: Int)

val f1 = Foo(1, 2)
val f2 = Foo(1, 2)
println(f1 == f2)
```

```
false
```

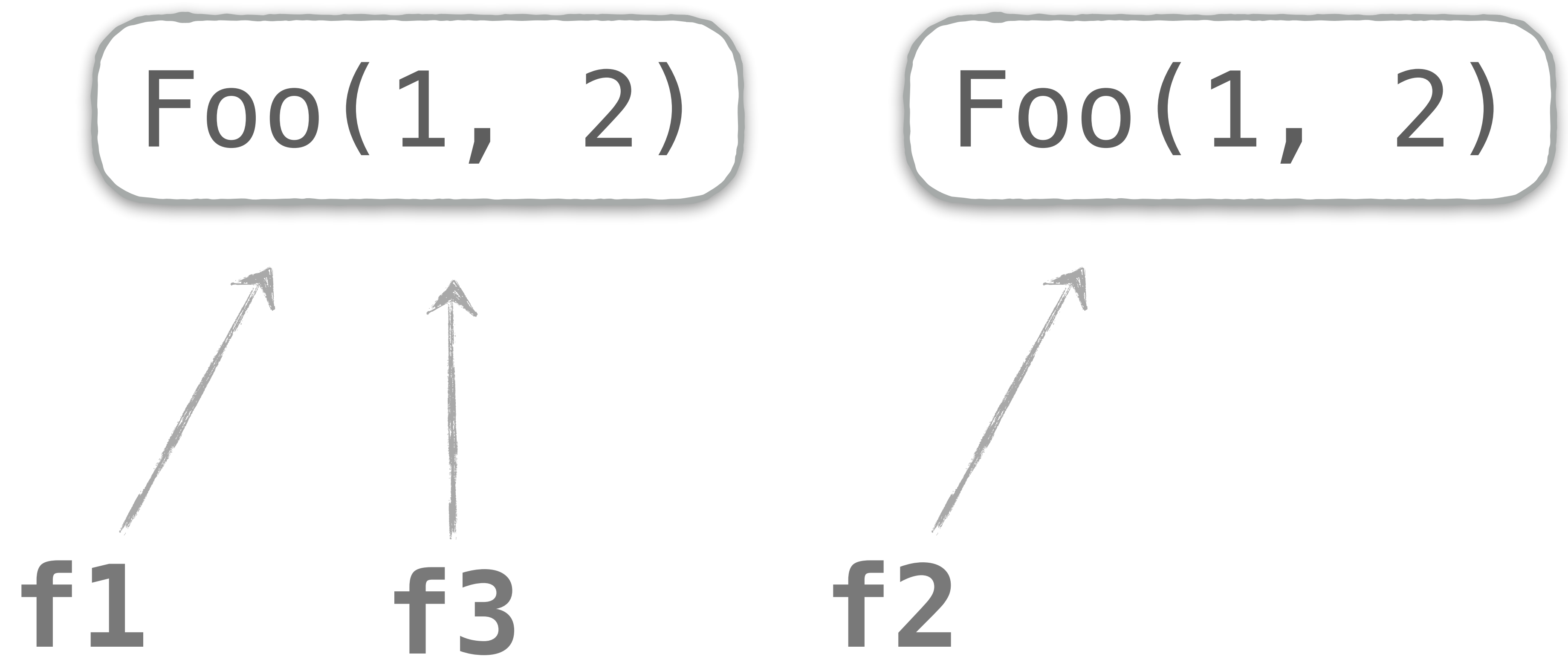


Default `equals` checks reference equality

```
class Foo(val first: Int, val second: Int)

val f1 = Foo(1, 2)
val f2 = Foo(1, 2)
println(f1 == f2) // false

val f3 = f1
println(f1 == f3) // true
```



Generated `equals` compares content

```
data class Bar(val first: Int, val second: Int) {
                              generated methods
    override fun equals(other: Any?): Boolean {
        if (this === other) return true
        if (other !is Bar) return false
        return (first == other.first
                && second == other.second)
    }

    override fun hashCode(): Int =
            first * 31 + second
}
```

Generated `equals` compares content

```kotlin
data class Bar(val first: Int, val second: Int)

val b1 = Bar(1, 2)
val b2 = Bar(1, 2)
println(b1 == b2)
```

```
true
```