

# Object expressions and Declarations

Sometimes we need to create an object of a slight modification of some class, without explicitly declaring a new subclass for it. **Java** handles this case with *anonymous inner classes*. **Kotlin** slightly generalizes this concept with **object** expressions and **object** declarations.

## Object expressions

To create an object of an anonymous class that inherits from some type (or types), one writes:

```
window.addMouseListener(object : MouseAdapter() {
    override fun mouseClicked(e : MouseEvent) {
        // ...
    }

    override fun mouseEntered(e : MouseEvent) {
        // ...
    }
})
```

If a supertype has a constructor, appropriate constructor parameters must be passed to it. Many supertypes may be specified as a comma-separated list after the colon:

```
open class A(x : Int) {
    public open val y : Int = x
}

trait B {...}

val ab = object : A(1), B {
    override val y = 15
}
```

If, by any chance, we need "just an object", with no nontrivial supertypes, we can simply say:

```
val adHoc = object {
    var x : Int = 0
    var y : Int = 0
}

print(adHoc.x + adHoc.y)
```

## Object declarations

**Singleton** is a very useful pattern, and **Kotlin** (after **Scala**) makes it easy to declare singletons:

```
object DataManager {
    fun registerDataProvider(provider : DataProvider) {
        // ...
    }

    val allDataProviders : Collection<DataProvider>
    get() = // ...
}
```

This is called an *object declaration*. If there's a name following the **object** keyword, we are not talking about an *expression* any more. We cannot assign such a thing to a variable, but we can refer to it by its name. Such objects can have supertypes:

```
object DefaultListener : MouseAdapter() {
    override fun mouseClicked(e : MouseEvent) {
        // ...
    }

    override fun mouseEntered(e : MouseEvent) {
        // ...
    }
}
```

## Semantical difference between object expressions and declarations

There is one important semantical difference between **object** expressions and **object** declarations:

- **object** declarations are initialized lazily, when accessed for the first time
- **object** expressions are executed (and initialized) immediately, where they are used

### What's next

- [Generics](#)