### **Constructors**

A class in Kotlin can have a **primary constructor** and one or more **secondary constructors**. The primary constructor is part of the class header: it goes after the class name (and optional type parameters).

**class** Person **constructor**(firstName: String) {

}

If the primary constructor does not have any annotations or visibility modifiers, the constructor keyword can be omitted:

**class** Person(firstName: String) {

}

The primary constructor cannot contain any code. Initialization code can be placed in **initializer blocks**, which are prefixed with the init keyword:

**class** Customer(name: String) {

**init** {

logger.info("Customer initialized with value ${name}")

}

}

Note that parameters of the primary constructor can be used in the initializer blocks. They can also be used in property initializers declared in the class body:

**class** Customer(name: String) {

**val** customerKey = name.toUpperCase()

}

In fact, for declaring properties and initializing them from the primary constructor, Kotlin has a concise syntax:

**class** Person(**val** firstName: String, **val** lastName: String, **var** age: Int) {

// ...

}

Much the same way as regular properties, the properties declared in the primary constructor can be mutable (var) or read-only (val).

If the constructor has annotations or visibility modifiers, the constructor keyword is required, and the modifiers go before it:

**class** Customer **public** @Inject **constructor**(name: String) { ... }

**Secondary Constructors**

The class can also declare **secondary constructors**, which are prefixed with constructor:

**class** Person {

**constructor**(parent: Person) {

parent.children.add(**this**)

}

}

If the class has a primary constructor, each secondary constructor needs to delegate to the primary constructor, either directly or indirectly through another secondary constructor(s). Delegation to another constructor of the same class is done using the this keyword:

**class** Person(**val** name: String) {

**constructor**(name: String, parent: Person) : **this**(name) {

parent.children.add(**this**)

}

}

If a non-abstract class does not declare any constructors (primary or secondary), it will have a generated primary constructor with no arguments. The visibility of the constructor will be public. If you do not want your class to have a public constructor, you need to declare an empty primary constructor with non-default visibility:

**class** DontCreateMe **private** **constructor** () {

}

**NOTE**: On the JVM, if all of the parameters of the primary constructor have default values, the compiler will generate an additional parameterless constructor which will use the default values. This makes it easier to use Kotlin with libraries such as Jackson or JPA that create class instances through parameterless constructors.

**class** Customer(**val** customerName: String = "")

### **Creating instances of classes**

To create an instance of a class, we call the constructor as if it were a regular function:

**val** invoice = Invoice()

**val** customer = Customer("Joe Smith")

Note that Kotlin does not have a new keyword.

### **Class Members**

Classes can contain:

* [Constructors and initializer blocks](https://kotlinlang.org/docs/reference/classes.html#constructors)
* [Functions](https://kotlinlang.org/docs/reference/functions.html)
* [Properties](https://kotlinlang.org/docs/reference/properties.html)
* [Nested and Inner Classes](https://kotlinlang.org/docs/reference/nested-classes.html)
* [Object Declarations](https://kotlinlang.org/docs/reference/object-declarations.html)