



# Kotlin Classes

---

## Sources:

<http://kotlinlang.org/docs/reference/basic-syntax.html>

<http://petersommerhoff.com/dev/kotlin/kotlin-for-java-devs/>

<https://www.programiz.com/kotlin-programming>

<https://medium.com/@napperley/kotlin-tutorial-5-basic-collections-3f114996692b>

# Writing Classes – properties

---

In Kotlin, classes cannot have fields; they have properties.

**var** properties are mutable.

**val** properties cannot be changed.

# Writing Classes – 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 and it goes after the class name:

```
class Person constructor(firstName: String) {  
}
```

```
class Person(firstName: String, lastName: String) {  
}
```


```
class Person(val firstName: String, val lastName: String) {  
}
```

# Writing Classes – primary constructors

---

```
class Person(val firstName: String, val lastName: String) {  
  
}
```

```
fun main(args: Array<String>) {  
  
    val person = Person("Joe", "Soap")  
  
    println("First Name = ${person.firstName}")  
    println("Surname = ${person.lastName}")  
}
```

 Console X

<terminated> Config - Main.kt [Java Application] C:

First Name = Joe

Surname = Soap

```
class Person( _firstName: String = "UNKNOWN FIRSTNAME",
              _lastName: String = "UNKNOWN LASTNAME") {

    val firstName = _firstName
    val lastName  = _lastName

    // initializer block
    init {
        println("First Name = $firstName")
        println("Last Name = $lastName\n")
    }
}
```

## Writing Classes – primary constructors

The **primary constructor** cannot contain any code; initialisation code is placed in the **init** block.

The use of `_` prefixing constructor variables is standard.

```
class Person( _firstName: String = "UNKNOWN FIRSTNAME",
              _lastName: String = "UNKNOWN LASTNAME") {

    val firstName = _firstName
    val lastName  = _lastName

    // initializer block
    init {
        println("First Name = $firstName")
        println("Last Name = $lastName\n")
    }
}
```

# Writing Classes – primary constructors

```
fun main(args: Array<String>) {

    println("person1 is instantiated")
    val person1 = Person("Joe", "Soap")

    println("person2 is instantiated")
    val person2 = Person("Jack")

    println("person3 is instantiated")
    val person3 = Person()

}
```

# Writing Classes – secondary constructors

---

The **secondary constructor** is prefixed with the keyword **constructor**. They are not very common in Kotlin.

More info here: <http://kotlinlang.org/docs/reference/classes.html>

```
class Person {  
  
    constructor(parent: Person) {  
        parent.children.add(this)  
    }  
}
```

# Writing Classes – getters and setters

---

In Kotlin, getters (val and var) and setters (var) are optional and are auto-generated if you do not create them in your program.

```
class Person {  
    var name: String = "defaultValue"  
}
```

Is  
equivalent  
to

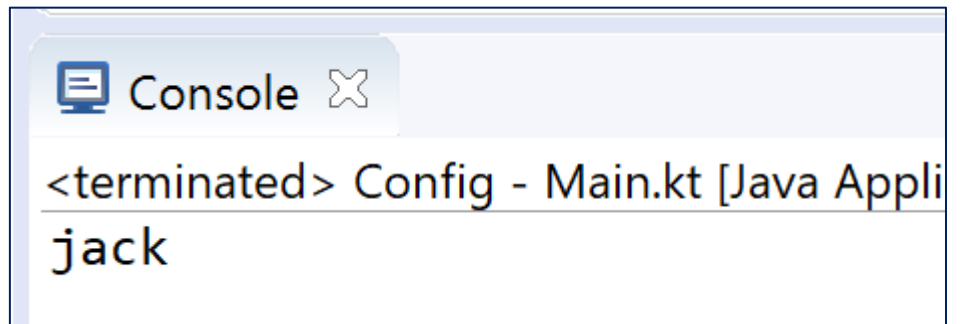
```
class Person {  
    var name: String = "defaultValue"  
  
    // getter  
    get() = field  
  
    // setter  
    set(value) {  
        field = value  
    }  
}
```



# Writing Classes – getters and setters

---

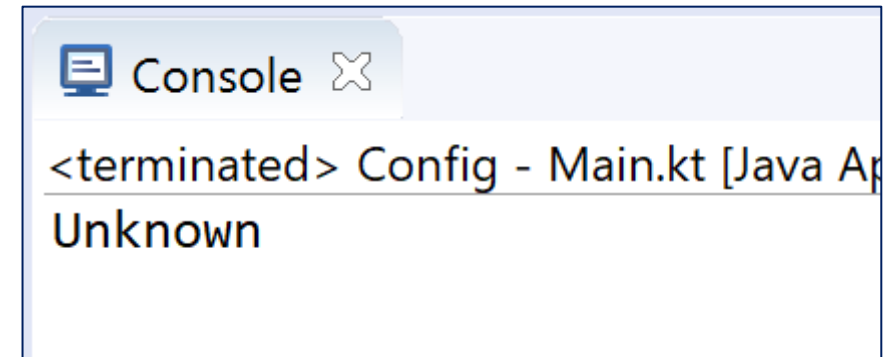
```
fun main(args: Array<String>) {  
  
    val person = Person()  
    person.name = "jack"  
    print(person.name)  
  
}
```



```
class Person {  
    var name: String = "defaultValue"  
  
    // getter  
    get() = field  
  
    // setter  
    set(value) {  
        field = value  
    }  
  
}
```

# Writing Classes – getters and setters

```
fun main(args: Array<String>) {  
  
    val person = Person()  
    person.name = ""  
    print(person.name)  
  
}
```



When you want to  
add validation to your  
setter...

```
class Person {  
    var name: String = "defaultValue"  
  
    get() = field  
  
    set(value) {  
        field = if (value.equals(""))  
                "Unknown"  
                else  
                value  
    }  
}
```

# Data Classes

---

# Data Classes

---

We have created classes to solely to hold data (i.e. models).

We can use the `data` class prefix to simply create a data class.

The compiler automatically generates methods such as `equals()`, `hashCode()`, `toString()`, `copy()` from the primary constructor.

```
data class Person( var firstName: String,  
                  var lastName: String)  
{  
}
```

# Data Classes - Requirements

---

1. The primary constructor must have at least one parameter
2. The parameters of the primary constructor must be marked as either `var` or `val`
3. The class cannot be open, abstract, inner or sealed
4. The class may extend other classes or implement interfaces

```
data class Person( var firstName: String,  
                  var lastName: String)  
{  
}
```



# Data Classes – `copy` and `toString` Example

```
data class Person( var firstName: String,
                   var lastName: String) {
}
```

```
fun main(args: Array<String>) {
    val person1 = Person("John", "Murphy")

    // using copy function to create an object
    val person2 = person1.copy(firstName="Martin")



    println(person1)
    println(person2.toString())
}
```

 Console 

```
<terminated> Config - Main.kt [Java Application] C:\Program Files\J
Person(firstName=John, lastName=Murphy)
Person(firstName=Martin, lastName=Murphy)
```

# Data Classes – `copy`, `equals` and `hashCode` Example

```
fun main(args: Array<String>) {  
    val person1 = Person("John", "Murphy")  
    val person2 = person1.copy()  
    val person3 = person1.copy(firstName = "Martin")  
  
    println("person1 hashCode = ${person1.hashCode()}")  
    println("person2 hashCode = ${person2.hashCode()}")  
    println("person3 hashCode = ${person3.hashCode()}")  
  
    if (person1.equals(person2))  
        println("person1 is equal to person2.")  
    else  
        println("person1 is not equal to person2.")  
  
    if (person1.equals(person3))  
        println("person1 is equal to person3.")  
    else  
        println("person1 is not equal to person3.")  
}
```

 Console 

```
<terminated> Config - Main.kt [Java Application] C  
person1 hashCode = -1907212852  
person2 hashCode = -1907212852  
person3 hashCode = 525212252  
person1 is equal to person2.  
person1 is not equal to person3.
```