



Properties

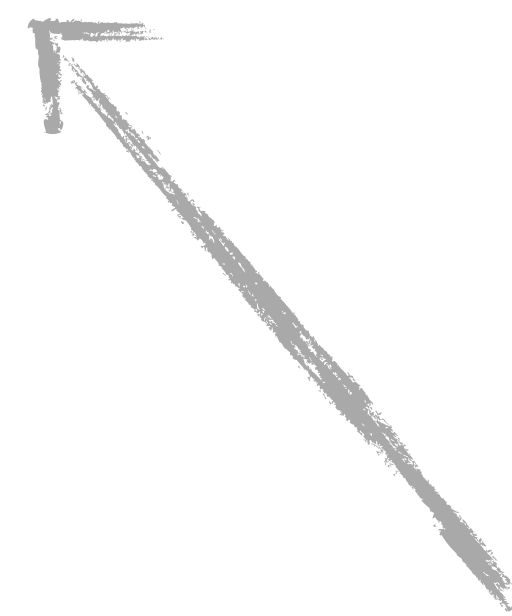
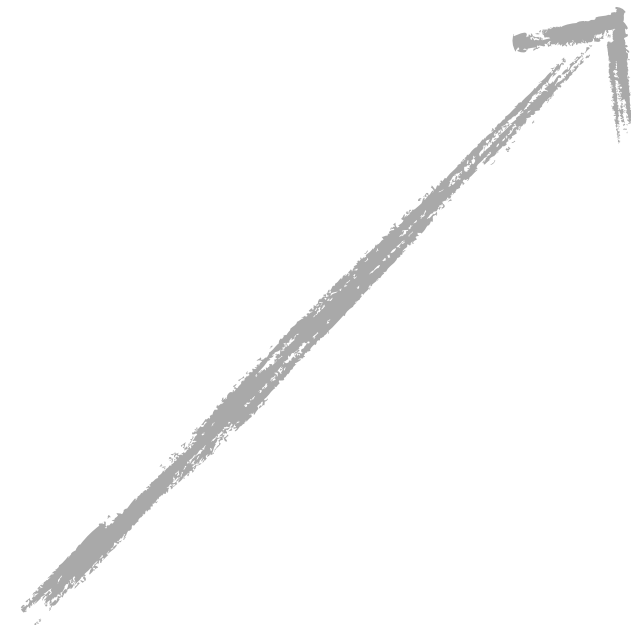

Property (val / var)

property = field + accessor(s)

read-only property = field + getter

mutable property = field + getter + setter


```
class Contact(  
    val name: String,  
    var address: String  
)
```



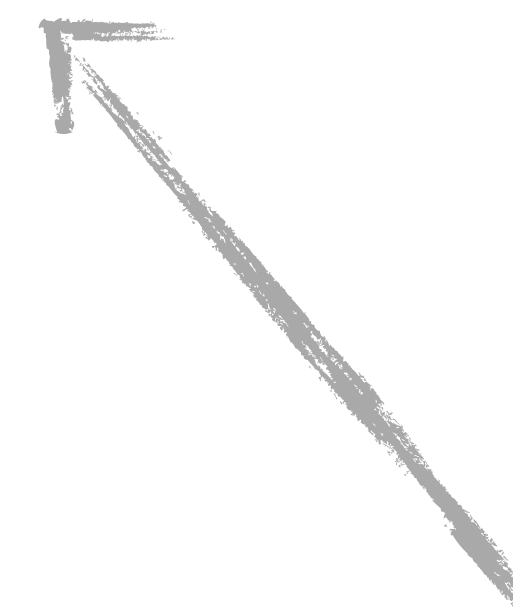
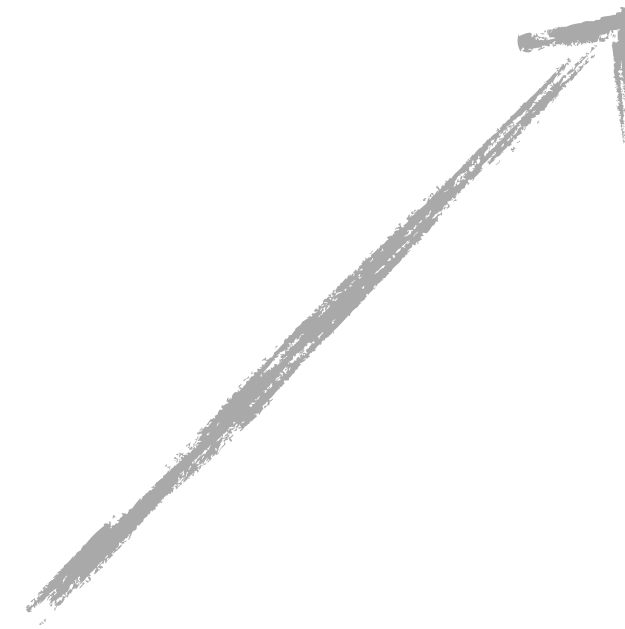
```
contact.address  
contact.address = "..."
```



```
contact.getAddress();  
contact.setAddress("...");
```



```
class Contact {  
    ...  
}
```



```
contact.address  
contact.address = "..."
```



```
contact.getAddress();  
contact.setAddress("...");
```





How many methods (excluding constructors) does the class Person have from Java's point of view?

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





How many methods (excluding constructors) does the class Person have from Java's point of view?

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

```
getName  
getAge  
setAge
```

3

```
public final class Person {
    @NotNull
    private final String name;
    private int age;

    public Person(@NotNull String name, int age) {
        this.name = name;
        this.age = age;
    }

    @NotNull
    public final String getName() {
        return this.name;
    }

    public final int getAge() {
        return this.age;
    }

    public final void setAge(int age) {
        this.age = age;
    }
}
```




Properties without fields

Backing field might be absent

property = (field) + accessor(s)

read-only property = (field) + getter

mutable property = (field) + getter + setter

Backing field might be absent

```
class Rectangle(val height: Int, val width: Int) {  
    val isSquare: Boolean  
        get() {  
            return height == width  
        }  
}
```



How many times the phrase “Calculating the answer...” will be printed?

```
val foo1 = run {  
    println("Calculating the answer...")  
    42  
}  
  
val foo2: Int  
    get() {  
        println("Calculating the answer...")  
        return 42  
    }  
  
fun main(args: Array<String>) {  
    println("$foo1 $foo1 $foo2 $foo2")  
}
```



The value is stored:

```
val foo1 = run {  
    println("Calculating the answer...")  
    42  
}
```

```
fun main(args: Array<String>) {  
    println("foo1:")  
    println("$foo1 $foo1")  
}
```

```
Calculating the answer...  
foo1:  
42 42
```

The value is calculated on each access:

```
val foo2: Int
    get() {
        println("Calculating the answer...")
        return 42
    }
```

```
fun main(args: Array<String>) {
    println("foo2:")
    println("$foo2 $foo2")
}
```

```
foo2:
Calculating the answer...
Calculating the answer...
42 42
```



How many times the phrase “Calculating the answer...” will be printed?

```
val foo1 = run {  
    println("Calculating the answer...")  
    42  
}  
  
val foo2: Int  
get() {  
    println("Calculating the answer...")  
    return 42  
}  
  
fun main(args: Array<String>) {  
    println("$foo1 $foo1 $foo2 $foo2")  
}
```

3



Fields

You can access `field` only inside accessors

```
class StateLogger {  
    var state = false  
    set(value) {  
        println("state has changed: " +  
                "$field -> $value")  
        field = value  
    }  
}
```

```
StateLogger().state = true
```

state has changed: false -> true

You always use property instead of getter or setter

```
class LengthCounter {  
    var counter: Int = 0  
  
    fun addWord(word: String) {  
        counter += word.length  
    }  
}
```

Inside the class the calls are optimized:

this.counter += ...

```
val lengthCounter = LengthCounter()  
lengthCounter.addWord("Hi!")  
println(lengthCounter.counter)
```

Getter is called under the hood:

lengthCounter.getCount();



Accessors visibility

Changing visibility of a setter

```
class LengthCounter {  
    var counter: Int = 0  
    private set  
  
    fun addWord(word: String) {  
        counter += word.length  
    }  
}
```