Kotlin, a Descendant of Java and Other Java Alternatives

Historical Development

Kotlin's Inception

- Released in early July of 2011
- Created by JetBrains
- Originally created because of the limitations of Java, Scala, and other Java alternatives
- Needed something that could coexist with there existing Java code

Google's Support

- Google announced it will support Kotlin on Android as a ‘first-class’ language
- JetBrains also made Android Studio for Google, so it only made sense to support the Kotlin project
- Kotlin can operate side by side with Android-Java code so switching is easy on the developer

Kotlin's Future

- Kotlin still receives regular updates and is also open source for the community to help build
- In 2021, over 4,800,000 developers used Kotlin for server-side, mobile multi-platform, Android, and front-end development
- Already estimated that 80% of Android apps are using Kotlin

Language Overview

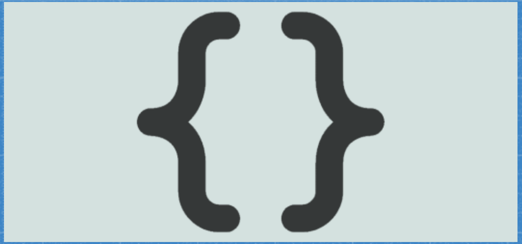
Language Overview

Kotlin

- Open-source
- Statically-typed
- Object-oriented and Functional
- 100% compatible with all existing Java code
- At least 20% less code compared to Java
- Null Safe
- Easy to learn

Language Features

General Syntax



- Curly Brace syntax
- Semicolons are optional at the end of a statement
- Package and import statements are the same as Java
- Main function is entry point of the whole program
- *fun* keyword and *class* keyword for functions and classes
- *open* keyword for inheritable classes
- *val* and *var* for constant and non-constant variables

Data Types

- Numerical types
- Logical types
- Char and String types
- Array types
- Unsigned numerical types and numerical array types
- Type checking and type casting with *is* and *as*

Note: A full list of data types can be found in Appendix B, Table 1

Primitive Operations

- 33 primary operators
- Normal mathematical operators
- Augmented assignment operators
- Increment/decrement operators
- Logical and Relational operators
- Null-based operators

Note: A full list of primary operators can be found in Appendix B, Table 2

Sequence Control

- Standard *if* statement
    - *if* statement can also take the form of an expression
- *when* statement, equivalent to C’s switch
- *for*, *while*, and *do-while* loops
- *break* and *continue* for loops
    - Labeled loops
- *return* statement for functions

Note: Examples of all the above can be found in Appendix A

Programming Environment

- IntelliJ
- Android Studio

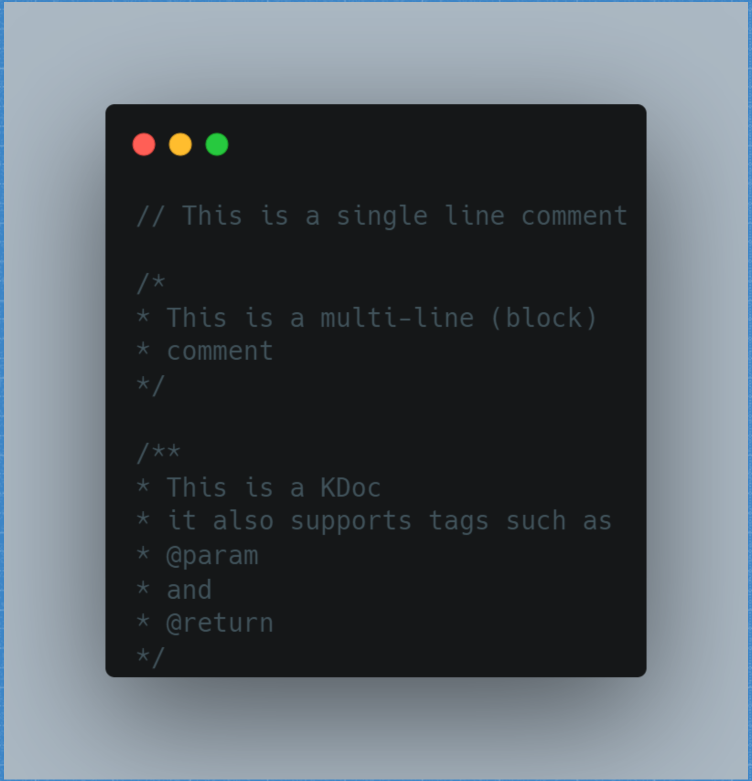
Kotlin Evaluation

Kotlin Evaluation

- Less code leading to increased readability
- Statically typed for better readability
- Small set of keywords
- Simple syntax
- Shorthand notations
- Type checking
- Null operators and null safe design

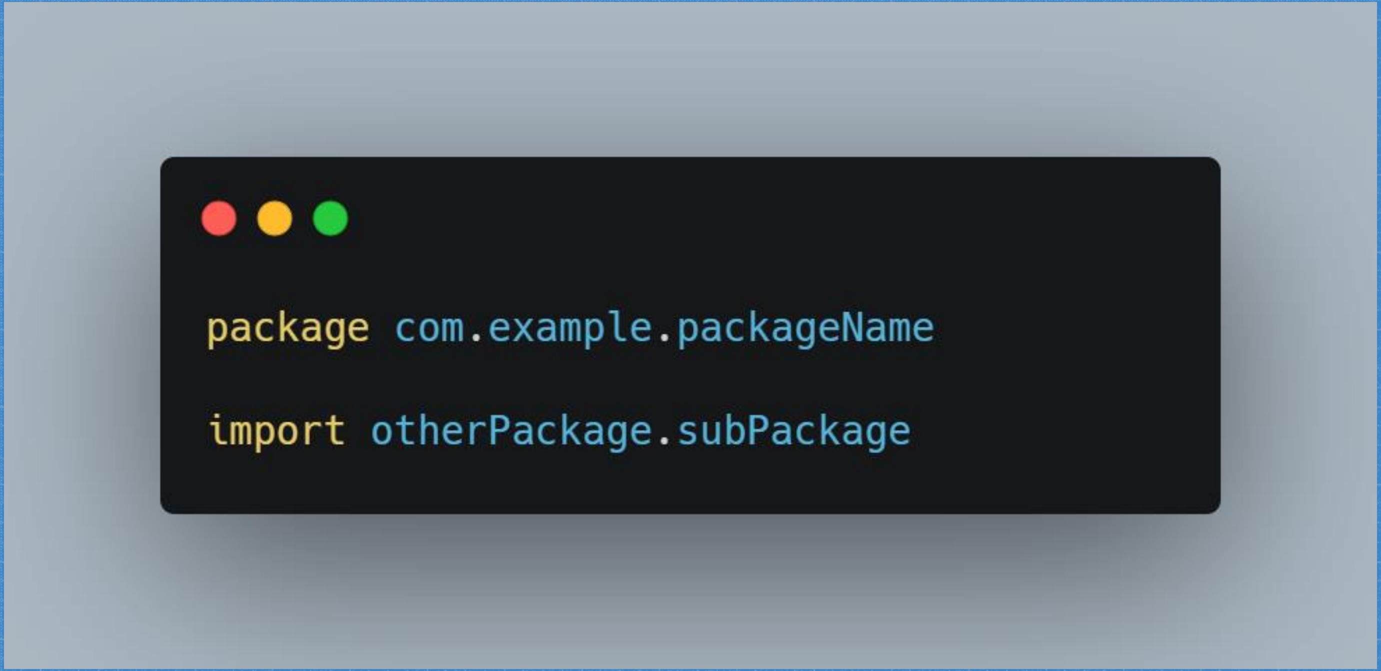
Appendix A

Example Programs



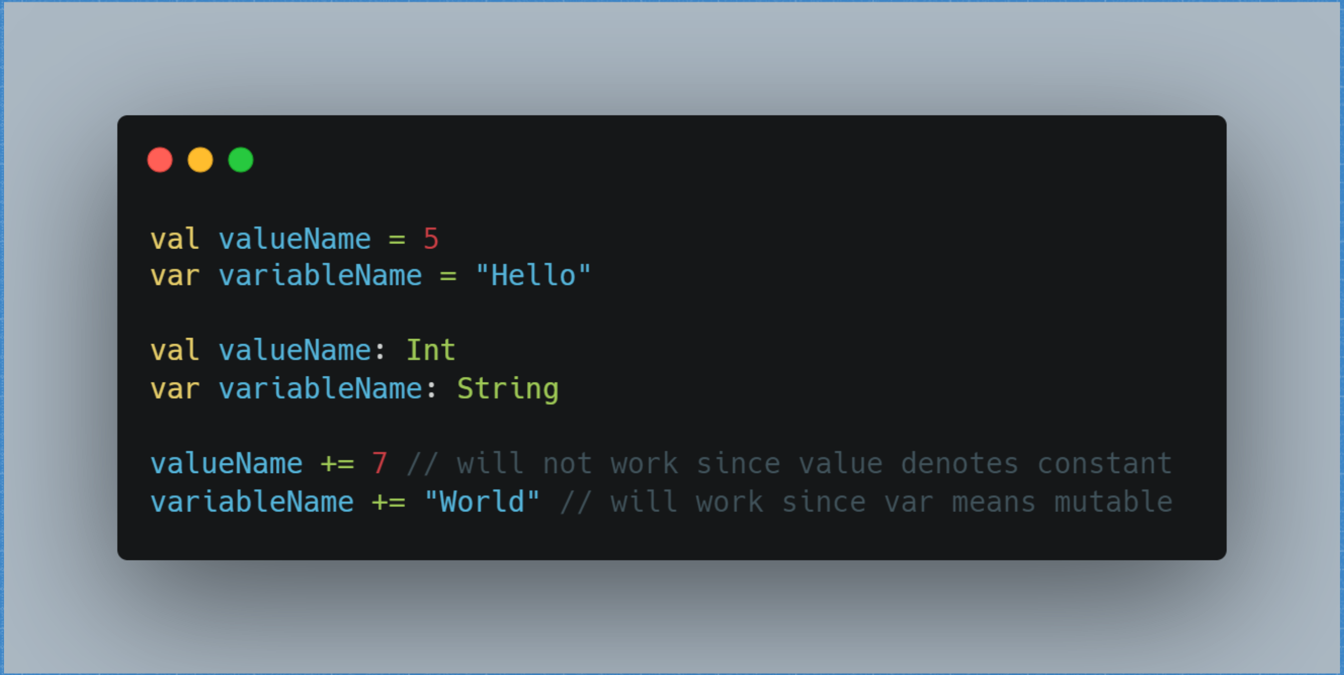
```
// This is a single line comment

/*
* This is a multi-line (block)
* comment
*/

/**
* This is a KDoc
* it also supports tags such as
* @param
* and
* @return
*/
```

Figure 1. An example of all types of comments.



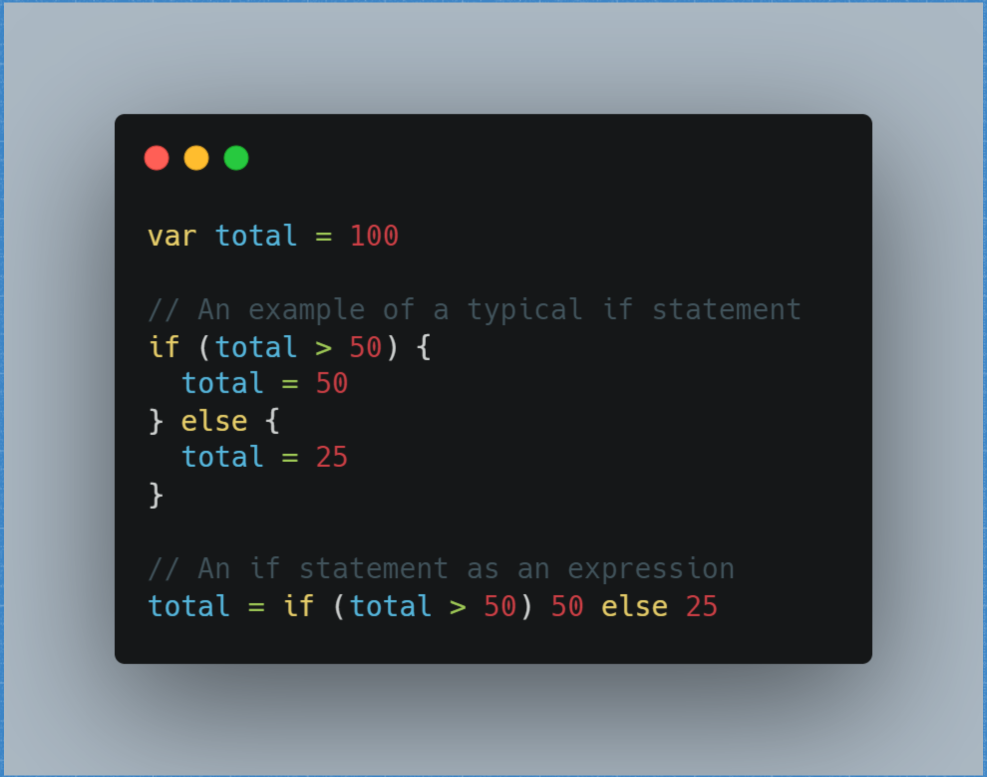
```
package com.example.packageName

import otherPackage.subPackage
```

Figure 2. An example of declaring the package a file is in, and importing another package



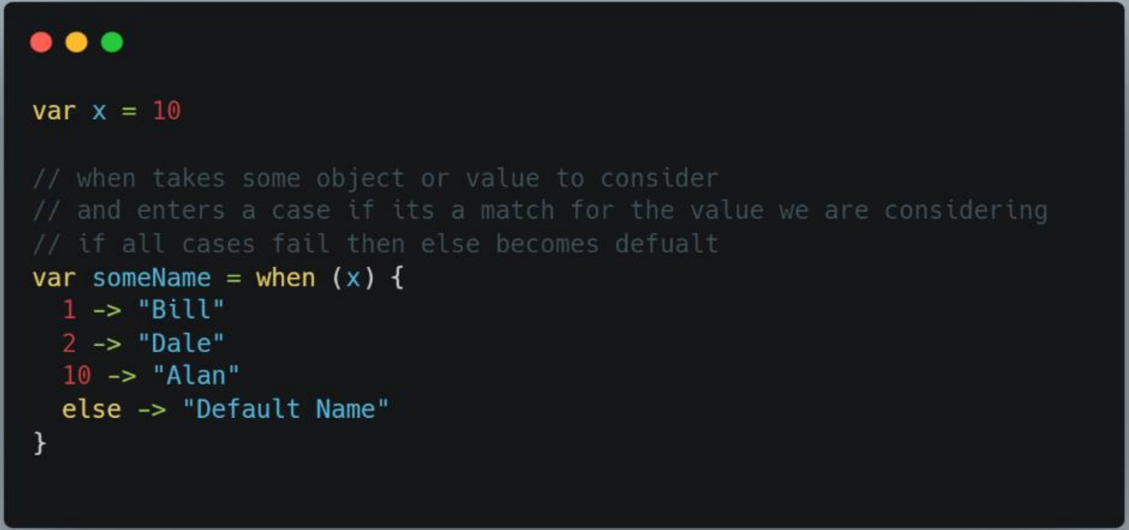
```
val valueName = 5
var variableName = "Hello"

val valueName: Int
var variableName: String

valueName += 7 // will not work since value denotes constant
variableName += "World" // will work since var means mutable
```

Figure 3. An example of *val* and *var* for variable declaration.



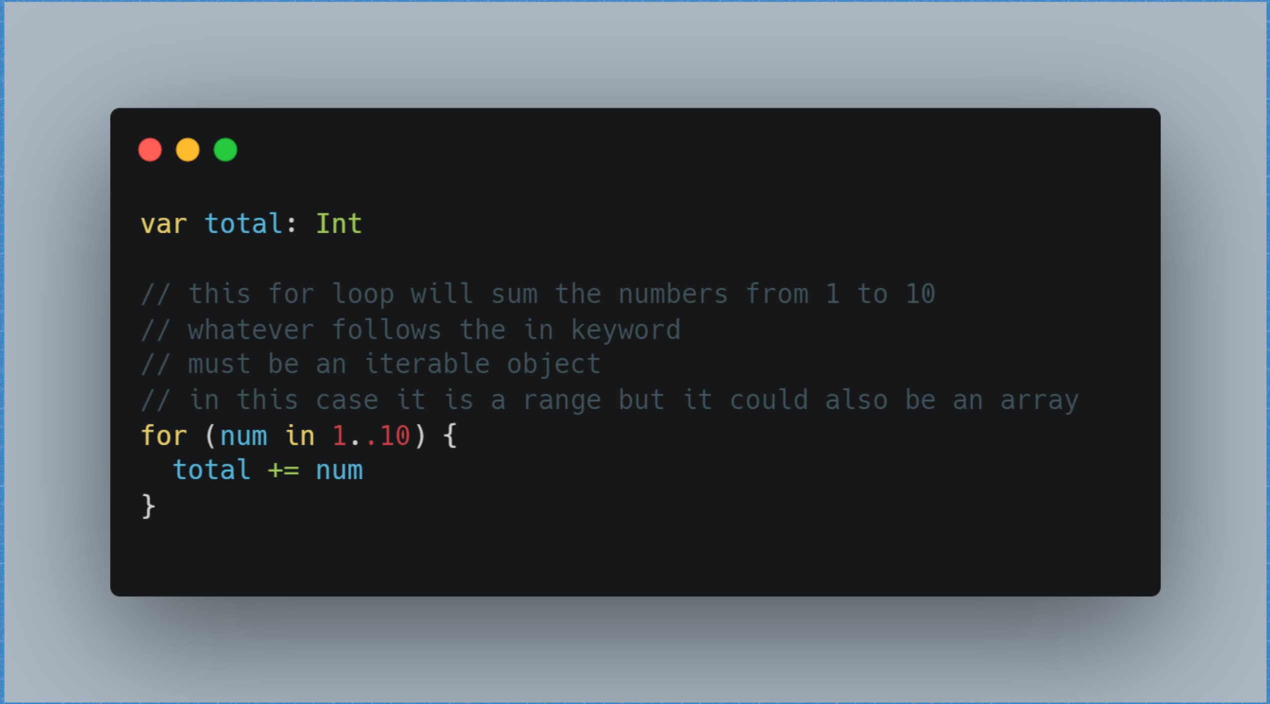
```
var total = 100

// An example of a typical if statement
if (total > 50) {
  total = 50
} else {
  total = 25
}

// An if statement as an expression
total = if (total > 50) 50 else 25
```

Figure 4. An example of both forms of the *if* statement.



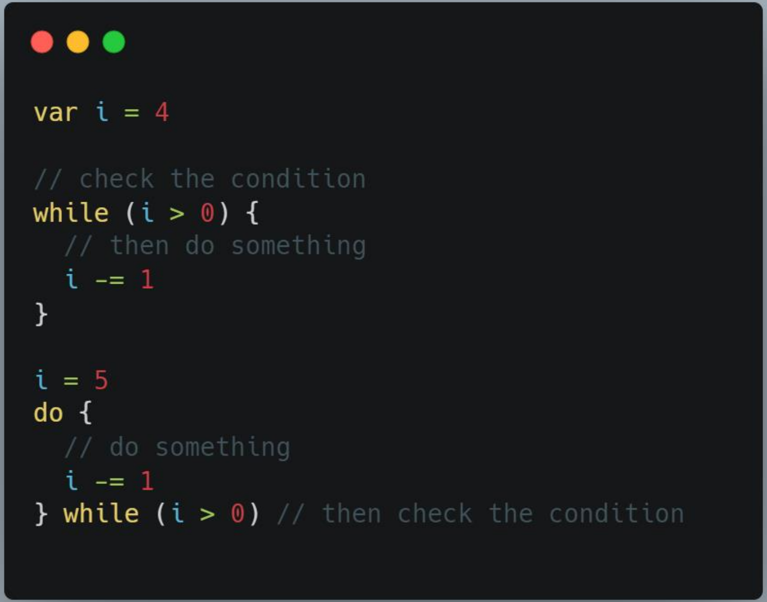
```
var x = 10

// when takes some object or value to consider
// and enters a case if its a match for the value we are considering
// if all cases fail then else becomes defualt
var someName = when (x) {
  1 -> "Bill"
  2 -> "Dale"
  10 -> "Alan"
  else -> "Default Name"
}
```

Figure 5. An example of the *when* statement.



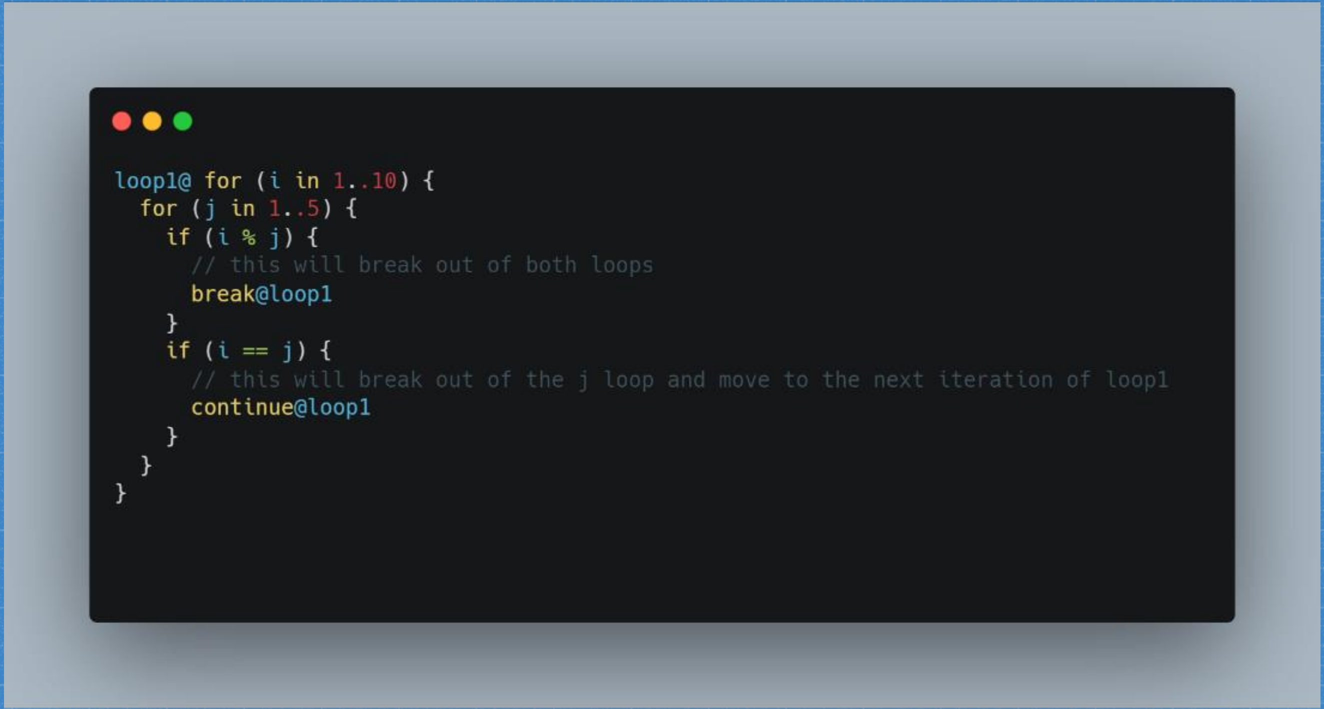
```
var total: Int

// this for loop will sum the numbers from 1 to 10
// whatever follows the in keyword
// must be an iterable object
// in this case it is a range but it could also be an array
for (num in 1..10) {
  total += num
}
```

Figure 6. An example of a *for* loop.



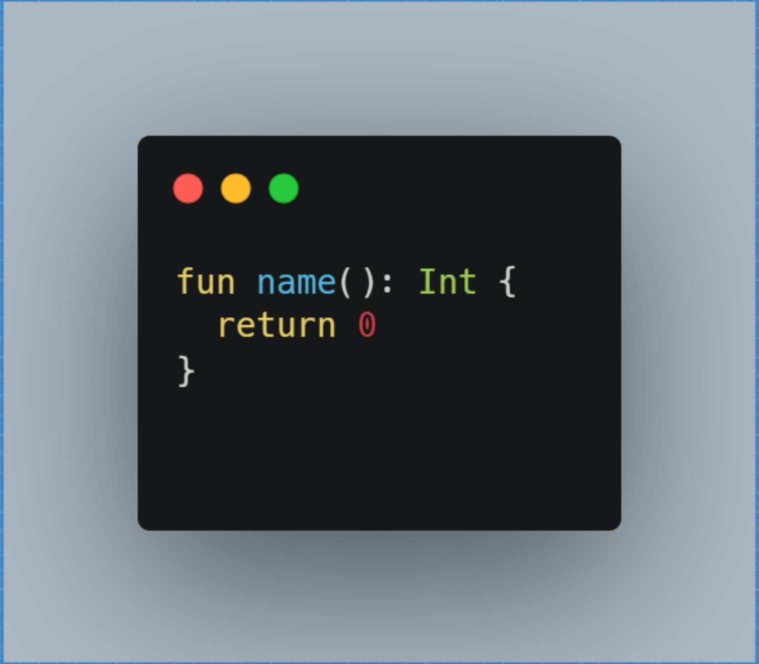
```
var i = 4

// check the condition
while (i > 0) {
  // then do something
  i -= 1
}

i = 5
do {
  // do something
  i -= 1
} while (i > 0) // then check the condition
```

Figure 7. An example of a *while* loop, and a *do-while* loop.



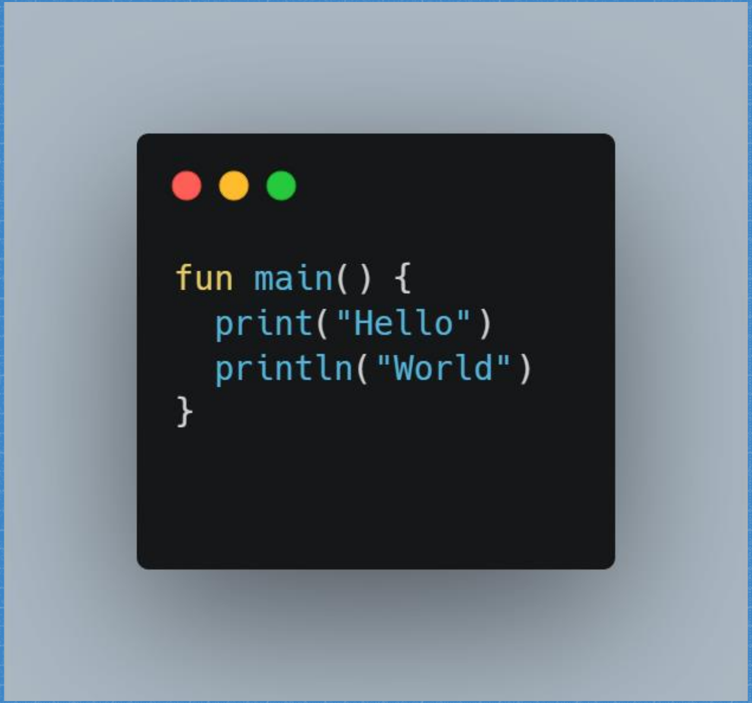
```
loop1@ for (i in 1..10) {
  for (j in 1..5) {
    if (i % j) {
      // this will break out of both loops
      break@loop1
    }
    if (i == j) {
      // this will break out of the j loop and move to the next iteration of loop1
      continue@loop1
    }
  }
}
```

Figure 8. An example of the *break* and *continue* statements. It also shows labeled loops.



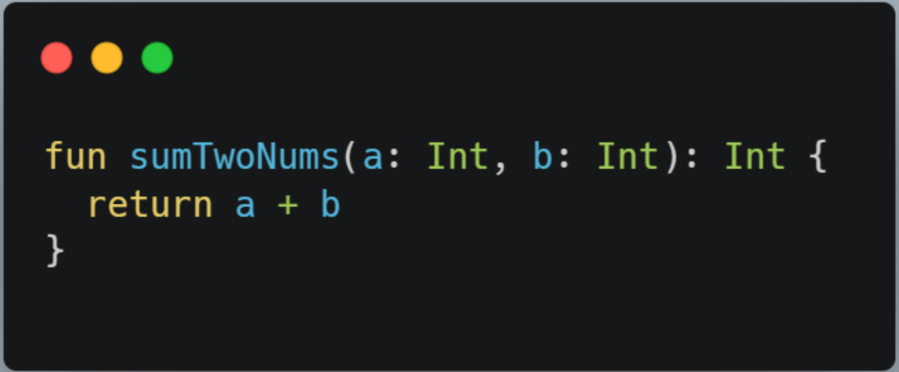
```
fun name(): Int {
  return 0
}
```

Figure 9. A simple *return* statement.



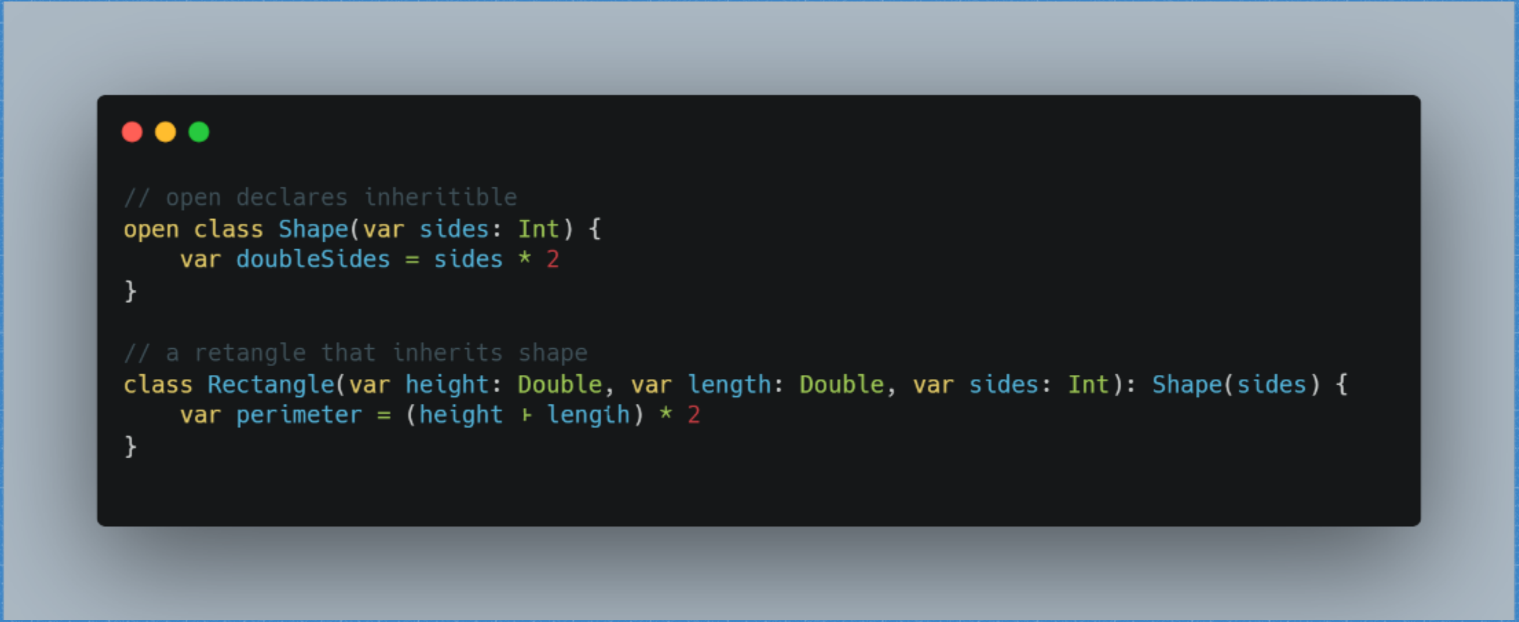
```
fun main() {
  print("Hello")
  println("World")
}
```

Figure 10. The syntax of the main function. It also shows the print statement syntax.



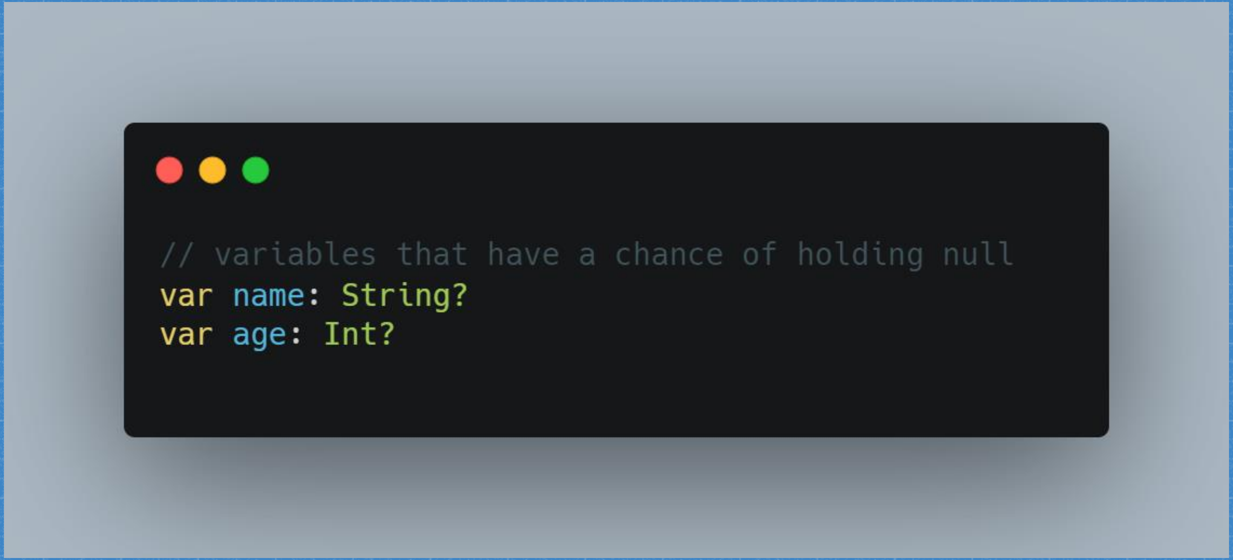
```
fun sumTwoNums(a: Int, b: Int): Int {
  return a + b
}
```

Figure 11. A simple summation function.



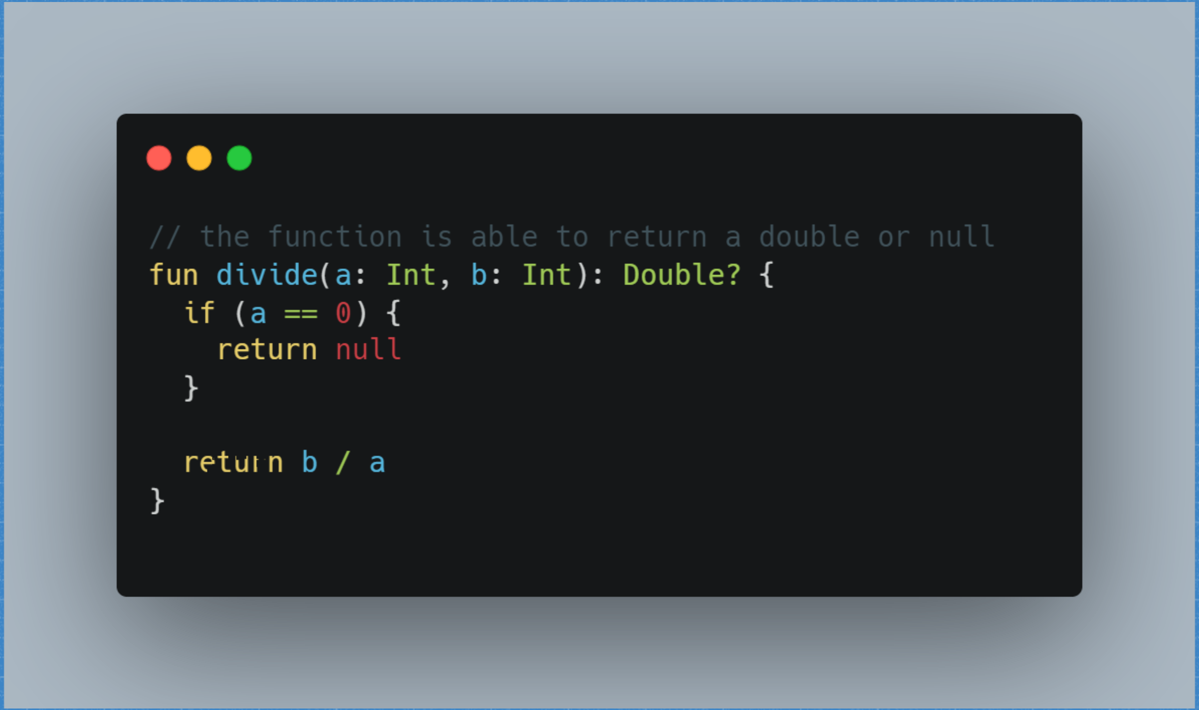
```
// open declares inheritible
open class Shape(var sides: Int) {
    var doubleSides = sides * 2
}

// a retangle that inherits shape
class Rectangle(var height: Double, var length: Double, var sides: Int): Shape(sides) {
    var perimeter = (height + length) * 2
}
```

Figure 12. An example of classes and inheritance.



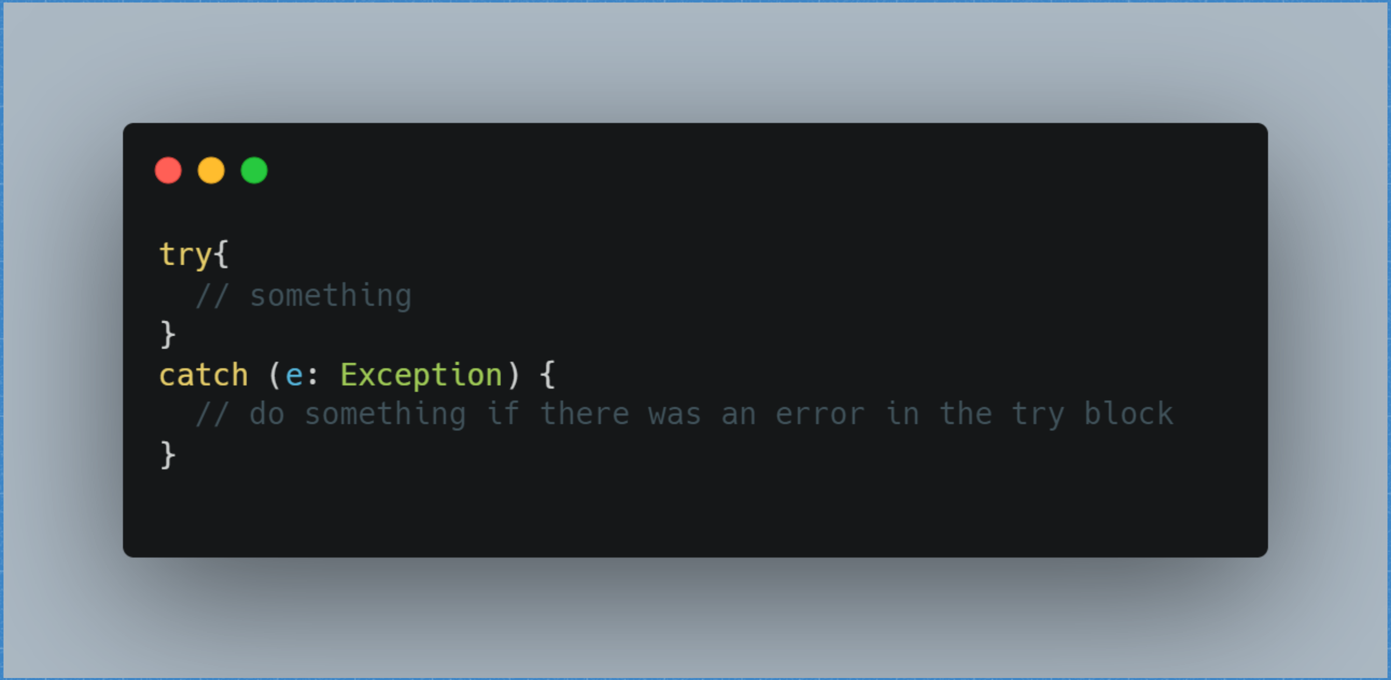
```
// variables that have a chance of holding null
var name: String?
var age: Int?
```

Figure 13. An example of variables that might hold null.



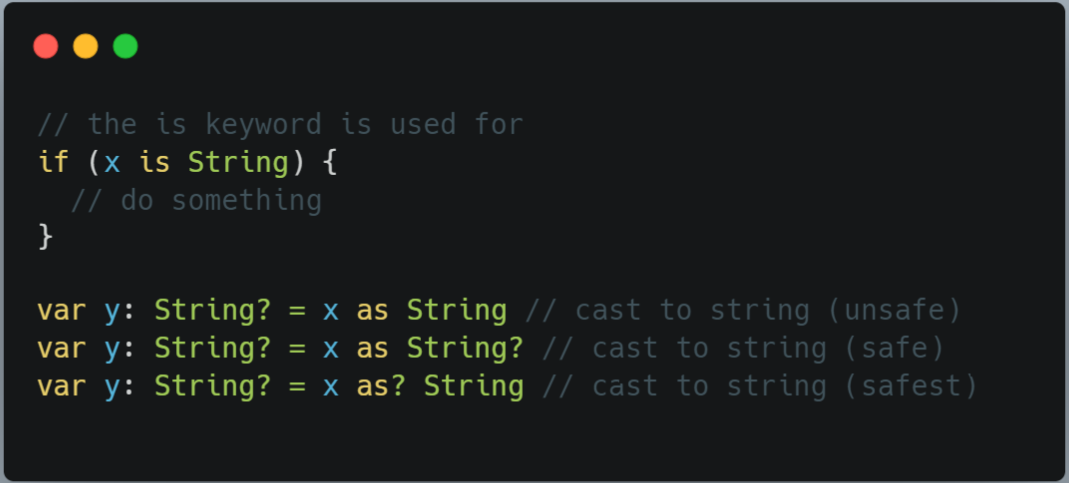
```
// the function is able to return a double or null
fun divide(a: Int, b: Int): Double? {
  if (a == 0) {
    return null
  }

  return b / a
}
```

Figure 14. An example of a function that can return a double or null.



```
try{
  // something
}
catch (e: Exception) {
  // do something if there was an error in the try block
}
```

Figure 15. An example of exception handling.



```
// the is keyword is used for
if (x is String) {
  // do something
}

var y: String? = x as String // cast to string (unsafe)
var y: String? = x as String? // cast to string (safe)
var y: String? = x as? String // cast to string (safest)
```

Figure 16. An example of type checking, and typecasting.

Appendix B

Types & Operators

Numerical	Logical	Characters / Strings	Arrays	Unsigned
Byte	Boolean	Char	ByteArray	UByte
Short		String	ShortArray	UShort
Int			IntArray	UInt
Long			LongArray	ULong
Float			FloatArray	
Double			DoubleArray	UByteArray
			CharArray	UShortArray
				UIntArray
				ULongArray

Table 1. All available types.

Operator	Purpose
+, -, *, /, %	Mathematical operators
=	Assignment operator
+=, -=, *=, /=, %=	Augmented assignment operators
++, --	Increment and decrement operators
&&, ||, !	Logical 'and', 'or', 'not' operators
==, !=	Equality operators
===, !==	Referential equality operators
<, >, <=, >=	Comparison operators
!!	Assert that an expression is not null
?.	Performs a safe call
?:	Takes right-hand if the left-hand value is null (Elvis operator)
::	Method reference operator
..	Range operator
:	Separates variable name and type
?	Marks a type as possibly null
;	Separates multiple statements on the same line
$	References a variable in a string template

Table 2. All primary operators.

Works Cited

Works Cited

Android Developers. “Kotlin Overview.” *Android Developers*, Google Developers, 27 Dec. 2019, https://developer.android.com/kotlin/overview.

Bogode, Stanley, et al. “Discover the History of Kotlin.” *OpenClassrooms*, OpenClassrooms, 28 June 2022, https://openclassrooms.com/en/courses/5774406-learn-kotlin/5930526-discover-the-history-of-kotlin.

Garg, Priyanka. “What Is Kotlin? 12 Interesting Facts about Kotlin.” *OpenXcell*, OpenXcell, 20 Dec. 2021, https://www.openxcell.com/blog/12-things-must-know-kotlin/.

Kotlin Foundation. “Kotlin Programming Language.” *Kotlin*, JetBrains, https://kotlinlang.org.

Miller, Paul. “Google Is Adding Kotlin as an Official Programming Language for Android Development.” *The Verge*, Vox Media, 17 May 2017, https://www.theverge.com/2017/5/17/15654988/google-jet-brains-kotlin-programming-language-android-development-io-2017.

Naik, Amit Raja. “Ten Years of Kotlin Programming Language.” *Analytics India Magazine*, AIM, 13 Oct. 2021, https://analyticsindiamag.com/ten-years-of-kotlin-programming-language/.

Zubchenko, Alexander. “The Complete Kotlin Programming Language Review - Software Development.” *Waverley*, Waverley Software Inc, 7 Oct. 2022, https://waverleysoftware.com/blog/kotlin-review/.