

Kotlin for Beginners

WIT - NYC

April Selby - April 12th 2023

About me: April Selby

- Senior Engineering Manager in Finance
- Previously worked in DX and MARs
- UK -> NYC in 2020
- Currently in a Kotlin based role
- Mainly worked in Golang and Java



About Kotlin

- First appeared in July 2011
- Developed by JetBrains
- In 2019 Google announced it was the preferred language for Android development
- Has both object oriented and functional constructs
- Much more concise than Java



Kotlin Concepts

- Data classes
- Null safety
- Smart casting
- Extension functions



Kotlin

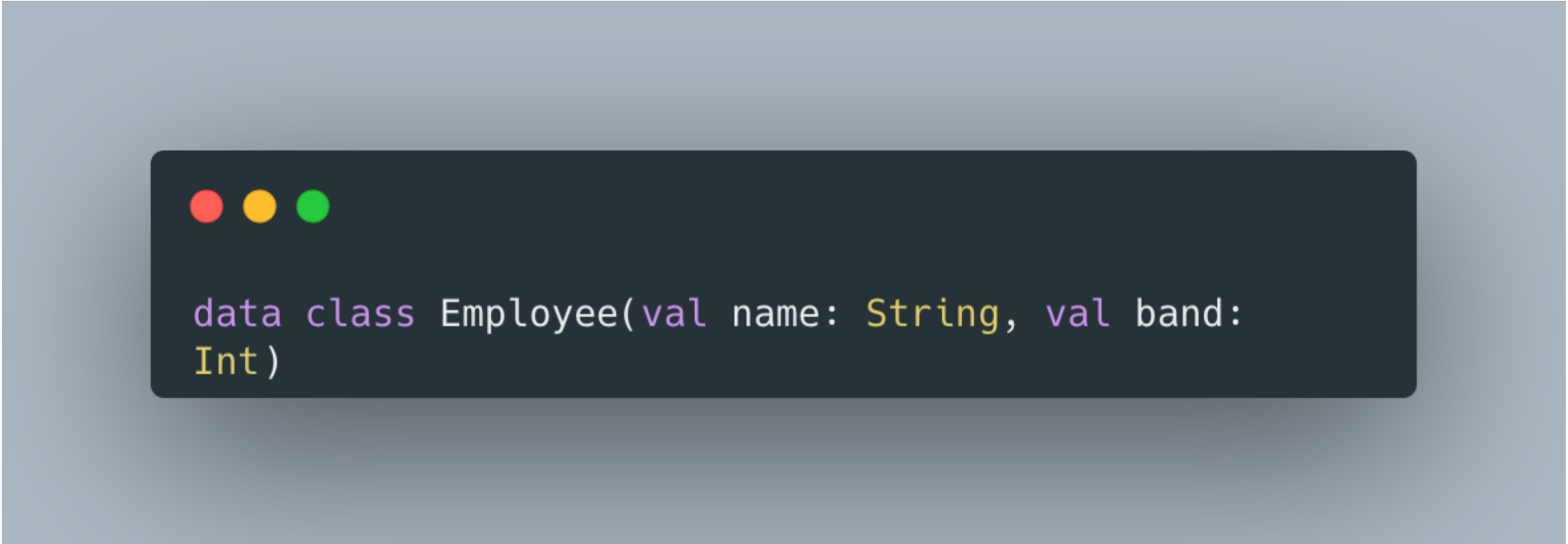
Coding Exercise: Library

- Clone from: <https://github.com/aselby88/library.git>
- Library that has:
 - Books



Data Classes

- Compiler provides:
 - toString()
 - equals() / hashCode()
 - copy()
- Val vs var
- To see the Java code:
 - In IntelliJ click Tools -> Kotlin -> Show Kotlin Bytecode, click Decompile

A screenshot of a code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The code is written in Kotlin and defines a data class named 'Employee' with two properties: 'name' of type 'String' and 'band' of type 'Int'.

```
data class Employee(val name: String, val band: Int)
```

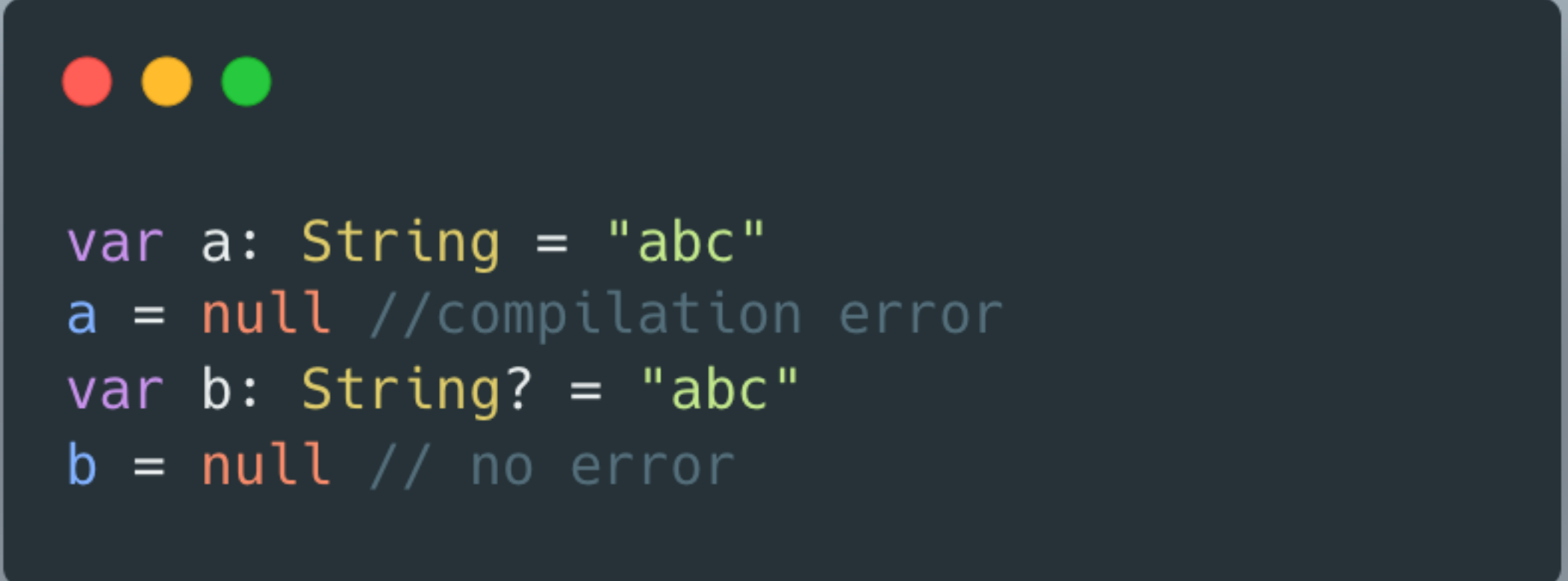
Create a User class

- Create a user with:
 - Name (String)
 - ArrayList of books currently borrowing
 - ArrayList of borrowed books
- Then inside of the main class create yourself as a user and print yourself
- Hint:
 - Create a Kotlin file
 - `arrayListOf()`



Null Safety

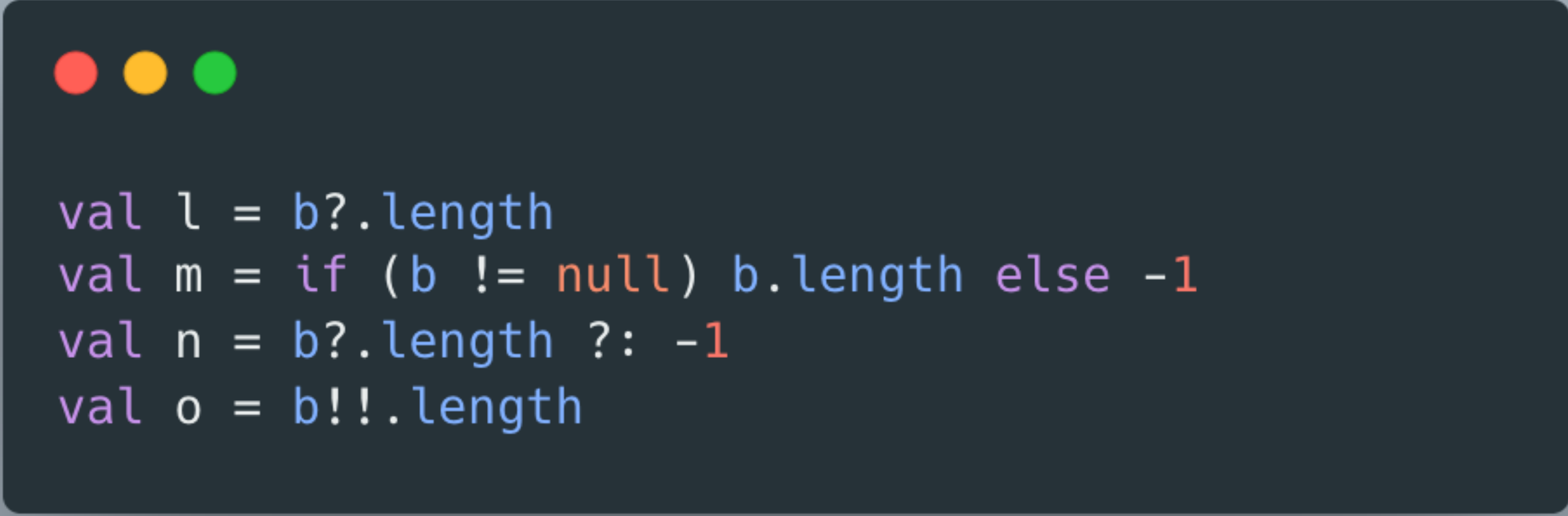
- Kotlin's type system is aimed at eliminating the danger of null references



```
var a: String = "abc"  
a = null //compilation error  
var b: String? = "abc"  
b = null // no error
```


Null Safety

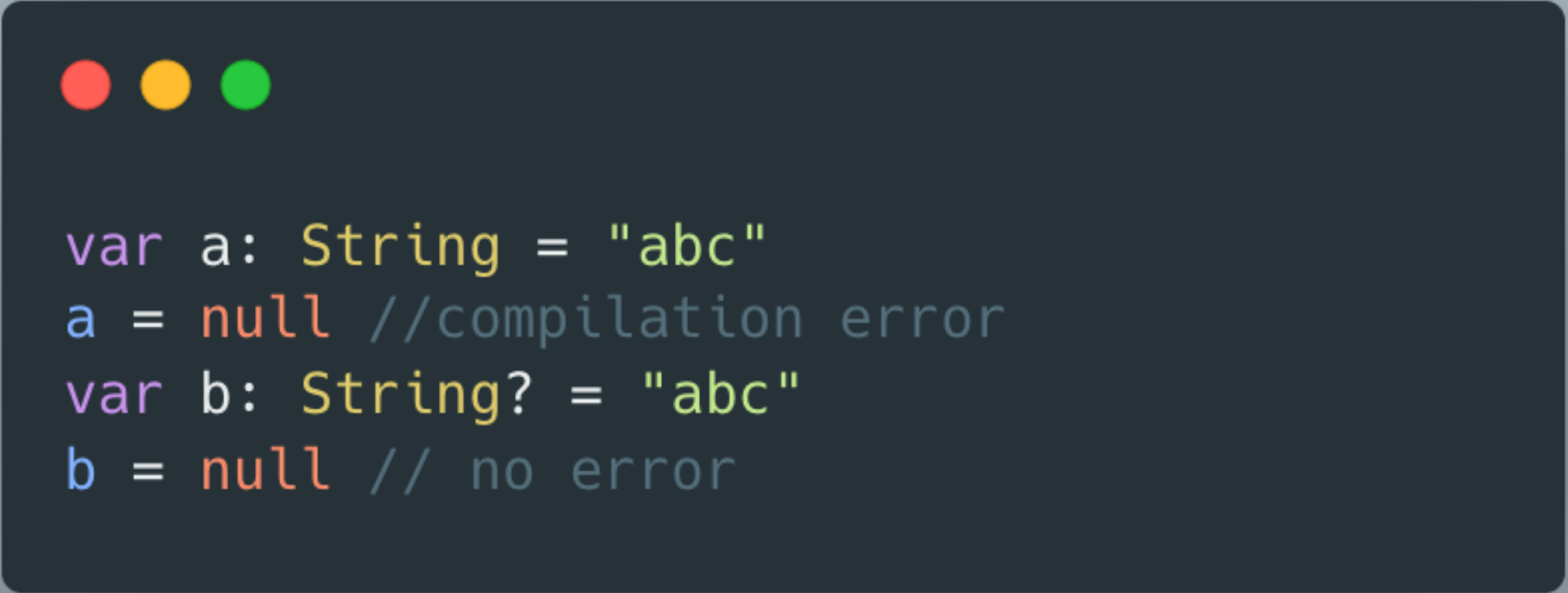
- Safe calls
- Elvis operator
- The !! Operator



```
val l = b?.length  
val m = if (b != null) b.length else -1  
val n = b?.length ?: -1  
val o = b!!.length
```

Add a rating to the Book class

- Add an string rating to the book class that can be:
 - Null
 - A value



```
var a: String = "abc"  
a = null // compilation error  
var b: String? = "abc"  
b = null // no error
```



Smart Casting

- Smart cast is a feature in which the Kotlin compiler tracks conditions inside of expressions
- Tracks object type and whether it can be null
- Allows you to call function without explicit casting

```
val obj: Any = "abc"
if(obj is String) {
    // No Explicit Casting needed.
    println("String length is ${obj.length}")
}
```

```
Object object = "abc";
String myString;
if (object instanceof String) {
    myString = (String) object;
    System.out.println("String length is " + myString.length)
}
```

Create a print function

- Create a print function that takes Any as an argument
- Should print the title if a book
- Should print the name if a user
- Hint:
 - Kotlin has an is operator



```
fun printObject(x: Any) {  
  
}
```

Extension functions

- Kotlin provides the ability to extend a class or an interface with new functionality without having to inherit from the class



```
fun String.removeFirstLastChar(): String = this.substring(1, this.length - 1)

fun main(args: Array<String>) {
    val myString = "Hello Everyone"
    val result = myString.removeFirstLastChar() // ello Everyon
}
```


Extension functions - Caution



- Should not be for implementing business logic
- Extensions are resolved statically
 - Are not associated with an object instance
 - Resulting code is tightly coupled to the extension method implementation
 - Hard to test (makes mocking difficult)
- Good for utility functions, code you can't change, and keeping data objects clean

Book class: Return Unrated

- Add an extension function to the book class that returning the rating
- Challenge: Returns the string “Unrated” if the rating is null

```
fun String.removeFirstLastChar(): String = this.substring(1, this.length - 1)

fun main(args: Array<String>) {
    val myString = "Hello Everyone"
    val result = myString.removeFirstLastChar() // ello Everyon
}
```

```
val l = b?.length
val m = if (b != null) b.length else -1
val n = b?.length ?: -1
val o = b!!.length
```

Next Steps

- Pluralsight:
 - <https://app.pluralsight.com/library/courses/kotlin-fundamentals/table-of-contents>
- Kotlin documentation:
 - <https://kotlinlang.org/docs/home.html>
- Kotlin Cheatsheet:
 - <https://devhints.io/kotlin>