

Singleton

&

Prototype

The force has
Kotlin Language
to help you in
this journey.

```
object BooksRepository {
   private val books = mutableListOf<Book>()

fun saveOrUpdate(book: Book) {
    val existingBookIndex = books.indexOfFirst { it:Book
        it.id = book.id
    }
    books.removeAt(existingBookIndex)
    books.add(book)
}

fun all(): List<Book> = books

fun getBookById(id: UUID) = books.first { it.id = id }
}
```



```
data class Book(
   val id: UUID,
   val numberOfPages: Int,
   val title: String,
   val author: String,
```

Singleton, Kotlin has

```
BooksRepository {
    private val books = mutableListOf<Book>()

fun saveOrUpdate(book: Book) {
    val existingBookIndex = books.indexOfFirst { it:Book
        it.id = book.id
    }
    books.removeAt(existingBookIndex)
    books.add(book)
}

fun all(): List<Book> = books

fun getBookById(id: UUID) = books.first { it.id = id }

}
```



Practice we need.

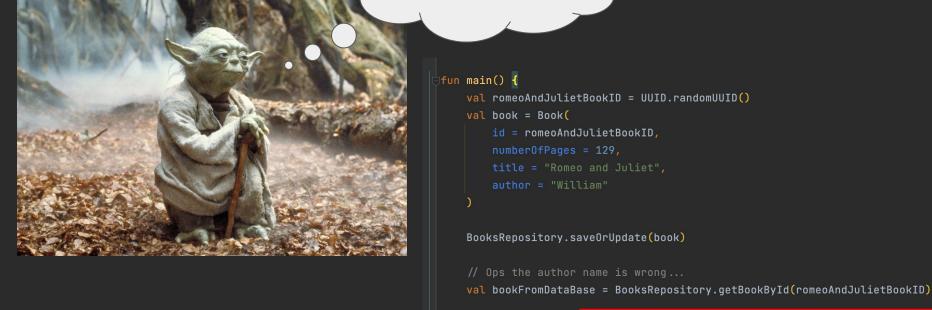


```
fun main() {
    val romeoAndJulietBookID = UUID.randomUUID()
    val book = Book(
        id = romeoAndJulietBookID,
    BooksRepository.saveOrUpdate(book)
    val bookFromDataBase = BooksRepository.getBookById(romeoAndJulietBookID)
    val updatedBook = bookFromDataBase.copy(author = "William Shakespeare")
    BooksRepository.saveOrUpdate(updatedBook)
```

Prototype, Kotlin has

val updatedBook = bookFromDataBase.copy(author = "William Shakespeare")

BooksRepository.saveOrUpdate(updatedBook)



Prototype, Kotlin has

```
fun main()
             data <mark>class Book(</mark>
   val ror
   val boo
                   val id: UUID,
                   val numberOfPages: Int,
                   val title: String,
                   val author: String,
   BooksR
   val bookFromDataBase = BooksRepository.getBookById(romeoAndJulietBookID)
   val updatedBook = bookFromDataBase.copy(author = "William Shakespeare")
   BooksRepository.saveOrUpdate(updatedBook)
```

