Refactoring Legacy Code with Kotlin

TCM Extended: Berlin 🇩🇪

@askashdavies

Legacy

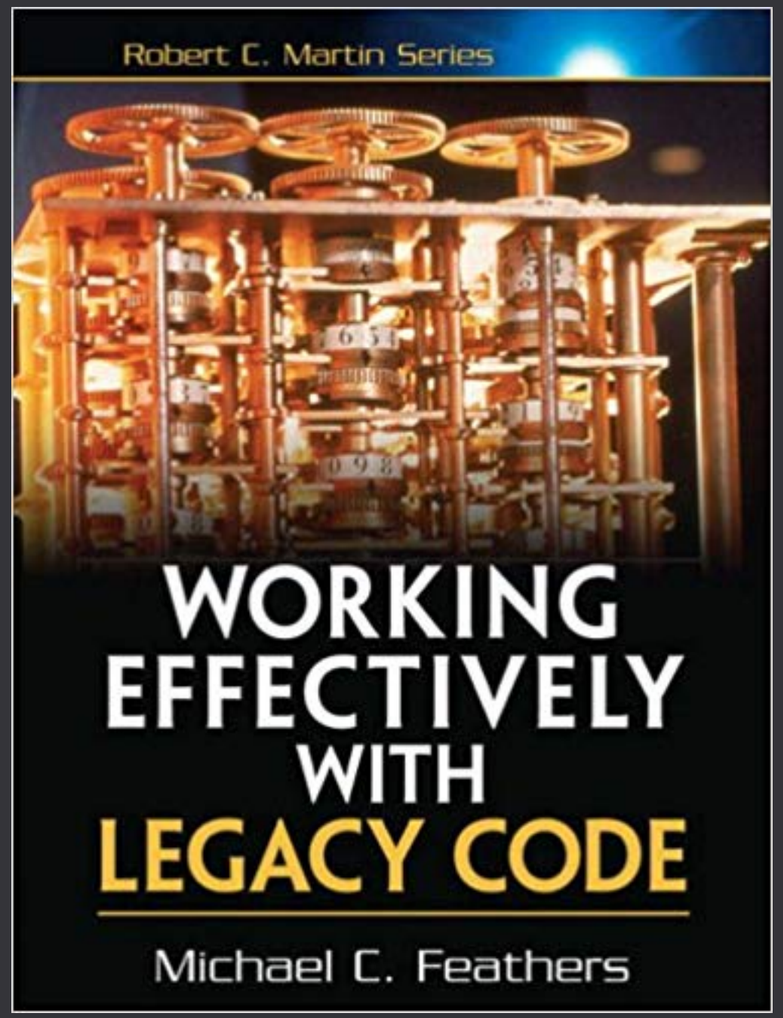
adj.

Denoting or relating to software or hardware that has been superseded but is difficult to replace because of its wide use.

Working Effectively with Legacy Code

Michael Feathers

- Untested code
- Regression tests
- Lack of confidence



Java 

JVM != ART

Corey Quinn
@QuinnyPig

Follow

WHO DID THIS

#oow19

List of largest law firms by revenue

From Wikipedia, the free encyclopedia

This is a **list of the world's largest law firms by revenue** in 2017.[1][2]

Rank	Firm	Revenue (US$)	Lawyers	Revenue per lawyer (US$)	Country
1	Oracle	$37,730,000,000	3,000	$12,570,000	United States
2	Latham & Watkins	$3,063,992,000	2,436	$1,258,000	United States
3	Baker McKenzie (verein)	$2,900,000,000	4,723	$614,000	United States
4	DLA Piper (verein)	$2,634,094,000	3,609	$730,000	United Kingdom United States
5	Skadden	$2,582,325,000	1,784	$1,447,000	United States
6	Dentons (verein)	$2,360,000,000	8,658	$273,000	China United States Canada United Kingdom

2:24 pm - 18 Sep 2019

935 Retweets **3,085** Likes

Oracle

46 935 3.1K

twitter.com/QuinnyPig/status/1174434046853599232

Java 8

- Lambda expressions
- Method references
- Default interfaces
- try-with-resources

Streams ↝


SO?
@askashdavies

Kotlin 

Null  

```
if (foo == null) {
  bar();
}
```

Defensive Code <  >

Offensive Code < 🖕 >

Urgency 

Kotlin

noun

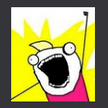
Freakin' awesome

Google IO 2018 

@askashdavies

Google IO 2019 

@askashdavies

Kotlin 

Libraries

Anko

RxKotlin

Kotlin Extensions for RxJava

RxDownload

kotlinx.coroutines

JB official | Download 0.10.8 | build passing | license Apache License 2.0

Ktor

Anko

Download 0.8.8 | build passing | issue resolution 9 d | Android Arsenal KAndroid | AndroidWeekly #148

RxKotlin

KAndroid

JB official | latest version v1.3.50

Kotlin Extensions for RxJava

Kotlin/Native

Kotter Knife

RxDownload

language kotlin | RxJava 2.0

kotlinx.coroutines

JB official | license Apache License 2.0 | Download 1.3.0

kotlin-core

This p
take a

JB official | Download 0.10.8 | build passing | license Apache License 2.0

Gradle Kotlin DSL Samples

build passing | licen

Ktor

Anko

Download 0.8.8 | build passing | issue resolution 9 d | Android Arsenal KAndroid | AndroidWeekly #148

RxKotlin

KAndroid

JB official | latest version v1.3.50

Kotlin Extensions for RxJava

Kotter Knife

Kotlin/Native

kotlin 1.0.2 | maven-central v2.0.0-ALPHA-03 | build passing | issues 13 open | license MIT | chat kotlin slack

Notice: Kodein and Injekt, much of the same

RxDownload

language kotlin | RxJava 2.0

kotlinx.coroutines

JB official | license Apache License 2.0 | Download 1.3.0

kotlin 1.3.10 | maven-central v2.6.0 | build passing | issues 1 open | license MIT | chat kotlin slack #kohesive

klutter


kotlin-core
Gradle Kotlin DSL Samples
build passing
This p
take a
JB official Download 0.10.8 build passing license Apache License 2.0
Ktor
Ankc
KotlinTest
Download 0.8.8 build passing issue resolution
KAndroid
JB official latest version v1.3.50
Kotlin Extensions for RxJava
Kotter Knife
Kotlin/Native
kotlin 1.0.2 maven-central v2.0.0-ALPHA-03 build passing issues 13 open license MIT chat kotlin slack
Notice:
Download 2.3.0 Android Arsenal LastAdapter License Apache 2.0 chat on gitter
RxDownloac
LastAdapter
Kanary
languag
powered by vok build passing gitter join chat tag v0.7.1 maven central 0.7.1
Welcome to Vaadin-On-Kotlin
kotlin 1.3.10 maven-central v2.6.0 build passing issues 1 open license MIT chat kotlin slack #kotlinslack
KANARY
klutter


build
passing

Kotlin 🤔

Idiomacy


@askashdavies

Idiomatic

adj.

Using, containing, or denoting expressions that are natural to a native speaker

Idiomatic use of Language Features

kotlinlang.org/docs/reference/coding-conventions.html#idiomatic-use-of-language-features

Idiomatic Code

- Consistent, easier to read
- Less cognitive load
- Less ambiguity
- Function > Style

Code Style

- kotlinlang.org/docs/reference/coding-conventions.html
- android.github.io/kotlin-guides/style.html

Refactoring Legacy Code

@askashdavies


^⌥⇧K


US FIGURE
SKATING


Please don't

```
public class BadJavaActivity extends Activity {

  @Inject Dependency dependency;
}
```

General Assumptions 👮

How I Met Your Mother, CBS

```
class BadKotlinActivity : Activity() {

  @Inject var dependency: Dependency? = null
}
```

```
class SlightlyLessBadKotlinActivity : Activity() {

  @Inject internal lateinit var dependency: Dependency
}
```

UninitializedPropertyAccessException!

```
lateinit var file: File

if (::file.isInitialized) { /* ... */ }
```

Nullability  

!!

```
// Throws NullPointerException 🔥
myNullVal.myMethod()
```

```
// Still throws NullPointerException 🔥
myNullVal!!.myMethod()
```

```
// Does nothing 🤷
myNullVal?.myMethod()
```

```
// Still does nothing 🤷
myNullVal?.also {
   it.myMethod()
}
```

```
// Defensive Code <🛡>
myNullVal?.also {
   it.myMethod()
}
```

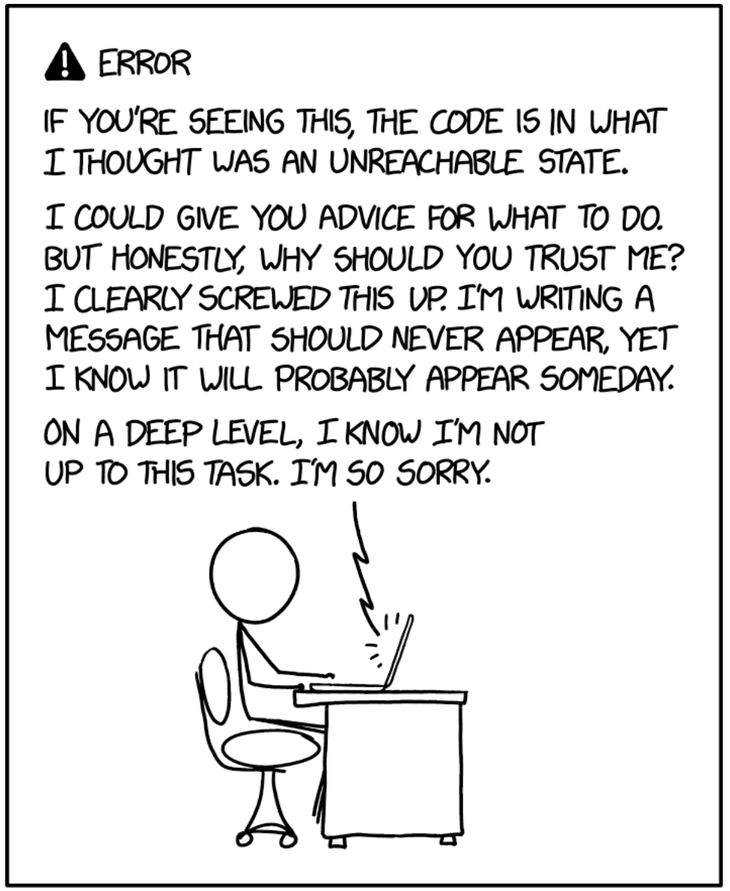
```
// Offensive Code <🖕>
myNullVal!!.myMethod()
```

```
// throws IllegalArgumentException("Required value was null.")
requireNotNull(myNullVal).myMethod()
```

```
// throws IllegalArgumentException("myNullVaL was null because ...")
requireNotNull(myNullVal) { "myNullVaL was null because ..." }.myMethod()
```

```
class ImpossibleException : IllegalArgumentException()

// throws ImpossibleException()
val myNotNullVal = myNullVall ?: throw ImpossibleException()

myNotNullVal.myMethod()
```

```
class ImpossibleException : IllegalArgumentException("""
    If you're seeing this, the code is in what I thought was an unreachable state.
    I could give you advice for what to do, but honestly, why should you trust me?
    I clearly screwed this up. I'm writing a message that should never appear,
    yet I know it will probably appear someday.

    On a deep level, I know I'm not up to this task.
    I'm so sorry.
""")

// throws ImpossibleException()
val myNotNullVal = myNullVall ?: throw ImpossibleException()

myNotNullVal.myMethod()
```



NEVER WRITE ERROR MESSAGES TIRED.

xkcd.com/2200/

Data Classes

```java
public class User {

  private String firstName;
  private String lastName;

  public User(String firstName, String lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
  }

  public String getFirstName() {
    return firstName;
  }

  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }

  public String getLastName() {
    return lastName;
  }

  public void setLastName(String lastName) {
    this.lastName = lastName;
  }

  @Override
  public boolean equals(Object o) {
    if (this == o) {
      return true;
    }
    if (o == null || getClass() != o.getClass()) {
      return false;
    }
    User user = (User) o;
    return Objects.equals(firstName, user.firstName) &&
        Objects.equals(lastName, user.lastName);
  }

  @Override
  public int hashCode() {
    return Objects.hash(firstName, lastName);
  }
}
```

```
class User(var firstName: String?, var lastName: String?) {

  override fun equals(o: Any?): Boolean {
    if (this === o) {
      return true
    }
    if (o == null || javaClass != o.javaClass) {
      return false
    }
    val user = o as User?
    return firstName == user!!.firstName && lastName == user.lastName
  }

  override fun hashCode(): Int {
    return Objects.hash(firstName, lastName)
  }
}
```

```
data class User(
  val firstName: String,
  val lastName: String
)
```

```
data class User(
  val firstName: String,
  val lastName: String? = null
)
```

```
@NotNull
public final User copy(@Nullable String firstName, @Nullable String lastName) {
  return new User(firstName, lastName);
}
```

Kotlin: Idioms

- Singleton objects
- String interpolation
- Elvis operator 🕺
- Destructuring
- Extension functions
- Scoping functions

kotlinlang.org/docs/reference/idioms.html

Maintaining History

Maintaining History

- Change extension `.java -> .kt`
- First commit
- Apply Kotlin conversion
- Second commit

Maintaining History

Refactoring

Refactoring

Refactoring

SOLID

- Single-responsibility
- Open-closed
- Liskov substitution
- Interface segregation
- Dependency inversion


JUST REMEMBER
BOB IS WATCHING YOU
@askashdavies

Refactoring

Single Responsibility

Perspective 

Static 

Asynchronicity

Concurrency is Hard 

Thread

```
new Thread(() -> {
   foo();
}).start();
```

CompletableFuture

```
CompletableFuture.supplyAsync(this::findSomeData)
   .thenApply(this:: intReturningMethod)
   .thenAccept(this::notify);
```

RxJava

```
Observable
  .fromIterable(resourceDraft.getResources())
  .flatMap(resourceServiceApiClient::createUploadContainer)
  .zipWith(Observable.fromIterable(resourceDraft.getResources()), Pair::create)
  .flatMap(uploadResources())
  .toList()
  .toObservable()
  .flatMapMaybe(resourceCache.getResourceCachedItem())
  .defaultIfEmpty(Resource.getDefaultItem())
  .flatMap(postResource(resourceId, resourceDraft.getText(), currentUser, resourceDraft.getIntent()))
  .observeOn(AndroidSchedulers.mainThread())
  .subscribeOn(Schedulers.io())
  .subscribe(
      resource -> repository.setResource(resourceId, resource, provisionalResourceId),
      resourceUploadError(resourceId, resourceDraft, provisionalResourceId)
  );
```

@askashdavies

What If? 🤔

Coroutines

@askashdavies

```
fun main() {
    GlobalScope.launch {
        delay(1000L)
        println("World!")
    }
    println("Hello,")
    Thread.sleep(2000L)
}

// Hello,
// World!
```

```
fun main() {
    GlobalScope.launch {
        delay(1000L)
        println("World!")
    }
    println("Hello,")
    Thread.sleep(2000L)
}

// Hello,
// World!
```

```
fun main() {
    GlobalScope.launch {
        delay(1000L)
        println("World!")
    }
    println("Hello,")
    Thread.sleep(2000L)
}

// Hello,
// World!
```

Stability

Annotations

`@Experimental`

Building SDKs - The Kotlin Way

Kotlin Everywhere: Hamburg - Jossi Wolf

speakerdeck.com/jossiwolf/building-sdks-the-kotlin-way-ccb4a237-b45c-42e7-8391-640dd058f50c

💪 Coroutines 💪

Native library

Efficient 

 Easy-to-use

 suspend fun

👌 Suspend

```
fun main() {
    GlobalScope.launch {
        delay(1000L)
        println("World!")
    }
    println("Hello,")
    Thread.sleep(2000L)
}

// Hello,
// World!
```

👌 Suspend

```
fun main() {
  GlobalScope.launch {
    doWorld()
    println("Hello,")
  }
  Thread.sleep(2000L)
}

suspend fun doWorld() {
  delay(1000L)
  println("World!")
}

// Hello,
// World!
```

@askashdavies

👌 Suspend

```
fun main() {
  GlobalScope.launch {
    launch { doWorld() }
    println("Hello,")
  }
  Thread.sleep(2000L)
}

suspend fun doWorld() {
  withContext(Dispatchers.IO) {
    delay(1000L)
    println("World!")
  }
}

// Hello,
// World!
```

@askashdavies

Testing

```
@Test
fun testFoo() = runBlockingTest {
  val actual = foo()
  // ...
}

suspend fun foo() {
  delay(1_000)
  // ...
}
```

Further Reading 📖

- **Google Codelab: Refactoring to Kotlin**
  codelabs.developers.google.com/codelabs/java-to-kotlin/
- **KotlinX Coroutine Test**
  github.com/Kotlin/kotlinx.coroutines/tree/master/kotlinx-coroutines-test
- **Sean McQuillan: Coroutines + Testing = ❤️**
  droidcon.com/media-detail?video=352671106
- **Ash Davies: RxJava & Coroutines: A Practical Analysis v3**
  speakerdeck.com/ashdavies/rxjava-and-coroutines-a-practical-analysis-v3

Thanks!

bit.ly/refactoring-legacy-code

@askashdavies