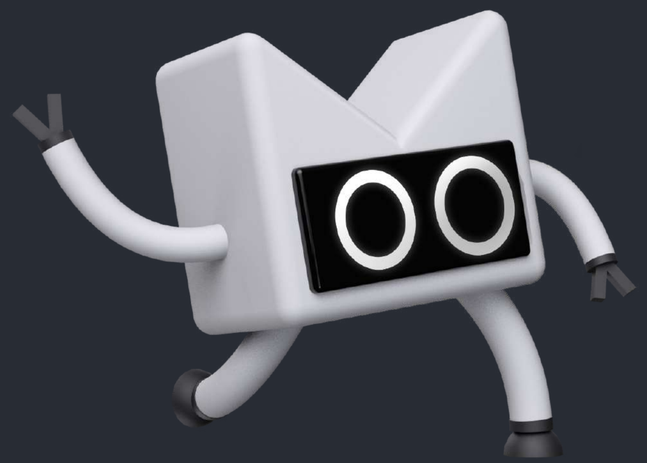
You Might Need to Resign Your Java Codes in Kotlin



Gary LO
@Gap撈Tech

My Kotlin Journey

since Kotlin 1.0

My Kotlin Journey

since Kotlin 1.0

2016 Feb • Kotlin 1.0 Released, self-learn with Java & Scala experience

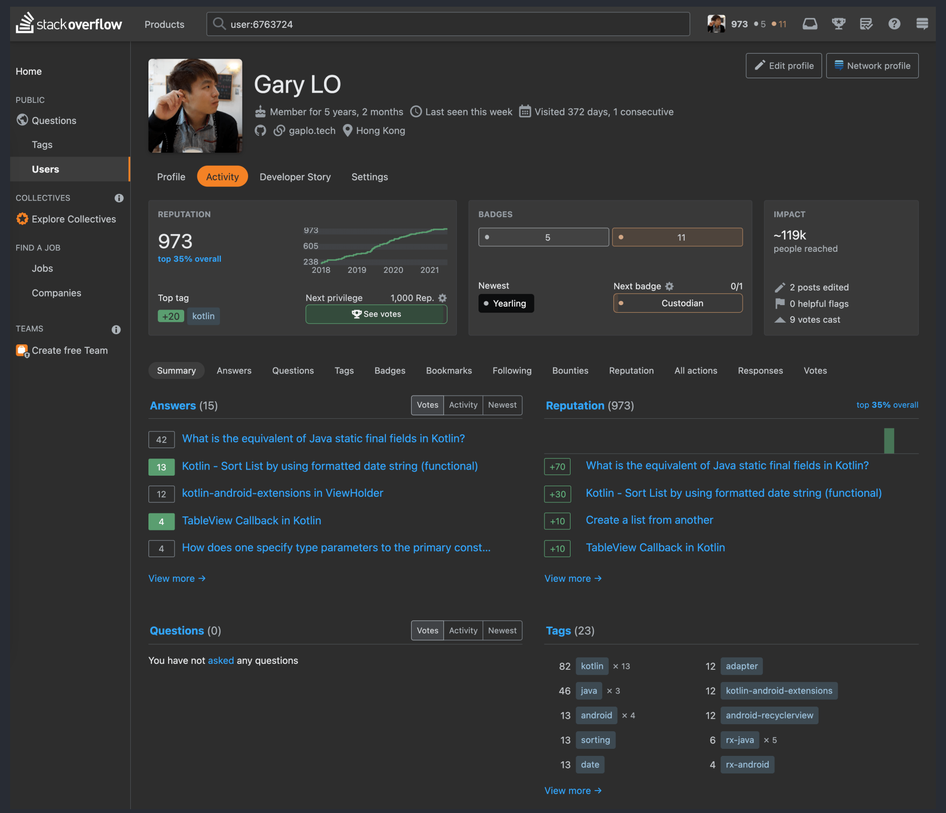
My Kotlin Journey

since Kotlin 1.0

2016 Feb — Kotlin 1.0 Released, self-learn with Java & Scala experience

2016 Jun — Lead mobile team to adopt Kotlin/JVM in Android
Answer StackOverflow Kotlin questions in leisure time

Picking Kotlin for Android



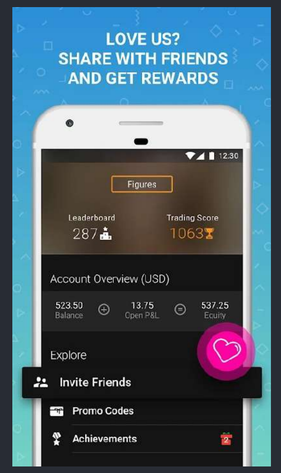
My Kotlin Journey

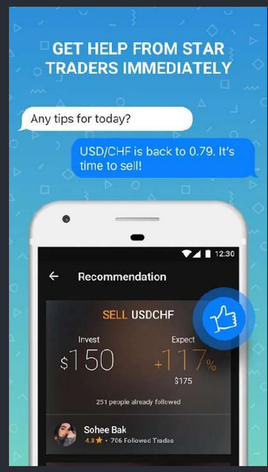
since Kotlin 1.0

2016 Feb — Kotlin 1.0 Released, self-learn with Java & Scala experience

2016 Jun — Lead mobile team to adopt Kotlin/JVM in Android
Answer StackOverflow Kotlin questions in leisure time

2016 Oct — Release 100% Kotlin Android FX Trading Application@1.0





My Kotlin Journey

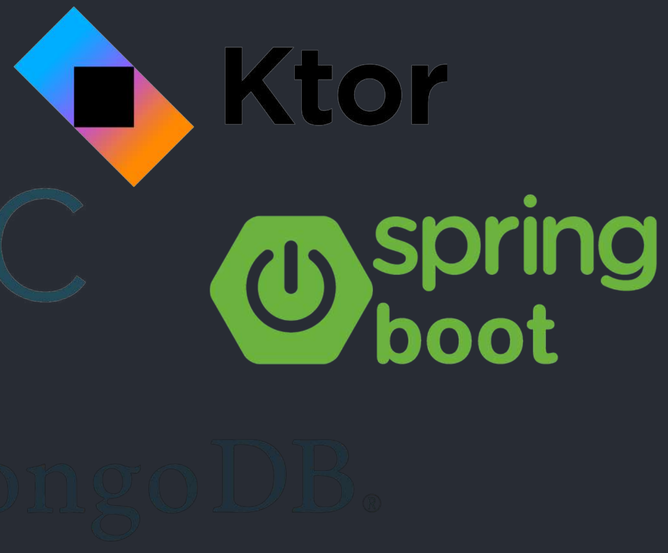
since Kotlin 1.0

2016 Feb — Kotlin 1.0 Released, self-learn with Java & Scala experience

2016 Jun — Lead mobile team to adopt Kotlin/JVM in Android
Answer StackOverflow Kotlin questions in leisure time

2016 Oct — Release 100% Kotlin Android FX Trading Application@1.0

2018 Jan — Lead backend team to adopt Kotlin/JVM for
building RESTful API Ktor, gRPC services, message broker connectors



My Kotlin Journey

since Kotlin 1.0

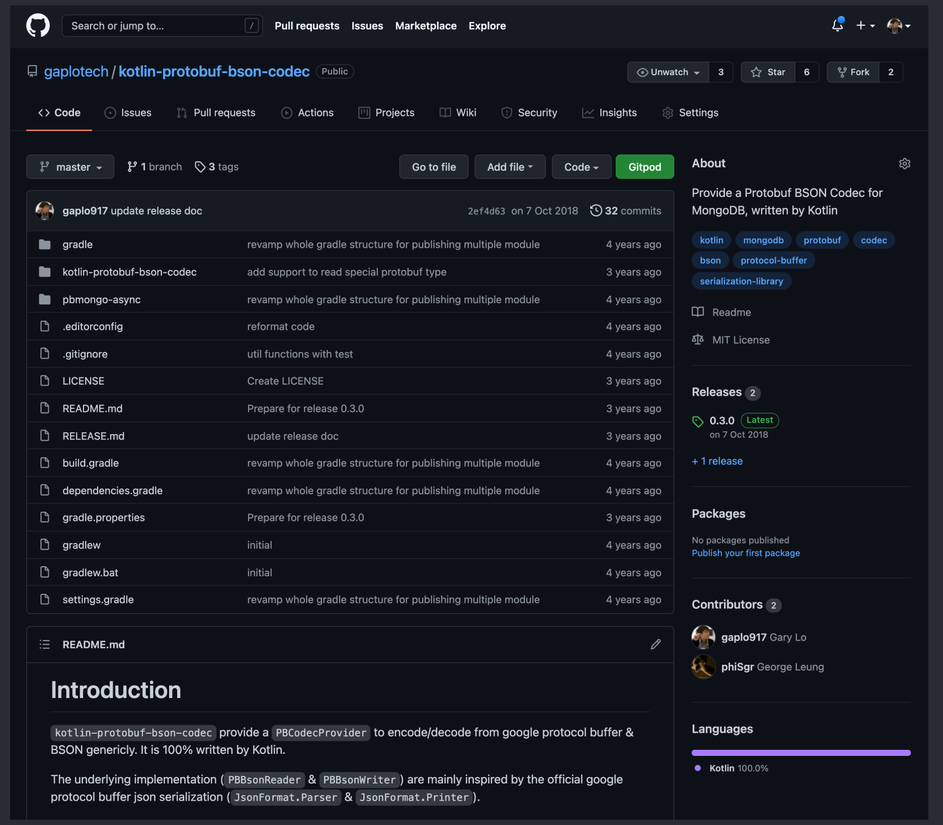
2016 Feb — Kotlin 1.0 Released, self-learn with Java & Scala experience

2016 Jun — Lead mobile team to adopt Kotlin/JVM in Android
Answer StackOverflow Kotlin questions in leisure time

2016 Oct — Release 100% Kotlin Android FX Trading Application@1.0

2018 Jan — Lead backend team to adopt Kotlin/JVM for
building RESTful API Ktor, gRPC services, message broker connectors

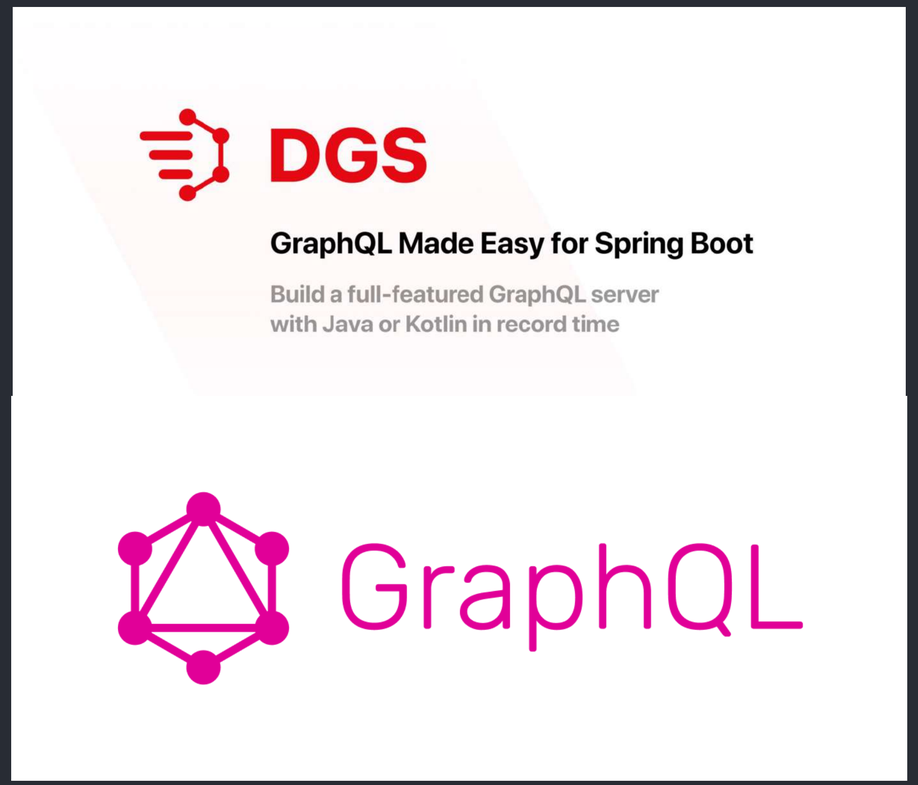
2018 Oct — Open Source Google Protobuf BSON Codec
for MongoDB & JVM Application



My Kotlin Journey

since Kotlin 1.0

2016 Feb	Kotlin 1.0 Released, self-learn with Java & Scala experience
2016 Jun	Lead mobile team to adopt Kotlin/JVM in Android Answer StackOverflow Kotlin questions in leisure time
2016 Oct	Release 100% Kotlin Android FX Trading Application@1.0
2018 Jan	Lead backend team to adopt Kotlin/JVM for building RESTful API Ktor, gRPC services, message broker connectors
2018 Oct	Open Source Google Protobuf BSON Codec for MongoDB & JVM Application
2021 Feb	Advocate Kotlin GraphQL (Netflix DGS framework) for new projects



My Kotlin Journey

since Kotlin 1.0

- 2016 Feb — Kotlin 1.0 Released, self-learn with Java & Scala experience
- 2016 Jun — Lead mobile team to adopt Kotlin/JVM in Android
  Answer StackOverflow Kotlin questions in leisure time
- 2016 Oct — Release 100% Kotlin Android FX Trading Application@1.0
- 2018 Jan — Lead backend team to adopt Kotlin/JVM for
  building RESTful API Ktor, gRPC services, message broker connectors
- 2018 Oct — Open Source Google Protobuf BSON Codec
  for MongoDB & JVM Application
- 2021 Feb — Advocate Kotlin GraphQL (Netflix DGS framework) for new projects
- 2021 - 2022 — Design & build Kotlin/Native & Kotlin/JS usage on WebView message bridge
  for next-gen mobile development

Recap about
Fundamental Kotlin Language Features

Recap about Fundamental Kotlin Language Features

changing the interface and model design

Recap about Fundamental Kotlin Language Features

changing the interface and model design

Recap about Fundamental Kotlin Language Features

changing the interface and model design

- Compile time null safety

Recap about Fundamental Kotlin Language Features

changing the interface and model design

- Compile time null safety
- Zero Overhead Nullable Type

Recap about Fundamental Kotlin Language Features

changing the interface and model design

- Compile time null safety
- Zero Overhead Nullable Type
- Nullable Chaining and Default Value

Recap about Fundamental Kotlin Language Features

changing the interface and model design

- Compile time null safety
- Zero Overhead Nullable Type
- Nullable Chaining and Default Value
- Everything is an expression

Recap about Fundamental Kotlin Language Features

changing the interface and model design

- Compile time null safety
- Zero Overhead Nullable Type
- Nullable Chaining and Default Value
- Everything is an expression
- Type Inference

Recap about Fundamental Kotlin Language Features

changing the interface and model design

- Compile time null safety
- Zero Overhead Nullable Type
- Nullable Chaining and Default Value
- Everything is an expression
- Type Inference
- Extension functions

Recap about Fundamental Kotlin Language Features

changing the interface and model design

- Compile time null safety
- Zero Overhead Nullable Type
- Nullable Chaining and Default Value
- Everything is an expression
- Type Inference
- Extension functions
- Sealed Class, Sealed Interface

Recap about Fundamental Kotlin Language Features

changing the interface and model design

- Compile time null safety
- Zero Overhead Nullable Type
- Nullable Chaining and Default Value
- Everything is an expression
- Type Inference
- Extension functions
- Sealed Class, Sealed Interface
- Infix function and operator overloading

Recap about Fundamental Kotlin Language Features

changing the interface and model design

- Compile time null safety
- Zero Overhead Nullable Type
- Nullable Chaining and Default Value
- Everything is an expression
- Type Inference
- Extension functions
- Sealed Class, Sealed Interface
- Infix function and operator overloading
- Coroutine

Why Redesign?

Kotlin can do what Java does in the Same Way. But...

Kotlin can do much better than Java

Kotlin can do much better than Java

You can reduce “debug complexity“
by
designing “Kotlin-way” abstraction.

Kotlin can do much better than Java

You can reduce “debug complexity“
by
designing “Kotlin-way” abstraction.

$O(N^2)$ -> $O(N \log N)$

Compile Time Null Safety

Compile Time Null Safty

Color Scheme:  GapStyle

Compile Time Null Safty

```java
public class NullSafeDemoJava {
    Integer something = null;

    public Integer someCalculation(Integer other) {
        return something * other;
    }

}
```

Easy do it wrong

Color Scheme: GapStyle

Compile Time Null Safty

```java
public class NullSafeDemoJava {
    Integer something = null;

    public Integer someCalculation(Integer other) {
        return something * other;
    }

}
```

Easy do it wrong

Do more work to let IDE alert the nullable value
by adding `@Nullable`

```java
import org.jetbrains.annotations.Nullable;

public class NullSafeDemoJava {
    @Nullable Integer something = null;

    public Integer someCalculation(@Nullable Integer other) {
        return something * other;
    }

}
```

Unboxing of 'other' may produce 'NullPointerException'

```
@Nullable
Integer other
```

Color Scheme: GapStyle

Compile Time Null Safty

```java
public class NullSafeDemoJava {
    Integer something = null;

    public Integer someCalculation(Integer other) {
        return something * other;
    }

}
```

Easy do it wrong

Do more work to let IDE alert the nullable value
by adding `@Nullable`

```java
import org.jetbrains.annotations.Nullable;

public class NullSafeDemoJava {
    @Nullable Integer something = null;

    public Integer someCalculation(@Nullable Integer other) {
        return something * other;
    }

}
```

Unboxing of 'other' may produce 'NullPointerException'

```java
@Nullable
Integer other
```

The above code still pass in Java compiler

Color Scheme: GapStyle

Compile Time Null Safty

```java
public class NullSafeDemoJava {
    Integer something = null;

    public Integer someCalculation(Integer other) {
        return something * other;
    }

}
```

Easy do it wrong

Do more work to let IDE alert the nullable value
by adding `@Nullable`

```java
import org.jetbrains.annotations.Nullable;

public class NullSafeDemoJava {
    @Nullable Integer something = null;

    public Integer someCalculation(@Nullable Integer other) {
        return something * other;
    }

}
```

Unboxing of 'other' may produce 'NullPointerException'

```
@Nullable
Integer other
```

The above code still pass in Java compiler

```kotlin
class NullSafeDemo {
    private var something: Int? = null

    fun someCalculation(other: Int?): Int? {
        return something * other
    }
}
```

Type mismatch.
Required: Int
Found: Int?

Add non-null asserted (!!) call ⌥⇧⏎ More actions... ⌥⏎

value-parameter other: Int?

The above code don't pass in Kotlin compiler

Color Scheme: GapStyle

Compile Time Null Safty

```java
public class NullSafeDemoJava {
    Integer something = null;

    public Integer someCalculation(Integer other) {
        return something * other;
    }

}
```

Easy do it wrong

Do more work to let IDE alert the nullable value
by adding `@Nullable`

```java
import org.jetbrains.annotations.Nullable;

public class NullSafeDemoJava {
    @Nullable Integer something = null;

    public Integer someCalculation(@Nullable Integer other) {
        return something * other;
    }

}
```

Unboxing of 'other' may produce 'NullPointerException'

```
@Nullable
Integer other
```

The above code still pass in Java compiler

```kotlin
class NullSafeDemo {
    private var something: Int? = null

    fun someCalculation(other: Int?): Int? {
        return something * other
    }
}
```

Type mismatch.
Required: Int
Found: Int?

Add non-null asserted (!!) call ⌥⇧⏎ More actions... ⌥⏎

value-parameter other: Int?

The above code don't pass in Kotlin compiler

```kotlin
class NullSafeDemo {
    private var something: Int? = null

    fun someCalculation(other: Int): Int? {
        return something * other
    }
}
```

Operator call corresponds to a dot-qualified call 'something.times(other)' which is not allowed on a nullable receiver 'something'.

Replace with safe (?.) call ⌥⇧⏎ More actions... ⌥⏎

The above code don't pass in Kotlin compiler

Color Scheme: GapStyle

Nullable Chaining

Nullable Chaining

Color Scheme: GapStyle

Nullable Chaining

```java
import java.util.Date;

public class NullableChainingJava {
    static class A {
        public B someB = null;
        static class B {
            public C someC = null;
            static class C {
                public Date date = null;
            }
        }
    }

    void run() {
        A a = new A();
        if(a.someB != null) {
            if(a.someB.someC != null){
                if(a.someB.someC.date != null) {
                    // safe to use date
                    System.out.println(a.someB.someC.date);
                }
            }
        }
    }
}
```

Chaining in Java

Color Scheme: GapStyle

Nullable Chaining

```java
import java.util.Date;

public class NullableChainingJava {
    static class A {
        public B someB = null;
        static class B {
            public C someC = null;
            static class C {
                public Date date = null;
            }
        }
    }

    void run() {
        A a = new A();
        if(a.someB != null) {
            if(a.someB.someC != null){
                if(a.someB.someC.date != null) {
                    // safe to use date
                    System.out.println(a.someB.someC.date);
                }
            }
        }
    }
}
```

Chaining in Java

```kotlin
class NullableChaining {
    internal class A {
        var someB: B? = null
        internal class B {
            var someC: C? = null
            internal class C {
                var date: Date? = null
            }
        }
    }

    fun run() {
        val a = A()
        println(a.someB?.someC?.date)
    }
}
```

Nullable Chaining in Kotlin

Color Scheme: GapStyle

Named Arguments and Default Value

Named Arguments and Default Value

Named Arguments and Default Value

```java
import org.jetbrains.annotations.Nullable;
import java.util.Date;

public class NamedArgumentsJava {
    @Nullable Integer someInt;
    @Nullable Boolean someBoolean;
    @Nullable Date someDate;

    void updateSomething(@Nullable Boolean bool) {
        updateSomething(bool,  integer: null,  someDate: null);
    }

    void updateSomething(@Nullable Integer integer) {
        updateSomething( bool: null, integer,  someDate: null);
    }

    void updateSomething(@Nullable Date date, @Nullable Integer integer) {
        updateSomething( bool: null, integer, date);
    }

    void updateSomething(@Nullable Boolean bool, @Nullable Integer integer, @Nullable Date someDate) {
        this.someBoolean = bool;
        this.someInt = integer;
        this.someDate = someDate;
        sideEffect();
    }


    void sideEffect() {}
}
```

Method Overloading in Java

Color Scheme: GapStyle

Named Arguments and Default Value

Named Arguments and Default Value

```
import java.util.*

class NamedArgumentsKt {
    private var someInt: Int? = null
    private var someBoolean: Boolean? = null
    private var someDate: Date? = null

    fun updateSomething(someBoolean: Boolean? = null, someInt: Int? = null, someDate: Date? = null) {
        this.someBoolean = someBoolean
        this.someInt = someInt
        this.someDate = someDate
        sideEffect()
    }

    private fun sideEffect() {}
}
```

Use Default Value in the parameter design

Color Scheme: GapStyle

Named Arguments and Default Value

```kotlin
import java.util.*

class NamedArgumentsKt {
    private var someInt: Int? = null
    private var someBoolean: Boolean? = null
    private var someDate: Date? = null

    fun updateSomething(someBoolean: Boolean? = null, someInt: Int? = null, someDate: Date? = null) {
        this.someBoolean = someBoolean
        this.someInt = someInt
        this.someDate = someDate
        sideEffect()
    }

    private fun sideEffect() {}
}
```

Use Default Value in the parameter design

```kotlin
    fun main(){
        val sth = NamedArgumentsKt()
        sth.updateSomething(someInt = 1, someBoolean = true, someDate = Date())
        sth.updateSomething(someBoolean = true)
        sth.updateSomething(someInt = 1, someDate = Date())
        sth.updateSomething(someDate = Date())
    }
```

Use Named Arguments in design

Color Scheme: GapStyle

Everything is an expression
&
Type Inference

Everything is an expression
&
Type Inference

Everything is an expression
&
Type Inference

```java
public class EverythingIsExpressionJava {
    String sth;
    String sth2;

    void run() {
        Integer a;
        if(sth.equals("something")) {
            a = 1;
        } else if(sth2.equals("something2")) {
            a = 2;
        } else {
            a = 3;
        }
        System.out.println(a);
    }
}
```

Standard Java Implementation

Color Scheme: GapStyle

Everything is an expression
&
Type Inference

```java
public class EverythingIsExpressionJava {
    String sth;
    String sth2;

    void run() {
        Integer a;
        if(sth.equals("something")) {
            a = 1;
        } else if(sth2.equals("something2")) {
            a = 2;
        } else {
            a = 3;
        }
        System.out.println(a);
    }
}
```

Standard Java Implementation

```kotlin
var sth: String? = null
var sth2: String? = null

fun run() {
    val a: Int?
    if (sth == "something") {
        a = 1
    } else if (sth2 == "something2") {
        a = 2
    } else {
        a = 3
    }
    println(a)
}
```

Kotlin in Java way

Color Scheme: GapStyle

Everything is an expression
&
Type Inference

```java
public class EverythingIsExpressionJava {
    String sth;
    String sth2;

    void run() {
        Integer a;
        if(sth.equals("something")) {
            a = 1;
        } else if(sth2.equals("something2")) {
            a = 2;
        } else {
            a = 3;
        }
        System.out.println(a);
    }
}
```

Standard Java Implementation

```kotlin
var sth: String? = null
var sth2: String? = null

fun run() {
    val a: Int?
    if (sth == "something") {
        a = 1
    } else if (sth2 == "something2") {
        a = 2
    } else {
        a = 3
    }
    println(a)
}
```

Kotlin in Java way

```kotlin
var sth: String? = null
var sth2: String? = null

fun optimized() {
    val a = when (sth) {
        "something" -> 1
        "something2" -> 2
        else -> 3
    }
    println(a)
}
```

Kotlin way

Color Scheme:  GapStyle

Extension Function

Extension Function

Extension Function

```java
public class StringUtils {
    String append2021Kotlin(String string) {
        return string + " " + "2021 kotlin";
    }

    String appendChineseMeetup(String string) {
        return string + " " + " 中文大會";
    }

    void run() {
        appendChineseMeetup(append2021Kotlin( string: "Gap is in"));
    }
}
```

Standard Java Implementation

Color Scheme: GapStyle

Extension Function

```java
public class StringUtils {
    String append2021Kotlin(String string) {
        return string + " " + "2021 kotlin";
    }

    String appendChineseMeetup(String string) {
        return string + " " + " 中文大會";
    }

    void run() {
        appendChineseMeetup(append2021Kotlin( string: "Gap is in"));
    }
}
```

Standard Java Implementation

```kotlin
object StringUtils {
    fun append2021Kotlin(string: String): String {
        return "$string 2021 kotlin"
    }

    fun appendChineseMeetup(string: String): String {
        return "$string 中文大會"
    }

    fun run() {
        appendChineseMeetup(append2021Kotlin( string: "Gap is in"))
    }
}
```

Kotlin in Java way

Color Scheme: GapStyle

Extension Function

Extension Function

```java
public class StringUtils {
    String append2021Kotlin(String string) {
        return string + " " + "2021 kotlin";
    }

    String appendChineseMeetup(String string) {
        return string + " " + " 中文大會";
    }

    void run() {
        appendChineseMeetup(append2021Kotlin( string: "Gap is in"));
    }
}
```

Standard Java Implementation

Color Scheme: GapStyle

Extension Function

```java
public class StringUtils {
    String append2021Kotlin(String string) {
        return string + " " + "2021 kotlin";
    }

    String appendChineseMeetup(String string) {
        return string + " " + " 中文大會";
    }

    void run() {
        appendChineseMeetup(append2021Kotlin( string: "Gap is in"));
    }
}
```

Standard Java Implementation

```kotlin
fun String.append2021Kotlin(): String {
    return "$this 2021 kotlin"
}

fun String.appendChineseMeetup(): String {
    return "$this 中文大會"
}

fun main() {
    "Gap is in"
        .append2021Kotlin()
        .appendChineseMeetup()
}
```

Kotlin Way

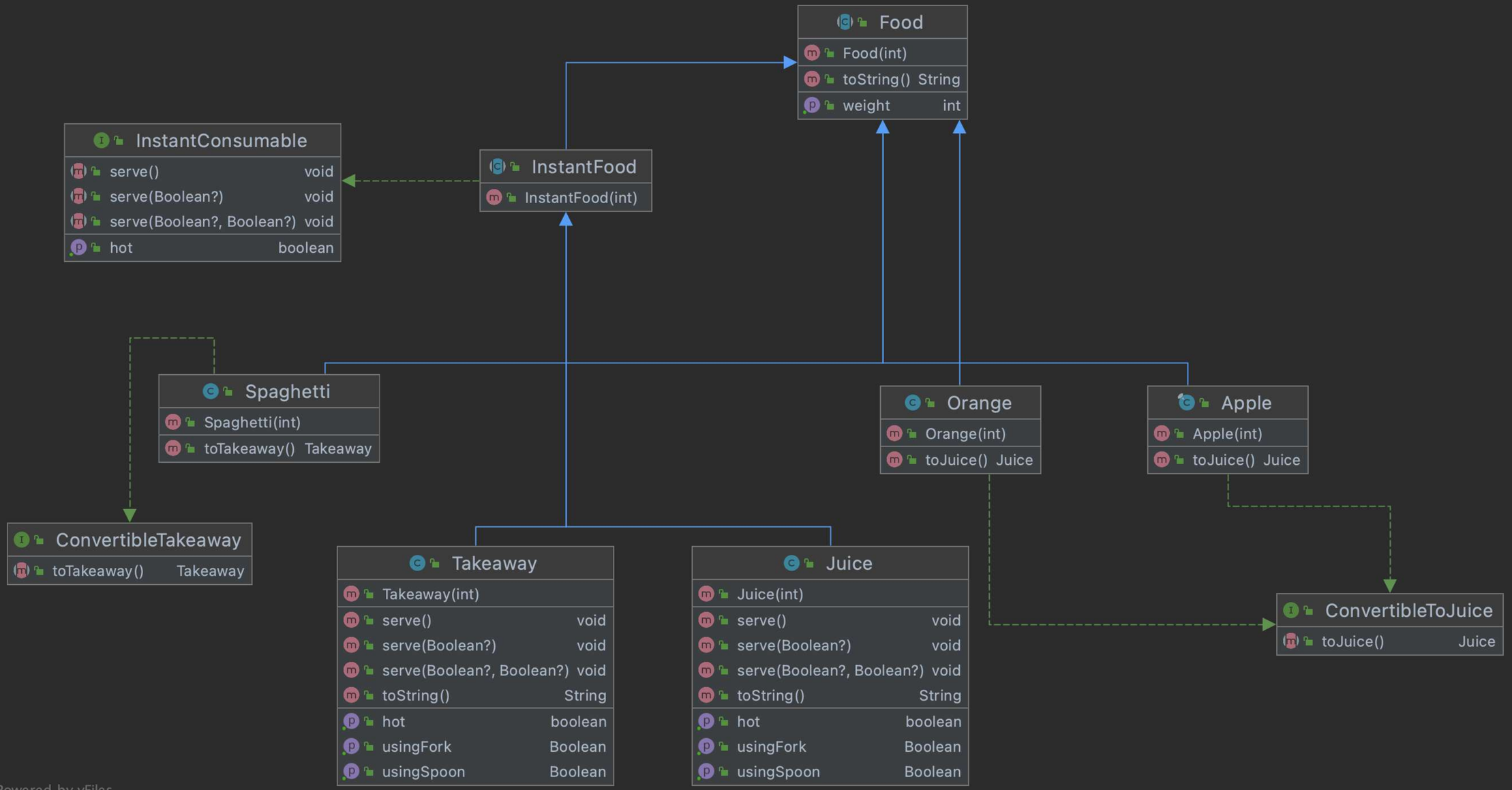
Color Scheme: GapStyle

Sample Use Case
Live Demo Migrate Java Code

Sample Use Case - Migrating Java Code

Color Scheme: GapStyle

Sample Use Case - Migrating Java Code



Live Demo

Challenge: Create your Bio with own DSL

GitHub: Source

```kotlin
package gaplotech

import java.util.concurrent.Future
import bio.*

/**
 * Created By Gary Lo on 30/3/2018
 * FacebookCoverPhoto.kt
 */
object GapLoTech: SoftwareEngineer(), Passion, ProgrammingLanguage by (Kotlin + Swift + Scala + Javascript) {

    fun write(): Future<Article> {
        return releaseEnergy()
            .exceptionally { throw GapShouldGoSleepException() }
            .thenApply { energy ->
                when(energy.type) {
                    is Work -> Experiences(
                        energy on FunctionalProgramming withLanguage Scala `for` "training declarative mindset",
                        energy on ReactiveProgramming withLanguage (Kotlin + Swift) `for` "training reactive mindset",
                        energy on Akka withLanguage Scala `for` "building highly concurrent & fault tolerant & distributed system",
                        energy on Coroutine withLanguage Kotlin `for` "building low latency & efficient application",
                        energy on (Mobile withPattern MVVM)
                                withTool (RxKotlin + RxSwift) `for` "building reactive mobile apps"
                    )

                    is Leisure -> Experiences(
                        energy on SideProject withLanguage Javascript `for` "fun",
                        energy on ModernWeb withTool (React + Vue + Babel + Webpack) `for` "progressive web app",
                        energy on Ops withTool (Docker + Kubernetes + GitLabCI) `for` "scaling & automation"
                    )
                }
            }
            .thenApply { experience -> Inspiration(source = experience) }
            .thenApply { inspiration -> Article(inspiration) }
    }

}

fun main(){
    GapLoTech.write()
}
```

Color Scheme: GapStyle

Thank You

Gap撈Tech

Blogs (Cantonese)
https://gaplo.tech

Blogs (English)
https://intl.gaplo.tech

Facebook Page
https://facebook.com/gaplotech