ThoughtWorks®

Concise & Safe

GETTING STARTED WITH KOTLIN ON ANDROID

Wang Zhiyong & Zhang Shuai

KOTLIN 简介

> Kotlin是一门与Swift类似的**静态类型**JVM语言，由JetBrains设计开发并开源。与Java相比，Kotlin的语法更**简洁**、更具**表达性**，而且提供了更多的**特性**。


Insights
Physical
Layout
Shop within
a budget
markdown over
bulk deals

神奇的KOTLIN

KOTLIN代码

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

函数

KOTLIN代码

```
fun sum(a: Int, b: Int): Int = a + b
```

函数

KOTLIN代码

```
fun sum(a: Int, b: Int) = a + b
```

函数

KOTLIN代码

```
private val mobileOpenDay = "Mobile Open Day"
```

```
private var phase: String = "1st"
```

```
phase = "4th"
```

```
mobileOpenDay = "BQMeetUp"
```

常量 & 变量

KOTLIN代码

```
private val topics = """
  《三生三世iOS布局》
  《移动测试的Mock实践》
  《Getting Start with Kotlin on Android》
"""
```

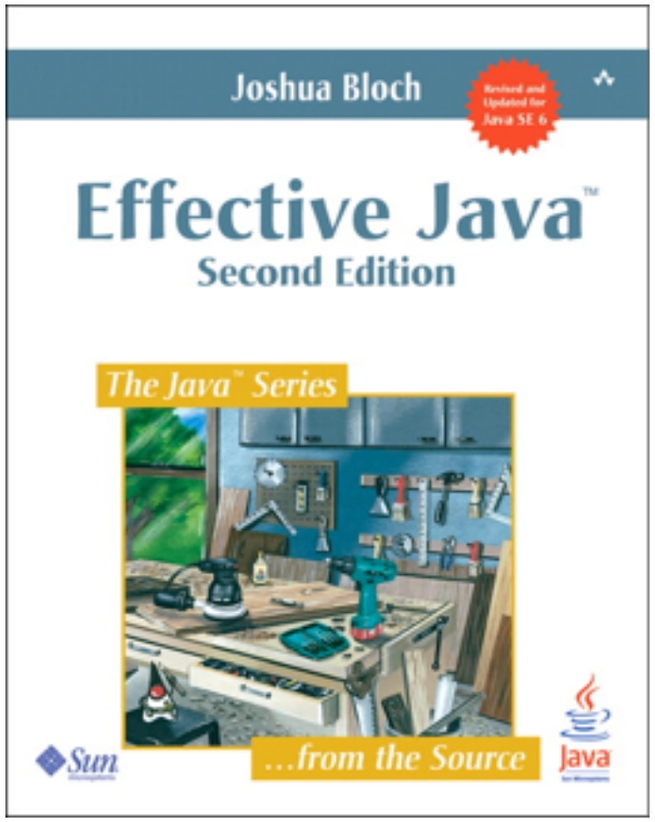
常量 & 变量

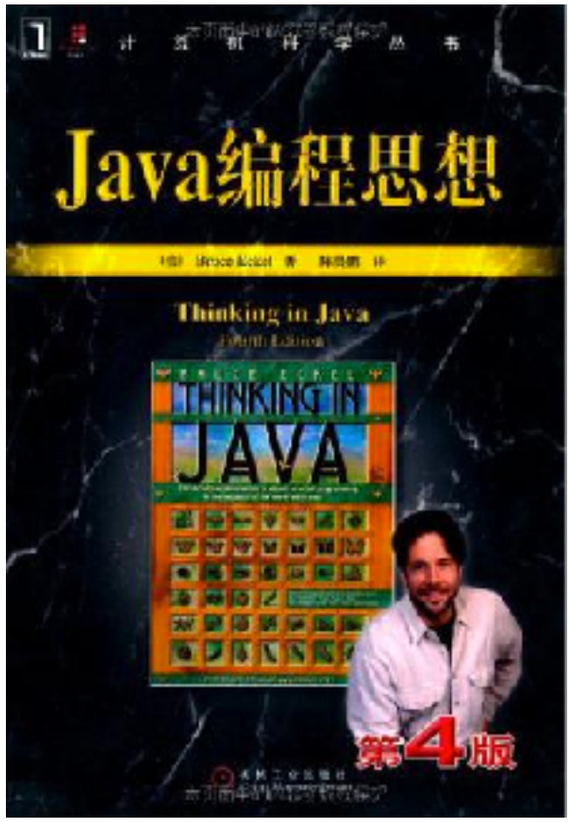
KOTLIN代码

```
private val topics = """
| 《三生三世iOS布局》
| 《移动测试的Mock实践》
| 《Getting Start with Kotlin on Android》
""".trimMargin()
```

常量 & 变量

JAVA VS KOTLIN


Joshua Bloch
Revised and Updated for Java SE 6
Effective Java™
Second Edition
The Java™ Series
Sun
...from the Source
Java


Java编程思想
Thinking in Java
THINKING IN JAVA
第4版

PAIN IN JAVA DEVELOPMENT

- 繁琐的语法，低级的API
- 随时可能出现的null pointer问题
- 默认可变的变量
- 各种各样的util类
- 混乱的泛型

“

Data class

”

JAVA代码

成员

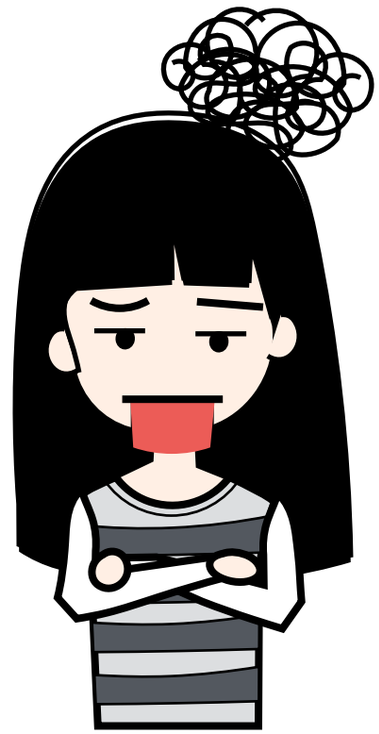
```
public class Session {
    private String topic;
    private String type;
    private List<String> speakers;
```

Getter/Setter

```
    public String getTopic() {…}

    public void setTopic(String topic) {…}

    public String getType() {…}

    public void setType(String type) {…}

    public List<String> getSpeakers() {…}

    public void setSpeakers(List<String> speakers) {…}
```

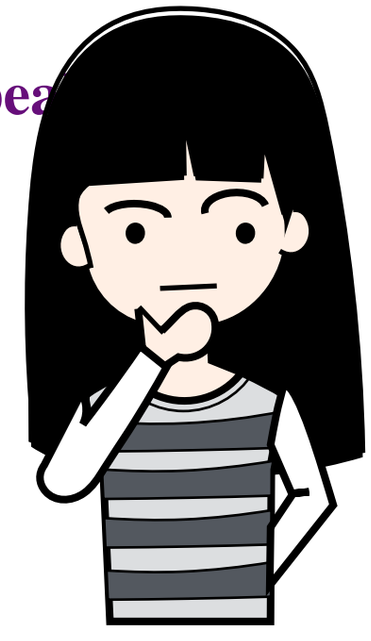
Others

```
    @Override
    public String toString() {…}

    @Override
    public int hashCode() {…}

    @Override
    public boolean equals(Object obj) {…}
}
```



KOTLIN代码

```
class SessionKT {
    val topic: String
    val type: String
    val speakers: MutableList<String>

    constructor(topic: String, type: String, speakers: MutableList<String>) {
        this.topic = topic
        this.type = type
        this.speakers = speakers
    }

    override fun toString(): String {
        return "SessionKT(topic='$topic', type='$type', speakers=$spea
    }
}
```



KOTLIN代码

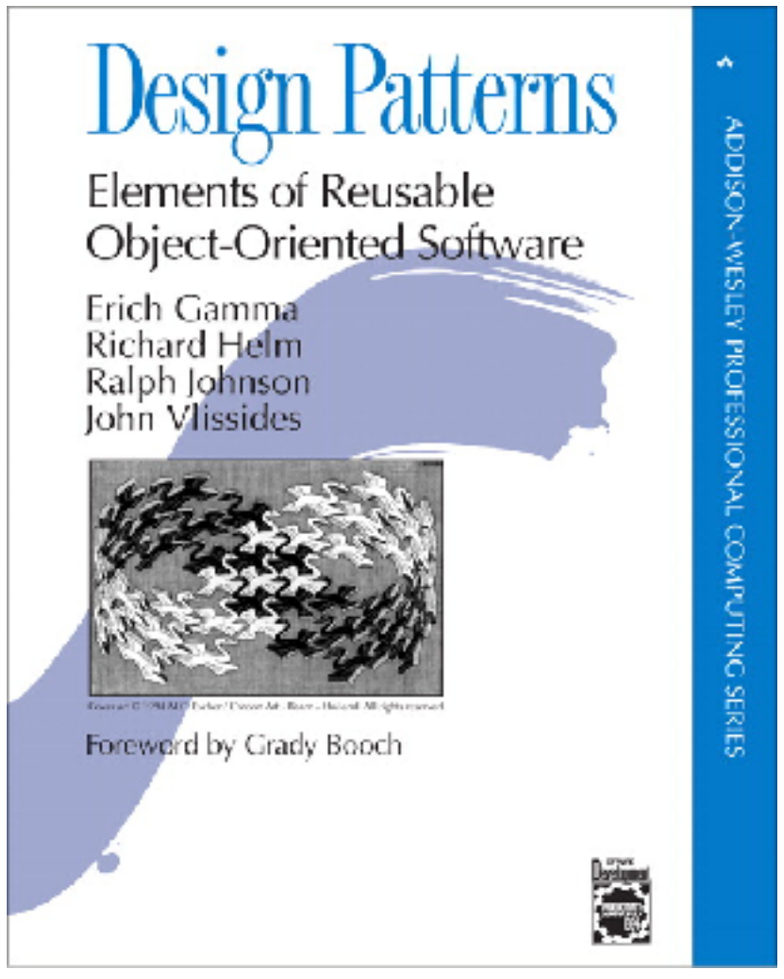
```
class SessionKT(val topic: String, val type: String, val speakers: MutableList<String>) {

    override fun toString(): String = "SessionKT(topic='$topic', type='$type',
speakers=$speakers)”

}
```

KOTLIN代码

```
data class SessionKT(val topic: String,
                     val type: String,
                     val speakers: MutableList<String>)
```

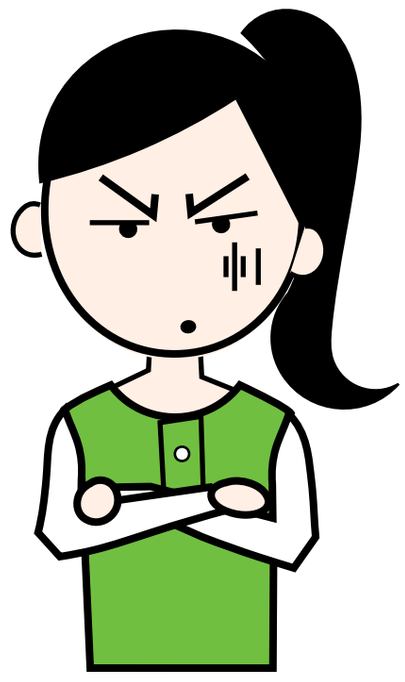
Kotlin教做人啊

Singleton


Design Patterns
Elements of Reusable
Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

JAVA实现一个单例

```
class SingletonImpl {
    private volatile static SingletonImpl instance;

    private SingletonImpl() {
    }

    public static SingletonImpl getInstance() {
        if (instance == null) {
            synchronized (SingletonImpl.class) {
                if (instance == null)
                    instance = new SingletonImpl();
            }
        }
        return instance;
    }
}
```



KOTLIN来实现一个单例

```
object SingletonImpl
```

Kotlin教做人啊

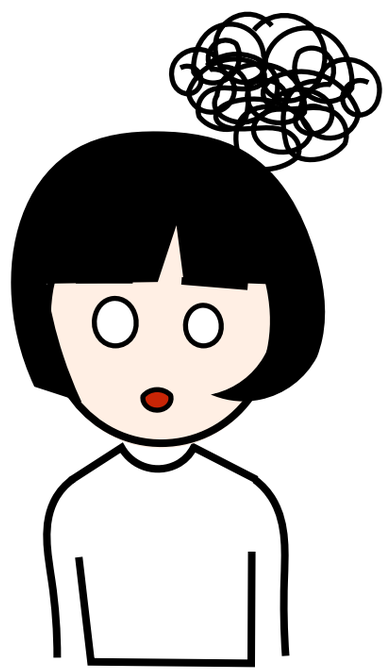
“Optional”

在代码的BUG中，NULL POINTER大家熟悉不过了

举个例子吧

```
String home = null;
System.out.println("home is: " + home +
home.length());
```

```
Exception in thread "main" java.lang.NullPointerException
    at com.thoughtworks.china.mobile.optional.Developer.main(Developer.java:26)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at com.intellij.rt.execution.application.AppMain.main(AppMain.java:147)
```



无处不在的NULL坑在等你

```
if (user.getCompany() != null
        && user.getCompany()
.getAddress() != null
        && user.getCompany().getAddress()
.getPostcode() != null) {
         user.getCompany().getAddress()
            .getPostcode();
}
```

```
user.company?.address?.postcode
```

JAVA VS KOTLIN

我们尝试使用GUAVA来解决问题

```
private Optional<String> home = Optional.absent();


if (!home.isPresent()) {
        home = Optional.of("New House");
}
System.out.println(" home is: " + home.get()
        + home.get().length());
```

引入Guava的Optional方法

KOTLIN怎么玩

```
var home: String? = null


println("home is: ${home ?: "New House”}, ${home?.length}")
when {
    home.equals(null) -> home = "New House"
}
println("home is: ${home}, length: ${home!!.length}”)
```

```
Zhang Shuai's home is: New House, length: null
Zhang Shuai's home is: New House, length: 9
```

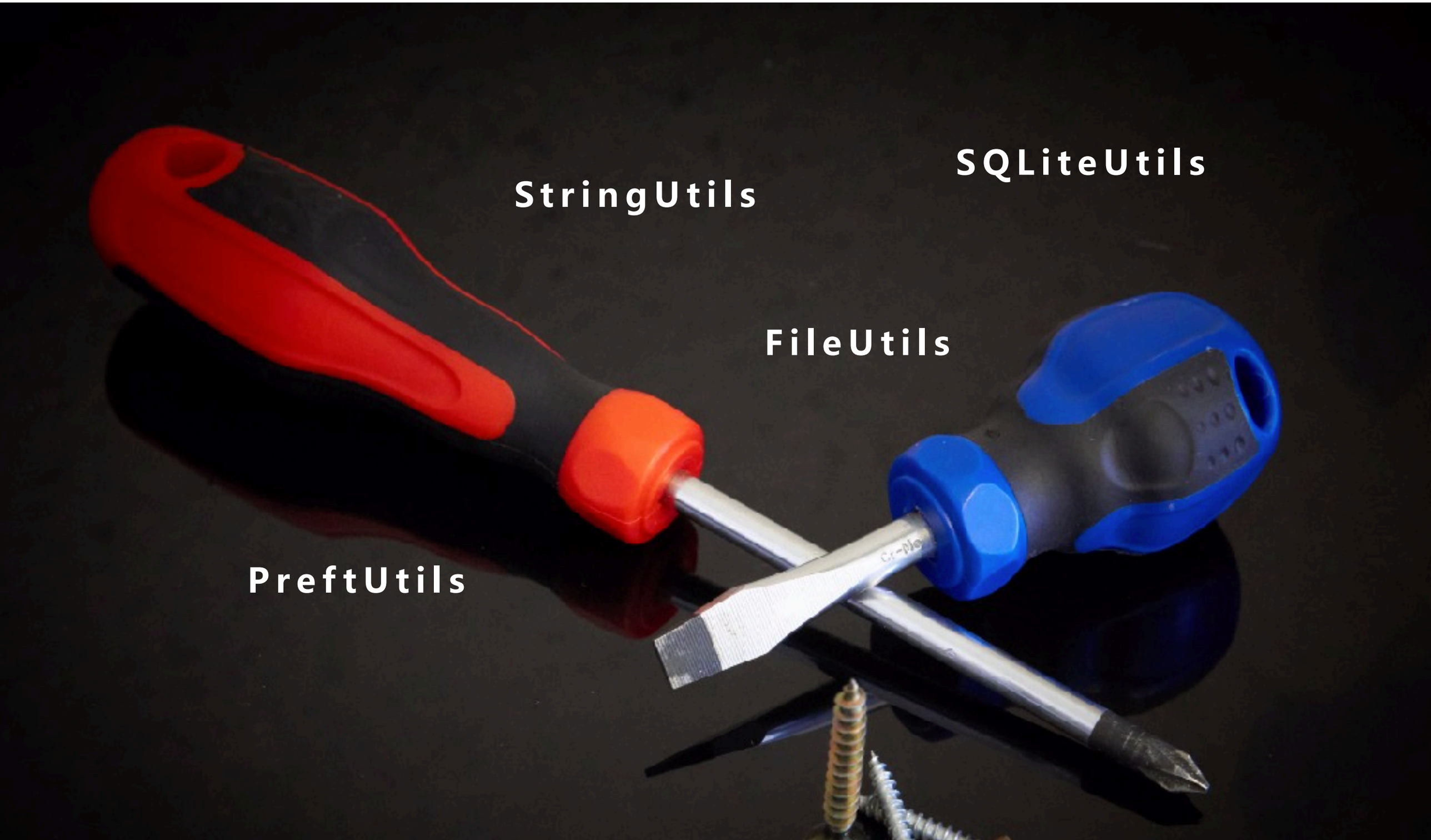
Late-Initialized & Lazy

KOTLIN怎么办

```
class Developer(var firstName: String, var lastName: String) {
    lateinit var company: String
     val fullName: String by lazy { "$lastName $firstName" }
}
```

Extensions

如果我们想要扩展已有的类，怎么办？



看个例子吧

```
public class DoubleExtension {
    public static double km(double value) {
        return value * 1000.0;
    }

    public static double m(double value) {
        return value;
    }

    public static double cm(double value) {
        return value / 100.0;
    }

    public static double mm(double value) {
        return value / 1000.0;
    }

    public static double ft(double value) {
        return value / 3.28084;
    }
}
```

使用方法：

```
double value = 2.0f;
System.out.println(String.format("
km: %f\nm: %f\ncm: %f\nmm:
%f\nft: %f",
        DoubleExtension.km(value),
        DoubleExtension.m(value),
        DoubleExtension.cm(value),
        DoubleExtension.mm(value),
        DoubleExtension.ft(value)));
```

KOTLIN怎么玩

```
fun Double.km() = this * 1_000.0

fun Double.m() = this

fun Double.cm() = this / 100.0

fun Double.mm() = this / 1_000.0

fun Double.ft() = this / 3.28084
```

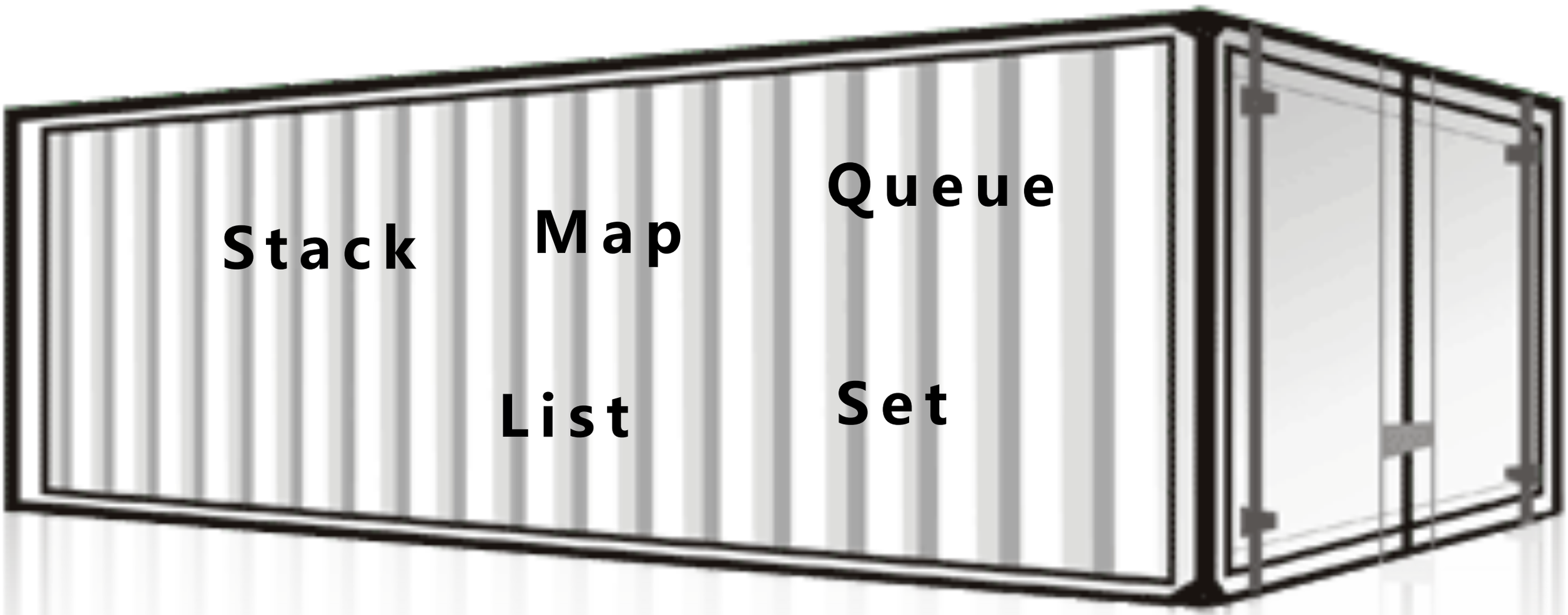
Kotlin教做人啊

使用方法：

```
val value: Double = 2.0
println("""
|km: ${value.km()}
|m: ${value.m()}
|cm: ${value.cm()}
|mm: ${value.mm()}
|ft: ${value.ft()}
""".trimMargin())
```

Collections

我们有什么种类的COLLECTIONS



看个例子吧

```
List<Integer> integers =
ImmutableList.of(1, 2, 3);
integers.add(4);
```

```
Exception in thread "main" java.lang.UnsupportedOperationException
    at com.google.common.collect.ImmutableCollection.add(ImmutableCollection.java:202)
    at com.thoughtworks.china.mobile.collection.Collections.main(Collections.java:18)
```

```
List<Integer> integers = newArrayList(1, 2, 3);
integers.add(4);
```

KOTLIN怎么玩

```
val integers = listOf(1, 2, 3)
integers.add(4)
```

就没有这个方法

```
val integers = mutableListOf(1, 2, 3)
integers.add(4)
```

这是可以编译的

Kotlin将Collections分为**mutable**跟**immutable**两类，
有助于消除错误，设计更好的API

Kotlin教做人啊

考虑到我们有这样一个需求

计算1到10所有奇数的平方的和

JAVA的实现

```
ArrayList<Integer> range = newArrayList(1, 2, 3, 4, 5, 6, 7,
8, 9, 10);
FluentIterable<Integer> temps = FluentIterable.from(range)
        .filter(new Predicate<Integer>() {
            @Override
            public boolean apply(Integer input) {
                return input % 2 == 1;
            }
        })
        .transform(new Function<Integer, Integer>() {
            @Override
            public Integer apply(Integer input) {
                return input * input;
            }
        });

int sum = 0;
for (int temp : temps) {
    sum += temp;
}
System.out.println(sum);
```

KOTLIN怎么玩

```
(1..10)
    .filter { it % 2 == 1 }          [1, 3, 5, 7, 9]
    .map { it * it }                 [1, 9, 25, 49, 81]
    .reduce { acc, i -> acc + i }    165
    .apply { println(this) }
```

Kotlin对**Stream**的支持更好，同时满足**Lambda**表达式

Kotlin教做人啊

Generics

GENERICS

- In Java ...

```
String[] strings = new String[]{"a"};
CharSequence[] charSequences = strings;
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/bin/java ...
Exception in thread "main" java.lang.ArrayStoreException: java.lang.StringBuffer
    at com.thoughtworks.china.mobile.generic.java.GenericJava.main(GenericJava.java:14)
```

GENERICS

- In Java …

```
List<String> strings = new ArrayList<>();
List<CharSequence> charSequences = strings;
```

```
List<String> strings = new ArrayList<>();
List<CharSequence> charSequences = strings;
```

```
List<String> strings = new ArrayList<>();
List<? extends CharSequence> charSequences = strings;

public <T> void addAll(List<? super T> to,
                       List<? extends T> from) {
    to.addAll(from);
}
```

GENERICS

- In Java ...

```java
String[] strings = new String[]{"a"};
CharSequence[] charSequences = strings;

charSequences[0] = new StringBuffer("abc");
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/bin/java ...
Exception in thread "main" java.lang.ArrayStoreException: java.lang.StringBuffer
    at com.thoughtworks.china.mobile.generic.java.GenericJava.main(GenericJava.java:14)
```

- In Kotlin ...

```kotlin
val strings = arrayOf("a")
val charSequences: Array<out CharSequence> = strings

charSequences[0] = StringBuffer("abc")
```

GENERICS

- In Java ...

```
List<String> strings = new ArrayList<>();
List<CharSequence> charSequences = strings;
```

```
List<String> strings = new ArrayList<>();
List<CharSequence> charSequences = strings;
```

```
List<String> strings = new ArrayList<>();
List<? extends CharSequence> charSequences = strings;

public <T> void addAll(List<? super T> to,
                       List<? extends T> from) {
    to.addAll(from);
}
```

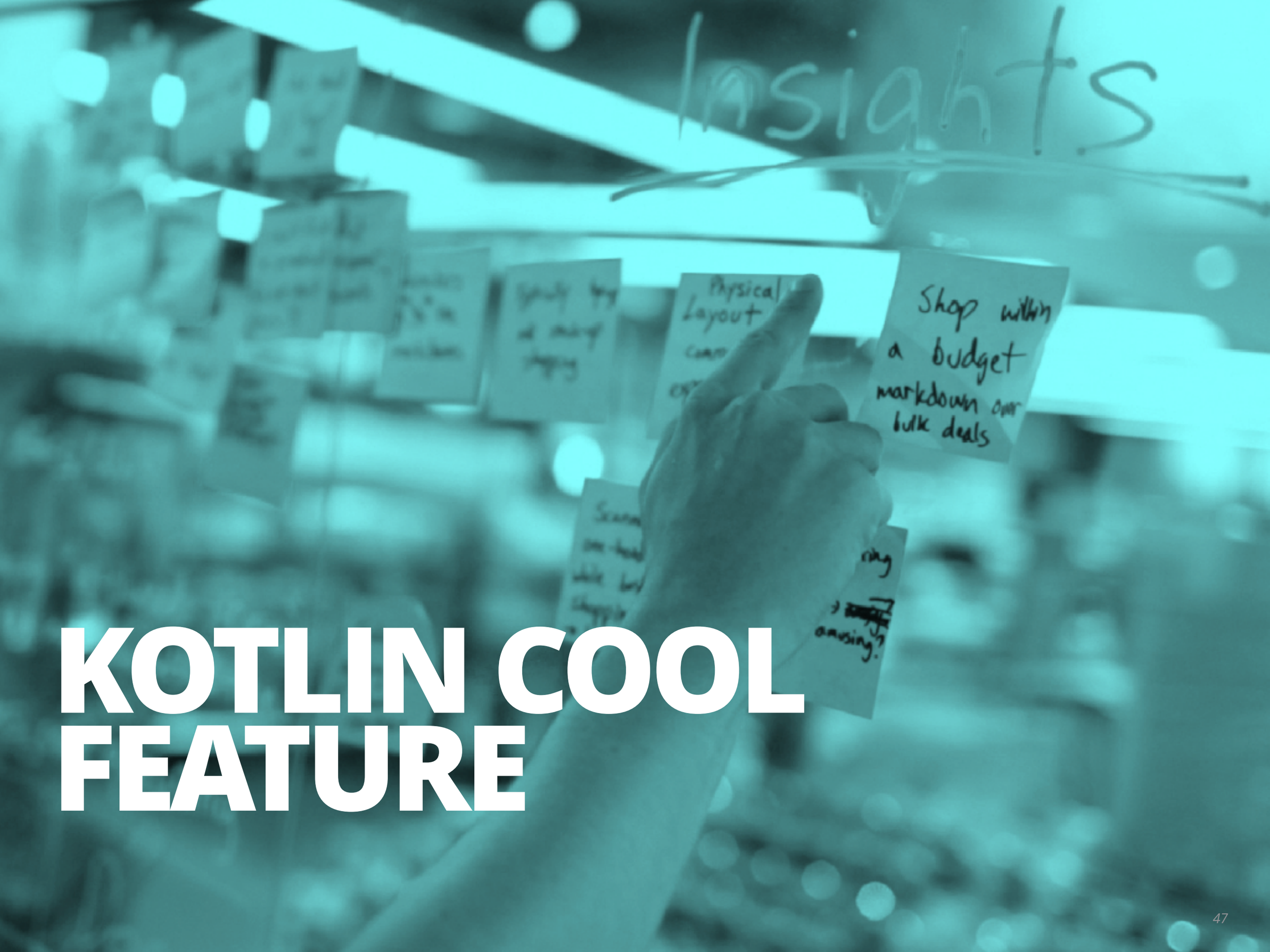
- In Kotlin ...

```
fun <T> MutableList<T>.addAll(from: List<T>) = addAll(from)
```


```
fun <T> MutableList<T>.addAll(from: List<T>) = addAll(from)
```

总结一下

- ~~繁琐的语法，低级的API~~
- 提供高级的语法例如data, object等
- ~~随时可能出现的null pointer问题~~
- 有optional的对象
- ~~默认可变的变量~~
- 有val，late-initialized，lazy， collections支持
- ~~各种各样的util类~~
- Extension帮了大忙
- ~~混乱的泛型~~
- Generics就是简单好用



KOTLIN COOL FEATURE

“Inline function”

INLINE

```
fun addAccount(username: String, password: String): String {
    val t1 = System.currentTimeMillis()
    val token = login(username, password)
    val t2 = System.currentTimeMillis()
    println("login ${t2 - t1}ms")
    val user = fetchUserDetail(token)
    println("fetch user detail ${System.currentTimeMillis() - t2}ms")
    return user
}
```

INLINE

```
fun <T> measureTime(msg: String, body: () -> T): T {
    val start = System.currentTimeMillis()
    val result = body()
    println("$msg ${System.currentTimeMillis() - start}ms")
    return result
}

fun addAccount(username: String, password: String): String {
    val token = measureTime("login") {
        login(username, password)
    }
    return measureTime("fetch user detail") {
        fetchUserDetail(token)
    }
}
```

INLINE

```
inline fun <T> measureTime(msg: String, body: () -> T): T {
    val start = System.currentTimeMillis()
    val result = body()
    println("$msg ${System.currentTimeMillis() - start}ms")
    return result
}

fun addAccount(username: String, password: String): String {
    val token = measureTime("login") {
        login(username, password)
    }
    return measureTime("fetch user detail") {
        fetchUserDetail(token)
    }
}
```

INLINE

```
fun addAccount(username: String, password: String): String {
    val t1 = System.currentTimeMillis()
    val token = login(username, password)
    val t2 = System.currentTimeMillis()
    println("login ${t2 - t1}ms")
    val user = fetchUserDetail(token)
    println("fetch user detail ${System.currentTimeMillis() - t2}ms")
    return user
}
```

INLINE

- reified

```
inline fun <reified T> fromJson(json: String): T {
    return Gson().fromJson(json, T::class.java)
}
```

```
fun demoReified(): String = fromJson("test")
```

- inline properties

```
inline var str: String
    get() {
        return ""
    }
    set(v) {}
```

Sealed Class

SEALED CLASS

```
open class Expr

data class Const(val number: Int) : Expr()
data class Add(val e1: Expr, val e2: Expr) : Expr()
data class Minus(val e1: Expr, val e2: Expr) : Expr()
data class Multiple(val e1: Expr, val e2: Expr) : Expr()
data class Divide(val e1: Expr, val e2: Expr) : Expr()

fun Expr.eval(): Const =
    when (this) {
        is Const -> this
        is Add -> Const(e1.eval().number + e2.eval().number)
        is Minus -> Const(e1.eval().number - e2.eval().number)
        is Multiple -> Const(e1.eval().number * e2.eval().number)
        is Divide -> Const(e1.eval().number / e2.eval().number)
        else -> throw UnsupportedOperationException("Unsupported expr")
    }
```

SEALED CLASS

```
sealed class Expr

data class Const(val number: Int) : Expr()
data class Add(val e1: Expr, val e2: Expr) : Expr()
data class Minus(val e1: Expr, val e2: Expr) : Expr()
data class Multiple(val e1: Expr, val e2: Expr) : Expr()
data class Divide(val e1: Expr, val e2: Expr) : Expr()

fun Expr.eval(): Const =
    when (this) {
        is Const -> this
        is Add -> Const(e1.eval().number + e2.eval().number)
        is Minus -> Const(e1.eval().number - e2.eval().number)
        is Multiple -> Const(e1.eval().number * e2.eval().number)
        is Divide -> Const(e1.eval().number / e2.eval().number)
    }
```

Delegated properties

DELEGATED PROPERTIES

```
class Delegate {
    operator fun getValue(thisRef: Any?, property: KProperty<*>): String {
        return "reading $thisRef ${property.name}"
    }

    operator fun setValue(thisRef: Any?, property: KProperty<*>, value: String) {
        println("modifying $thisRef ${property.name} to $value")
    }
}
```

```
object Main {
    var test: String by Delegate()
    @JvmStatic fun main(args: Array<String>) {
        test = "a"
        println(test)
    }
}
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/bin/java ...
modifying com.thoughtworks.china.mobile.delegateproperty.Main@38af3868 test to a
reading com.thoughtworks.china.mobile.delegateproperty.Main@38af3868 test

Process finished with exit code 0
```

DELEGATED PROPERTIES

```
class Person(sp: SharedPreferences) {
    var age: Int by sharedPreferenceDelegate(sp, "age")
    var height: Float by sharedPreferenceDelegate(sp, "height")
}
```

Coroutines

CALLBACK

```
fab.setOnClickListener {
    toast("You clicked fab")
    object : AsyncTask<Unit, Unit, String>() {

        override fun doInBackground(vararg params: Unit): String {
            Thread.sleep(5000)
            return "this string is generated from background thread"
        }

        override fun onPostExecute(str: String) {
            textView.text = str
        }
    }.execute()
}
```

COROUTINE

```
fab.setOnClickListener {
    UIContext {
        toast("You clicked fab")
        val str = async(CommonPool) {
            delay(5000)
            "this string is generated from background thread"
        }
        textView.text = str.await()
    }
}
```

总结一下

- Inline Function 减少运行开销
- Sealed Class 限制类的层级
- Delegate 使得代码更简洁
- Coroutines 提高异步代码可读性

KOTLIN IN ANDROID

VIEWHOLDER IN JAVA

```
class ViewHolder extends RecyclerView.ViewHolder {
    private ImageView image;
    private TextView textView;
    private CheckBox checkbox;
    private EditText editText;
    private Button button;

    public ViewHolder(View itemView) {
        super(itemView);
        image = (ImageView) itemView.findViewById(R.id.image);
        textView = (TextView) itemView.findViewById(R.id.textView);
        checkbox = (CheckBox) itemView.findViewById(R.id.checkbox);
        editText = (EditText) itemView.findViewById(R.id.editText);
        button = (Button) itemView.findViewById(R.id.button);
    }

    public void populate() {
        image.setImageDrawable(null);
        textView.setText(" ");
        checkbox.setChecked(false);
        editText.setHint(" ");
        button.setOnClickListener(...);
    }
}
```

VIEWHOLDER IN KOTLIN

```
class ViewHolder extends RecyclerView.ViewHolder {
        fun populate() {
            itemView.apply {
                imageView.setImageDrawable(null)
                textView.text = ""
                checkbox.isChecked = false
                editText.hint = ""
                button.setOnClickListener { }
            }
        }
}
```

KOTLIN'S REFERENCE

引入KOTLIN

```
buildscript {
    ...
    ext.kotlin_version = '1.1.1'
    dependencies {
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
```

```
apply plugin: 'kotlin-android'
dependencies {
    compile "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
```

已知问题

- 与mockito的兼容性
- 静态代码检查工具（findbugs, PMD ...) 兼容性
- 关键词和操作符语义变化（equals, ==)

https://github.com/ChinaMobileLab/java-vs-kotlin

谢谢

有问题请联系

张帅 & 王智勇

shuaiz@thoughtworks.com

ThoughtWorks®