ANDR*ID *ШИБКИ ПР*ЕКТИР*ВАНИЯ API

✌️

Абакар Магомедов

Android-Techlead

Альфа-Банк

АБАКАР МАГОМЕДОВ

- Android Techlead AlfaBank
- Пытаюсь писать статьи
- Участвовал в ПК Android Podlodka Crew
- Выступал с докладом Mobius

ПЛАН ДОКЛАДА

- Обсудим, как перед нами встал вопрос создания библиотек
- Какие ошибки были нами допущены при проектировании их публичного API
- Как мы решали эти проблемы
- Сделаем выводы

ВСТУПЛЕНИЕ:
API VS ABI

ФАКТЫ

API — application programming interface, например исходный код нашего приложения

ABI — application binary interface, например, класс файлы, которые были сгенерированы на основе нашего исходного кода

Почти всегда ABI напрямую связано с API, так как ABI — это скомпилированная форма API

НО ЭТА СВЯЗЬ РАБОТАЕТ НЕ ТАК, КАК МЫ ОЖИДАЕМ:

Source code:

```
public List<String> getStrings() {}

public List<Integer> getInts() {}
```

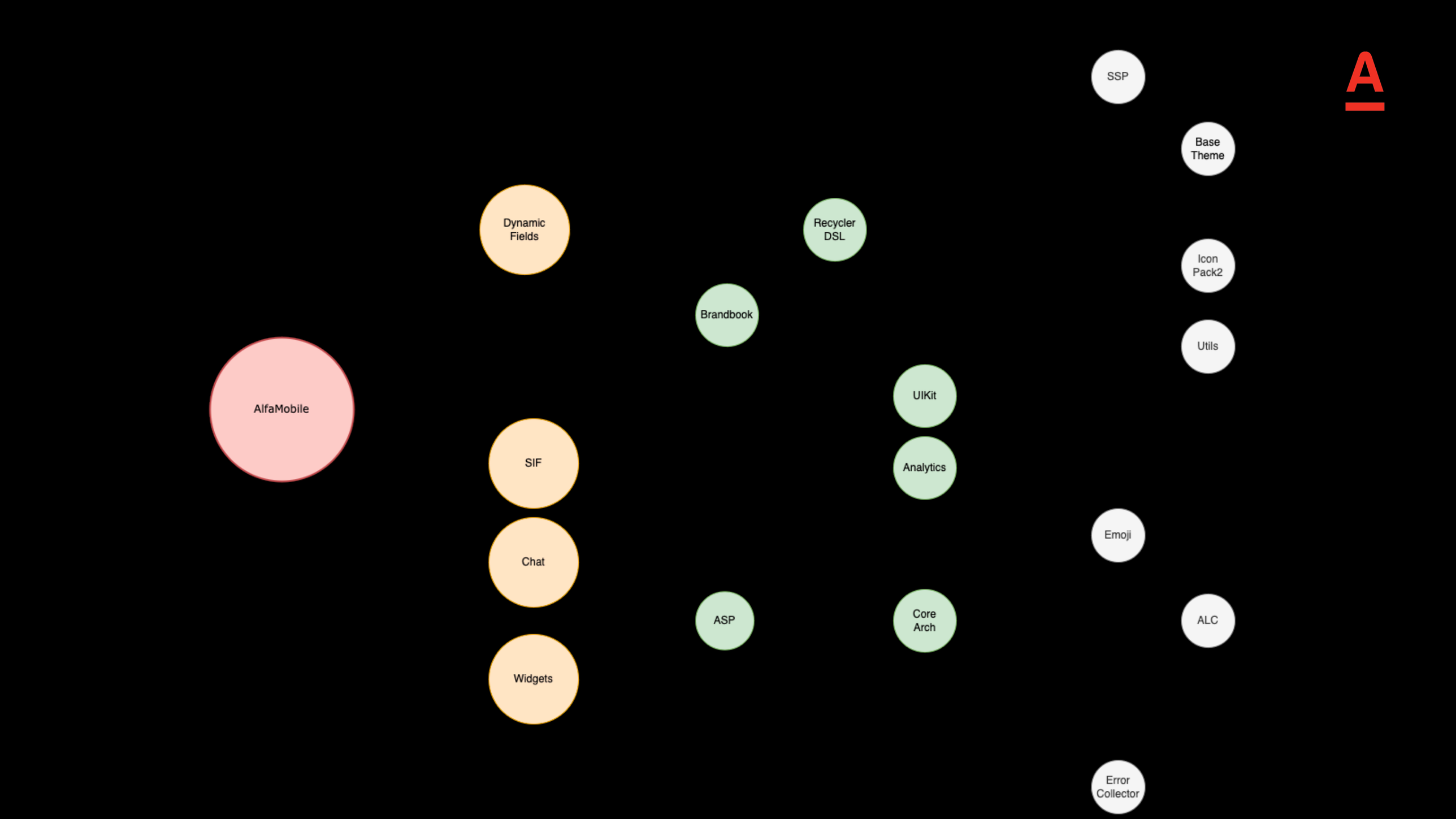
Class file:

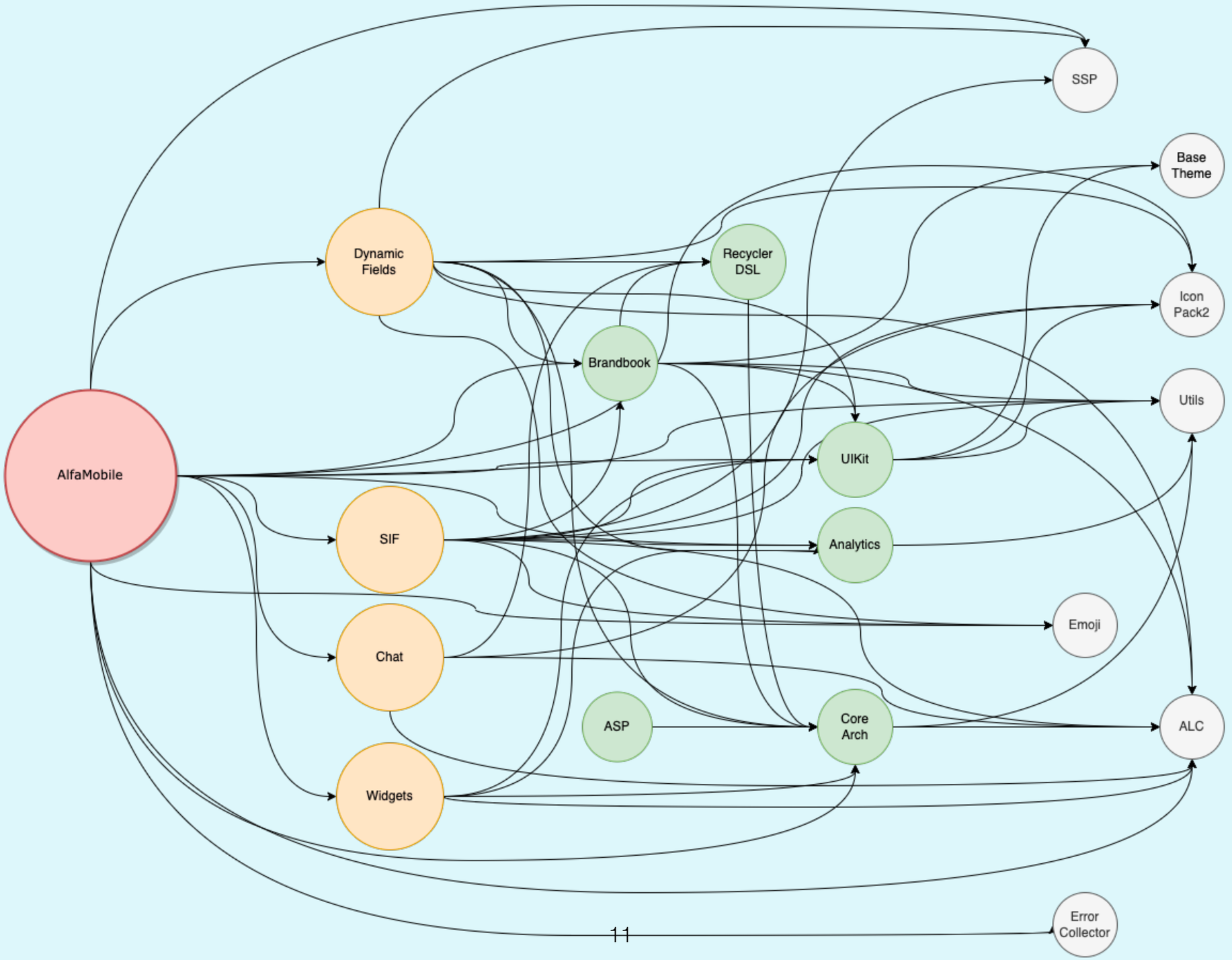
```
public List getStrings() {}

public List getInts() {}
```

А ТЕПЕРЬ ПРОДОЛЖИМ:

СХЕМА ЗАВИСИМОСТЕЙ НАШИХ БИБЛИОТЕК


AlfaMobile
Dynamic Fields
SIF
Chat
Widgets
Brandbook
ASP
Recycler DSL
UIKit
Analytics
Core Arch
SSP
Base Theme
Icon Pack2
Utils
Emoji
ALC
Error Collector


SSP
Base Theme
Dynamic Fields
Recycler DSL
Icon Pack2
Brandbook
Utils
AlfaMobile
UIKit
SIF
Analytics
Emoji
Chat
ASP
Core Arch
ALC
Widgets
Error Collector

ОШИБКА 0:

```
enum class DataViewVerticalPadding {

    DEFAULT,
    NOT_DEFAULT,
    JAK_FRESCO;

    abstract fun applyTo()
}
```

ФАКТЫ

Энам используется только внутри библиотеки, но по исходному коду мы никак это понять не можем

Если мы добавим новую вариацию энама, это будет ломающее API изменение

```
internal enum class DataViewVerticalPadding {

    DEFAULT,
    NOT_DEFAULT,
    JAK_FRESCO;

    abstract fun applyTo()
}
```

PUBLISHED API

```
class RecyclerConfig {

    internal val rows: MutableList<BaseViewHolderCreator> = mutableListOf()

    inline fun row(rowViewHolderCreator: BaseViewHolderCreator, action: () -> Unit) {
        action.invoke()
        rows.add(rowViewHolderCreator)
    }

}
```

```
class RecyclerConfig {

    internal val rows: MutableList<BaseViewHolderCreator> = mutableListOf()

    inline fun row(rowViewHolderCreator: BaseViewHolderCreator, action: () -> Unit) {
        action.invoke()
        rows.add(rowViewHolderCreator)
    }

}
```

Public-API inline function cannot access non-public-API 'internal final val rows: MutableList<BaseViewHolderCreator> defined in ru.alfabank.mobile.android.recycler.RecyclerConfig'

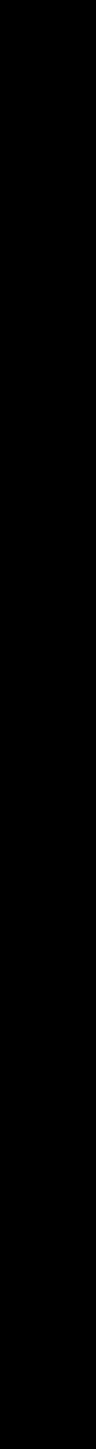
Make 'row' internal ⌥⇧⏎ More actions... ⌥⏎

```
class RecyclerConfig {

    @PublishedApi
    internal val rows: MutableList<BaseViewHolderCreator> = mutableListOf()

    inline fun row(rowViewHolderCreator: BaseViewHolderCreator, action: () -> Unit) {
        action.invoke()
        rows.add(rowViewHolderCreator)
    }
}
```

* this allows to call that declaration from public inline functions;
* the declaration becomes effectively public, and this should be considered with respect to binary compatibility maintaining.



```
kotlin {
    // for strict mode
    explicitApi()
    // or
    explicitApi = ExplicitApiMode.Strict

    // for warning mode
    explicitApiWarning()
    // or
    explicitApi = ExplicitApiMode.Warning
}
```

ФАКТЫ

Есть еще один вариант. Просто делать пакет internal в вашей библиотеке и располагать в нем сущности.

ОШИБКА 0:
~~РАЗРЕШАТЬ ВСЕ ЧТО НЕ ЗАПРЕЩЕНО~~

РЕШЕНИЕ 0:
ЗАПРЕЩАТЬ ВСЁ, ЧТО НЕ РАЗРЕШЕНО

ОШИБКА 1:

```
data class MarkdownModel(
    val size: Size? = null
)
```

RECAP

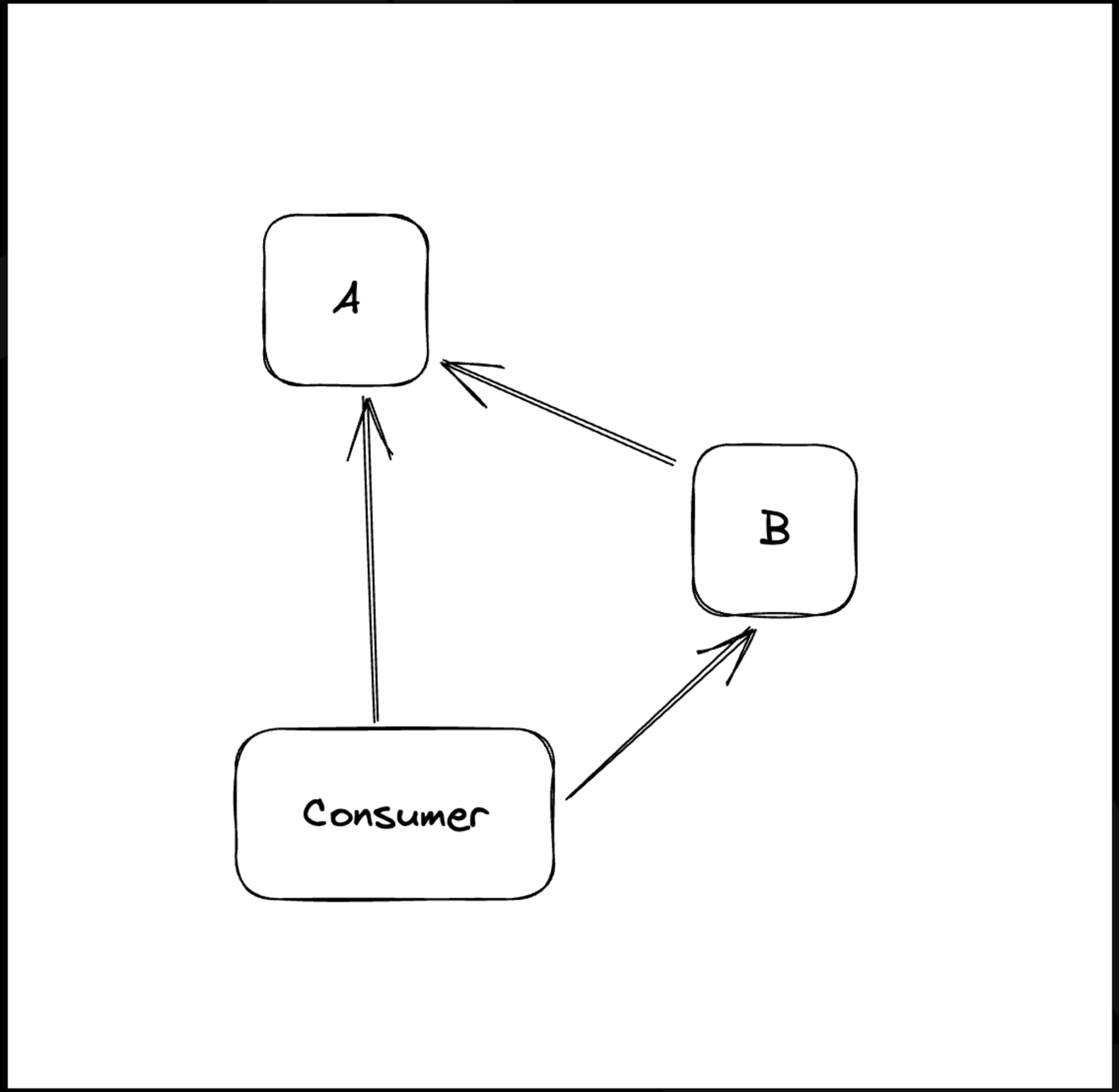
Компоненты находятся в состоянии постоянного развития и добавления новой функциональности

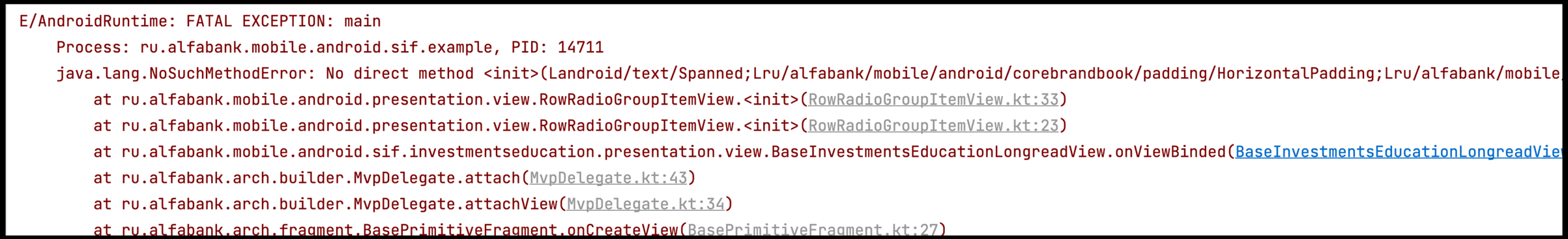
Соответственно в их модели добавляются новые параметры

ЕСТЬ ЛИ ПРОБЛЕМА С ТАКИМ ИЗМЕНЕНИЕМ ?

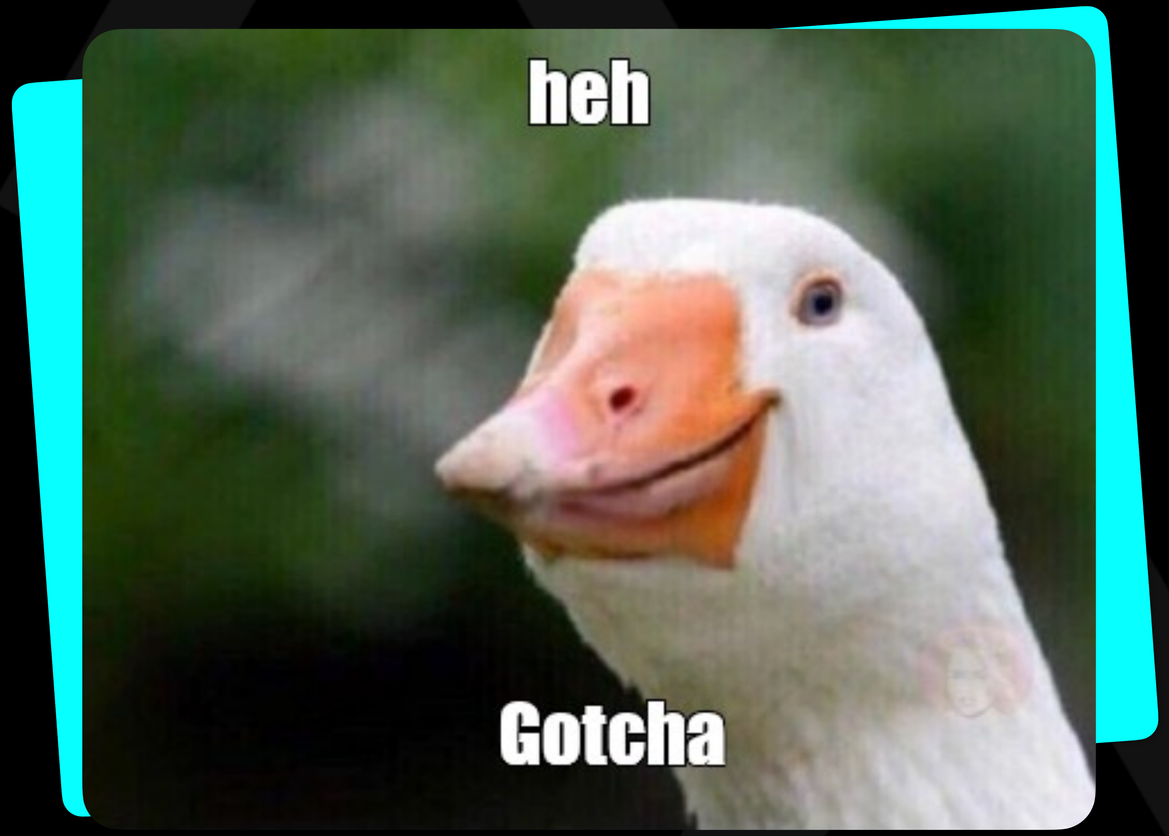
```
data class MarkdownModel(
    val size: Size? = null,
    val jackFresco: String? = null
)
```

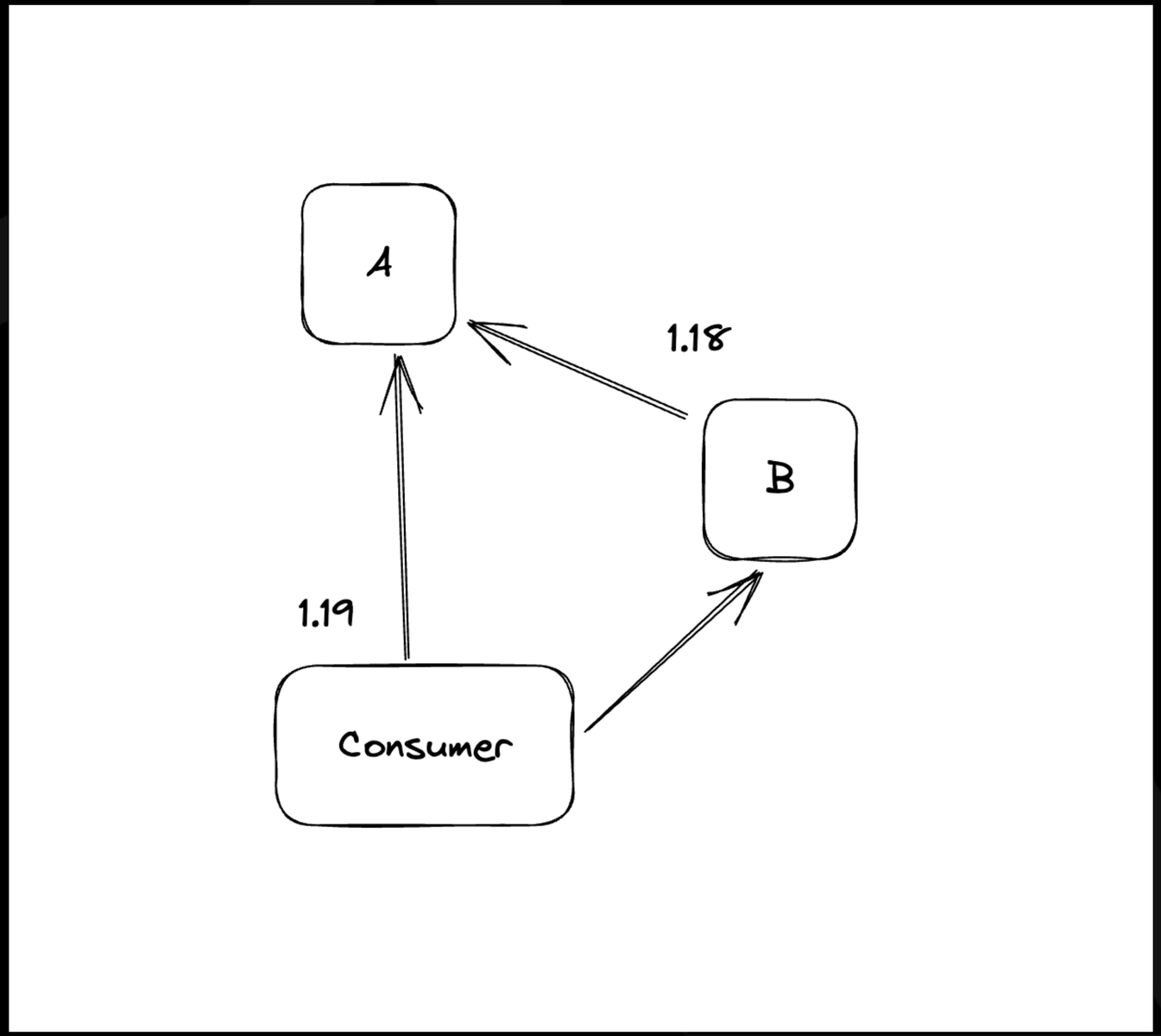
```
init {
    val k = MarkdownModel(size = null)
    println(k.copy())
}
```


A
B
Consumer


E/AndroidRuntime: FATAL EXCEPTION: main
Process: ru.alfabank.mobile.android.sif.example, PID: 14711
java.lang.NoSuchMethodError: No direct method <init>(Landroid/text/Spanned;Lru/alfabank/mobile/android/corebrandbook/padding/HorizontalPadding;Lru/alfabank/mobile
at ru.alfabank.mobile.android.presentation.view.RowRadioGroupItemView.<init>(RowRadioGroupItemView.kt:33)
at ru.alfabank.mobile.android.presentation.view.RowRadioGroupItemView.<init>(RowRadioGroupItemView.kt:23)
at ru.alfabank.mobile.android.sif.investmentseducation.presentation.view.BaseInvestmentsEducationLongreadView.onViewBinded(BaseInvestmentsEducationLongreadVie
at ru.alfabank.arch.builder.MvpDelegate.attach(MvpDelegate.kt:43)
at ru.alfabank.arch.builder.MvpDelegate.attachView(MvpDelegate.kt:34)
at ru.alfabank.arch.fragment.BasePrimitiveFragment.onCreateView(BasePrimitiveFragment.kt:27)


Process: ru.alfabank.mobile.android.sif.example, PID: 14711
java.lang.NoSuchMethodError: No direct method <init>(Landroi


heh
Gotcha


A
1.18
B
1.19
Consumer

```
data class MarkdownModel
@JvmOverloads constructor(
    val size: Size? = null,
    val jackFresco: String? = null
)
```

```
    --------- beginning of crash
E/AndroidRuntime: FATAL EXCEPTION: main
    Process: ru.alfabank.mobile.android.sif.example, PID: 27110
    java.lang.NoSuchMethodError: No static method copy$default(Lru/a
        at ru.alfabank.mobile.android.presentation.view.RowRadioGrou
```

```java
@NotNull
public final MarkdownModel copy(@Nullable Size size) {
   return new MarkdownModel(size);
}

// $FF: synthetic method
public static MarkdownModel copy$default(MarkdownModel var0, Size var1, String var2, Object var3) {
   if ((var2 & 1) != 0) {
      var1 = var0.size;
   }

   return var0.copy(var1);
}
```

```
@NotNull
public final MarkdownModel copy(@Nullable Size size, @Nullable String jackFresco) {
   return new MarkdownModel(text, size, jackFresco);
}

// $FF: synthetic method
public static MarkdownModel copy$default(MarkdownModel var0, Size var1, String var2, int var3, Object var4) {
   if ((var3 & 1) != 0) {
      var1 = var0.size;
   }

   if ((var3 & 2) != 0) {
      var2 = var0.jackFresco;
   }
   return var0.copy(var1, var2);
}
```

```
Execution failed for task ':brandbook:core_ui_brandbook:apiCheck'.
> API check failed for project core_ui_brandbook.
  --- /Users/abocha/StudioProjects/shared-libraries/brandbook/core_ui_brandbook/api/core_ui_brandbook.api
  +++ /Users/abocha/StudioProjects/shared-libraries/brandbook/core_ui_brandbook/build/api/core_ui_brandbook.api
  @@ -9107,12 +9107,15 @@
   public final class ru/alfabank/mobile/android/coreuibrandbook/mapview/MarkdownModel {
     public fun <init> (Landroid/text/Spanned;)V
     public fun <init> (Landroid/text/Spanned;Lru/alfabank/mobile/android/corebrandbook/size/Size;)V
  - public synthetic fun <init> (Landroid/text/Spanned;Lru/alfabank/mobile/android/corebrandbook/size/Size;ILkotlin/jvm/internal/DefaultConstructorMarker;)V
  + public fun <init> (Landroid/text/Spanned;Lru/alfabank/mobile/android/corebrandbook/size/Size;Ljava/lang/String;)V
  + public synthetic fun <init> (Landroid/text/Spanned;Lru/alfabank/mobile/android/corebrandbook/size/Size;Ljava/lang/String;ILkotlin/jvm/internal/DefaultConstructorMarker;)V
     public final fun component1 ()Landroid/text/Spanned;
     public final fun component2 ()Lru/alfabank/mobile/android/corebrandbook/size/Size;
  - public final fun copy (Landroid/text/Spanned;Lru/alfabank/mobile/android/corebrandbook/size/Size;)Lru/alfabank/mobile/android/coreuibrandbook/mapview/MarkdownModel;
  - public static synthetic fun copy$default (Lru/alfabank/mobile/android/coreuibrandbook/mapview/MarkdownModel;Landroid/text/Spanned;Lru/alfabank/mobile/android/corebrandbook/size/Size;ILjava/lang/(
  + public final fun component3 ()Ljava/lang/String;
  + public final fun copy (Landroid/text/Spanned;Lru/alfabank/mobile/android/corebrandbook/size/Size;Ljava/lang/String;)Lru/alfabank/mobile/android/coreuibrandbook/mapview/MarkdownModel;
  + public static synthetic fun copy$default (Lru/alfabank/mobile/android/coreuibrandbook/mapview/MarkdownModel;Landroid/text/Spanned;Lru/alfabank/mobile/android/corebrandbook/size/Size;Ljava/lang/S
     public fun equals (Ljava/lang/Object;)Z
  + public final fun getJackFresco ()Ljava/lang/String;
     public final fun getSize ()Lru/alfabank/mobile/android/corebrandbook/size/Size;
     public final fun getText ()Landroid/text/Spanned;
     public fun hashCode ()I
```

https://www.youtube.com/watch?v=-7qji4vZkGw

RECAP

Консьюмеры, которые будут вызывать метод copy получат NoSuchMethodError

Разработчики зачастую не сильно разбираются чем отличаются изменения ломающие api и изменения, которые ломают бинарную совместимость

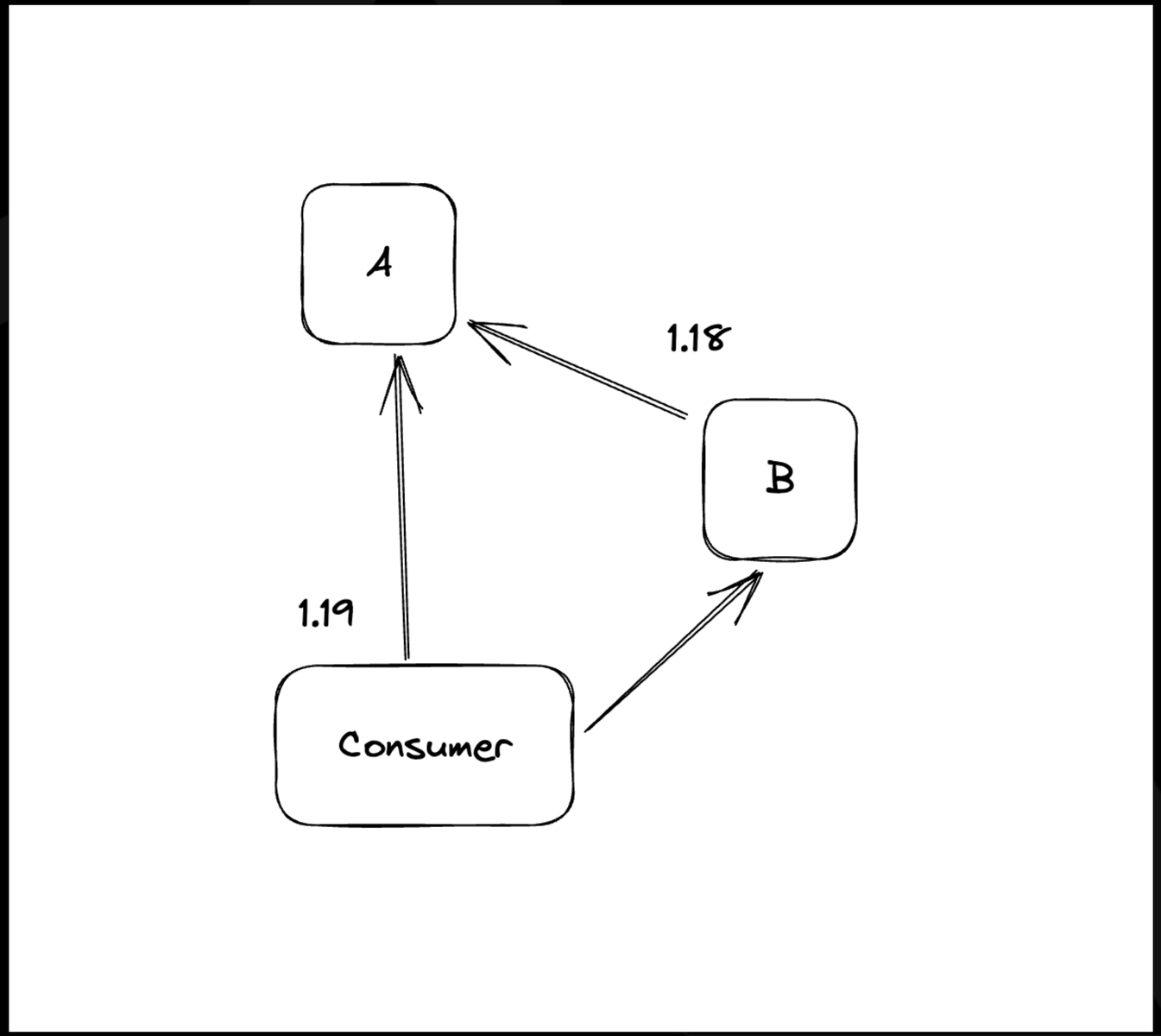
```
interface BaseAnalyticsEvents {

    val ERROR: String
        get() = "Error"
    val SUCCESS: String
        get() = "Success"
}
```

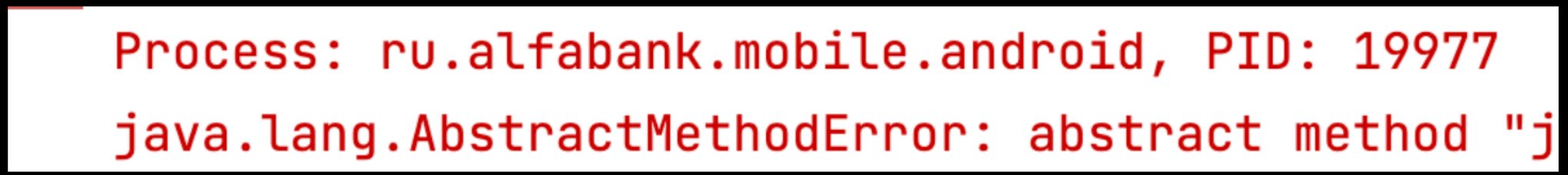
```
interface BaseAnalyticsEvents {

    val ERROR: String
        get() = "Error"
    val SUCCESS: String
        get() = “Success”
    val LONG_TAP: String
        get() = “Long Tap”
}
```

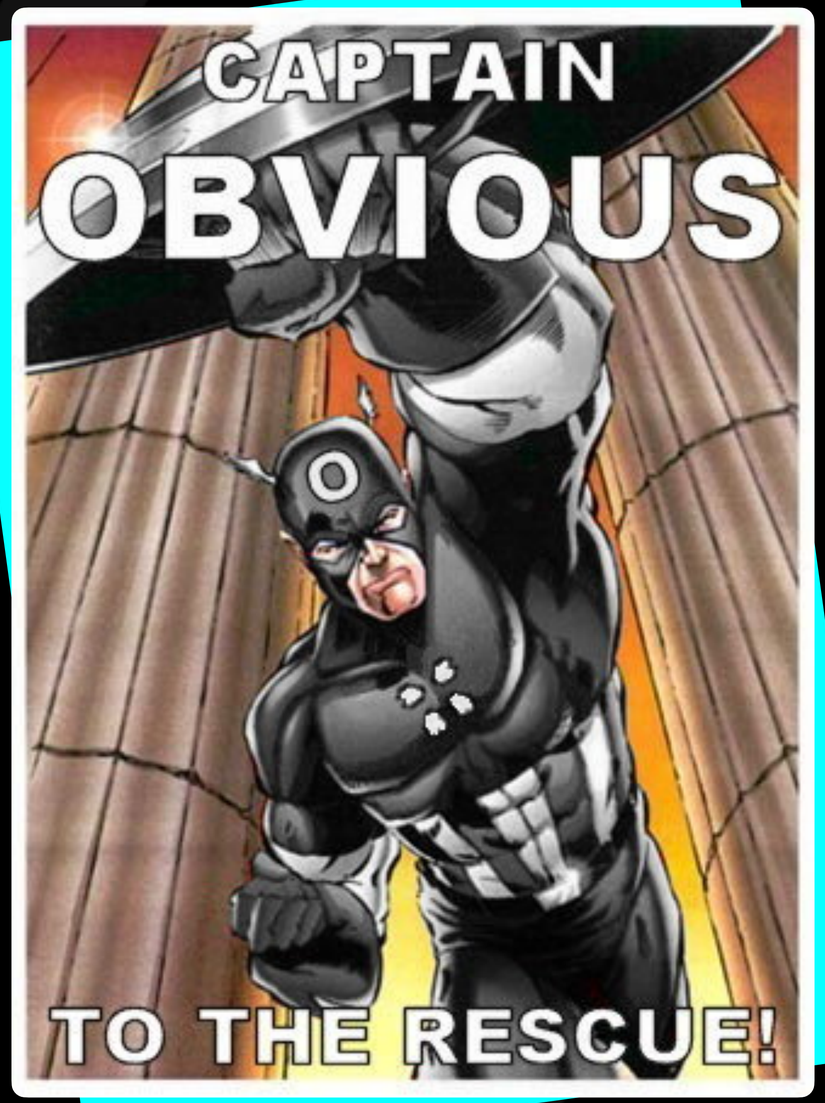

fun trackCardItemClick() {
sendEvent(
action = LONG_TAP,
label = "Card Item"
)
}


A
1.18
B
1.19
Consumer


D Shutting down VM
E FATAL EXCEPTION: main
Process: ru.alfabank.mobile.android, PID: 19977
java.lang.AbstractMethodError: abstract method "java.lang.String ru.alfabank.mobile.android.analytics.dto.BaseAnalyticsEvents.getLONG_TAP()"
at ru.alfabank.mobile.android.investmentsmain.analytics.InvestmentsMainEvents.trackCardItemClick(InvestmentsMainEvents.kt:28)
at ru.alfabank.mobile.android.investmentsmain.presentation.presenter.InvestmentsMainPresenter.loadWidgets(InvestmentsMainPresenter.kt:28)
at ru.alfabank.mobile.android.investmentsmain.presentation.presenter.InvestmentsMainPresenter.onViewCreated(InvestmentsMainPresenter.kt:24)
at ru.alfabank.arch.builder.MvpDelegate.attach(MvpDelegate.kt:44)


Process: ru.alfabank.mobile.android, PID: 19977
java.lang.AbstractMethodError: abstract method "j

ОШИБКА 1:
~~НЕ ЗАДУМЫВАТЬСЯ О БИНАРНОЙ СОВМЕСТИМОСТИ~~


CAPTAIN
OBVIOUS
TO THE RESCUE!

RECAP

Но все не так просто, мы уже используем очень много компонентов, которые завязаны на data классы

Мы используем плагин, который подскажет разработчику, ломает ли его изменение бинарную совместимость

Мы встроили прогон этого плагина на наш CI

РЕШЕНИЕ 1:
ЗАДУМЫВАТЬСЯ О БИНАРНОЙ СОВМЕСТИМОСТИ

ОШИБКА 2:

```
open class DynamicDataRow(
    @SerializedName("id")
    open val id: String,
    @SerializedName("label")
    open var label: String?
)
```

PROBLEMS

Наружу нашей библиотеки торчит api gson

Это связывает консьюмеров, так как в нашей библиотеке очень много тайп адаптеров, написанных для gson

Приходится делать бриджи

PROBLEMS

В нашем публичном api есть зависимость на библиотеку десериализации

Еще хуже, если у нас еще будет зависимость и на сетевой клиент

Лучше описать свой контракт, а ответственность за его поддержку будет лежать на стороне консьюмера

```
class WidgetDataTypeAdapter
    : JsonDeserializer<Map<WidgetDataType, Any>>, JsonSerializer<Map<WidgetDataType, Any>> {

    private val widgetTypes = WidgetDataType.values().map { it.serverKey }
    private val gson = Gson()

    override fun deserialize(
        json: JsonElement?,
        typeOfT: Type,
        context: JsonDeserializationContext
    ): Map<WidgetDataType, Any> {
        val jsonObject = json?.asJsonObject
        removeUnknownTypes(jsonObject)
        val map = context.deserialize<Map<WidgetDataType, Any>>(jsonObject, typeOfT).toMutableMap()
        deserializeKeyToMap(json, map, context, WidgetDataType.INSETS, WidgetInsets::class.java)
        deserializeKeyToMap(json, map, context, WidgetDataType.TITLE_INSETS, WidgetInsets::class.java)
        deserializeKeyToMap(json, map, context, WidgetDataType.PADDINGS, EdgeOffsetsDto::class.java)
        deserializeKeyToMap(json, map, context, WidgetDataType.TITLE_PADDINGS, EdgeOffsetsDto::class.java)
        return map
    }

    private fun <T: Any> deserializeKeyToMap(
        json: JsonElement?,
        map: MutableMap<WidgetDataType, Any>,
        context: JsonDeserializationContext,
        key: WidgetDataType,
        className: Class<T>
    ) {
        if (map.containsKey(key)) {
            val jsonObject = json?.asJsonObject?.getAsJsonObject(key.serverKey)
            map[key] = context.deserialize<T>(jsonObject, className)
        }
    }

    private fun removeUnknownTypes(jsonObject: JsonObject?) {
        val iterator = jsonObject?.entrySet()?.iterator()
        if (iterator != null) {
            deleteUnknownTypes(iterator)
        }
    }
```

RECAP

В публичном api должно быть минимум внешних зависимостей

В планах автоматизация, через кастомные линт рулы

ОШИБКА 2:
~~ДЕРЖАТЬ В ПУБЛИЧНОМ API СТОРОННИЕ ЗАВИСИМОСТИ~~

РЕШЕНИЕ 2:
НЕ ДЕРЖАТЬ В ПУБЛИЧНОМ API СТОРОННИЕ ЗАВИСИМОСТИ

ОШИБКА 2.1:

```
class Banner {

    fun inflateViews(layoutIds: List<Int>) {
         val inflater = LayoutInflater.from(this.context)
         innerView = inflater.inflate(layoutIds[0], null)
         containerView.removeAllViews()
         containerView.addView(innerView)
     }

    fun populate(model: BannerWrapperModel<T>) {
         with(model) {
             val innerModel = innerViewModel
             populateInnerView(innerModel)
         }
     }
}
```

```
class Banner {

    fun inflateViews(layoutIds: List<Int>) {
         val inflater = LayoutInflater.from(this.context)
         innerView = inflater.inflate(layoutIds[0], null)
         containerView.removeAllViews()
         containerView.addView(innerView)
     }

    fun populate(mo    l:    nerWrapperModel<T>) {
         with(model)
             val inn      del = innerViewModel
             popul      rView(innerModel)
         }
     }
}
```

RECAP

Такое api дает нарушить инварианты

С точки зрения пользователя мне приходится залезать в имплементацию, чтобы понять как пользоваться библиотекой

```
class BannerWrapper {

    private fun inflateViews(layoutIds: List<Int>) {
        val inflater = LayoutInflater.from(this.context)
        innerView = inflater.inflate(layoutIds[0], null)
        containerView.removeAllViews()
        containerView.addView(innerView)
    }

    override fun populate(model: BannerWrapperModel<T>) {
        if (!isHierarchyInitialized) {
            inflateViews(model.layoutIds)
        }
        with(model) {
            val innerModel = innerViewModel
            populateInnerView(innerModel)
        }
    }
}
```

```
class BannerWrapper {

    private fun inflateViews(layoutIds: List<Int>) {
        val inflater = LayoutInflater.from(this.context)
        innerView = inflater.inflate(layoutIds[0], null)
        containerView.removeAllViews()
        containerView.addView(innerView)
    }

    override fun populate(model: BannerWrapperModel<T>) {
        if (!isHierarchyInitialized) {
            inflateViews(model.layoutIds)
        }
        with(model) {
            val innerModel = innerViewModel
            populateInnerView(innerModel)
        }
    }
}
```

ОШИБКА 2.1:
~~ПОЗВОЛЯТЬ НАРУШАТЬ ИНВАРИАНТЫ~~

РЕШЕНИЕ 2.1:
СТАРАТЬСЯ НЕ ПОЗВОЛЯТЬ НАРУШАТЬ ИНВАРИАНТЫ

РЕШЕНИЕ 2.1:
А ЕСЛИ РАЗРАБУ НЕ НАДО ЛЕЗТЬ В БИБЛИОТЕКУ ВООБЩЕ ИДЕАЛЬНО

ОШИБКА 3:

```
class ServerDrivenMapper constructor(
    private val buttonMapper: ButtonViewMapper
)
```

```
fun mapItem(
    layoutElement: LayoutElement,
    serverDrivenActionDelegate: ServerDrivenActionDelegate? = null
): UIComponentModel? {

    return when (content) {
        is ButtonModelField -> buttonMapper.map(content)
        else -> null
    }
}
```

ВСЕ ЛИ В ПОРЯДКЕ С ЭТИМ ИЗМЕНЕНИЕМ?

```
class ServerDrivenMapper constructor(
    private val buttonMapper: ButtonViewMapper,
    private val textMapper: TextViewMapper
)
```

PROBLEMS

Каждый раз когда мы добавляем поддержку нового компонента нам приходится делать ломающее изменение

Это выглядит максимально странно, мы просто добавляем что-то новое, но при этом ломаем бинарную совместимость

RECAP

- Нельзя проектировать публичное api так, что невозможно добавить новую функциональность не ломая обратную совместимость

- В примере выше от проблему можно было изначально избежать и есть несколько способов

```
class ServerDrivenMapper {

    private val mappers = mutableMapOf<ServerDrivenType, Mapper<LayoutElement, UIComponentModel>>()

    init {
        mappers[ServerDrivenType.BUTTON] = ButtonViewMapper()
    }

    fun registerMapper(type: ServerDrivenType, mapper: Mapper<LayoutElement, UIComponentModel>) {
        mappers[type] = mapper
    }

    fun mapItem(layoutElement: LayoutElement): UIComponentModel? {
        return mappers[layoutElement.serverDrivenType]?.map(layoutElement)
    }
}



interface Mapper<in LayoutElement, out UiComponentModel> {
    fun map(element: LayoutElement): UiComponentModel
}
```

```
class ServerDrivenMapper {

    private val mappers = mutableMapOf<ServerDrivenType, Mapper<LayoutElement, UIComponentModel>>()

    init {
        mappers[ServerDrivenType.BUTTON] = ButtonViewMapper()
    }

    fun registerMapper(type: ServerDrivenType, mapper: Mapper<LayoutElement, UIComponentModel>) {
        mappers[type] = mapper
    }

    fun mapItem(layoutElement: LayoutElement): UIComponentModel? {
        return mappers[layoutElement.serverDrivenType]?.map(layoutElement)
    }
}
```

```
interface Mapper<in LayoutElement, out UiComponentModel> {
    fun map(element: LayoutElement): UiComponentModel
}
```

```
class ServerDrivenMapper {

    private val mappers = mutableMapOf<ServerDrivenType, Mapper<LayoutElement, UIComponentModel>>()

    init {
        mappers[ServerDrivenType.BUTTON] = ButtonViewMapper()
    }

    fun registerMapper(type: ServerDrivenType, mapper: Mapper<LayoutElement, UIComponentModel>) {
        mappers[type] = mapper
    }

    fun mapItem(layoutElement: LayoutElement): UIComponentModel? {
        return mappers[layoutElement.serverDrivenType]?.map(layoutElement)
    }
}



interface Mapper<in LayoutElement, out UiComponentModel> {
    fun map(element: LayoutElement): UiComponentModel
}
```

```
class ServerDrivenMapper {

    private val mappers = mutableMapOf<ServerDrivenType, Mapper<LayoutElement, UIComponentModel>>()

    init {
        mappers[ServerDrivenType.BUTTON] = ButtonViewMapper()
    }

    fun registerMapper(type: ServerDrivenType, mapper: Mapper<LayoutElement, UIComponentModel>) {
        mappers[type] = mapper
    }

    fun mapItem(layoutElement: LayoutElement): UIComponentModel? {
        return mappers[layoutElement.serverDrivenType]?.map(layoutElement)
    }
}



interface Mapper<in LayoutElement, out UiComponentModel> {
    fun map(element: LayoutElement): UiComponentModel
}
```

А СРАБОТАЕТ ТАК?

```
class ServerDrivenMapper constructor(
    private val buttonMapper: ButtonViewMapper,
    private val textMapper: TextViewMapper = TextViewMapper()
)
```

```
public final class ServerDrivenMapper {
    public ServerDrivenMapper(@NotNull ButtonViewMapper buttonViewMapper) {
        Intrinsics.checkNotNullParameter(buttonViewMapper, "buttonViewMapper");
        super();
    }
}
```

```
public final class ServerDrivenMapper {
   public ServerDrivenMapper2(@NotNull ButtonViewMapper buttonViewMapper, @NotNull TextViewMapper textViewMapper) {
      Intrinsics.checkNotNullParameter(buttonViewMapper, "buttonViewMapper");
      Intrinsics.checkNotNullParameter(textViewMapper, "textViewMapper");
      super();
   }

   // $FF: synthetic method
   public ServerDrivenMapper2(ButtonViewMapper var1, TextViewMapper var2, int var3, DefaultConstructorMarker var4) {
      if ((var3 & 2) != 0) {
         var2 = new TextViewMapper();
      }

      this(var1, var2);
   }
}
```

```
class ServerDrivenMapper @JvmOverloads constructor(
    private val buttonMapper: ButtonViewMapper,
    private val textMapper: TextViewMapper = TextViewMapper()
)
```

```
public final class ServerDrivenMapper {
   @JvmOverloads
   public ServerDrivenMapper(@NotNull ButtonViewMapper buttonViewMapper, @NotNull TextViewMapper textViewMapper) {
      Intrinsics.checkNotNullParameter(buttonViewMapper, "buttonViewMapper");
      Intrinsics.checkNotNullParameter(textViewMapper, "textViewMapper");
      super();
   }

   // $FF: synthetic method
   public ServerDrivenMapper(ButtonViewMapper var1, TextViewMapper var2, int var3, DefaultConstructorMarker var4) {
      if ((var3 & 2) != 0) {
         var2 = new TextViewMapper();
      }

      this(var1, var2);
   }

   @JvmOverloads
   public ServerDrivenMapper(@NotNull ButtonViewMapper buttonViewMapper) {
      this(buttonViewMapper, (TextViewMapper)null, 2, (DefaultConstructorMarker)null);
   }
}
```

```
public final class ServerDrivenMapper2 {
   @JvmOverloads
   public ServerDrivenMapper2(@NotNull ButtonViewMapper buttonViewMapper, @NotNull TextViewMapper textViewMapper) {
      Intrinsics.checkNotNullParameter(buttonViewMapper, "buttonViewMapper");
      Intrinsics.checkNotNullParameter(textViewMapper, "textViewMapper");
      super();
   }

   // $FF: synthetic method
   public ServerDrivenMapper2(ButtonViewMapper var1, TextViewMapper var2, int var3, DefaultConstructorMarker var4) {
      if ((var3 & 2) != 0) {
         var2 = new TextViewMapper();
      }

      this(var1, var2);
   }

    @JvmOverloads
    public ServerDrivenMapper2(@NotNull ButtonViewMapper buttonViewMapper) {
       this(buttonViewMapper, (TextViewMapper)null, 2, (DefaultConstructorMarker)null);
    }
}
```

ОШИБКА 3:
~~ПРОЕКТИРОВАНИЕ НЕ РАСШИРЯЕМОГО API~~

РЕШЕНИЕ 3:
ПРОЕКТИРОВАНИЕ РАСШИРЯЕМОГО API

БОНУСНАЯ ЧАСТЬ

ОШИБКА 4:

```kotlin
@MakeDataFactory
data class ButtonViewModel(
    override val size: ButtonSize = ButtonSize.LARGE,
    override val maxLines: TextMaxLinesStyle = TextMaxLinesStyle.ELLIPSIZE,
    override val isEnabled: Boolean = true
)
```

```
fun map(buttonModel: ButtonModelField): ButtonViewModel {
    with(buttonModel) {
        return ButtonViewModel(
            size = buttonSize?.transformedSize ?: ButtonSize.LARGE,
            maxLines = buttonMaxLinesStyle.transformedMaxLinesStyle,
            isEnabled = isEnabled ?: true
        )
    }
}
```

```
fun map(buttonModel: ButtonModelField): ButtonViewModel {
    with(buttonModel) {
        return ButtonViewModel(
            size = buttonSize?.transformedSize ?: ButtonSize.LARGE,
            maxLines = buttonMaxLinesStyle.transformedMaxLinesStyle,
            isEnabled = isEnabled ?: true
        )
    }
}
```

```
fun map(buttonModel: ButtonModelField): ButtonViewModel {
    with(buttonModel) {
        return buttonViewModelFactory(
            size = buttonSize?.transformedSize,
            maxLines = buttonMaxLinesStyle?.transformedMaxLinesStyle,
            isEnabled = isEnabled
        )
    }
}
```

```
fun map(buttonModel: ButtonModelField): ButtonViewModel {
    with(buttonModel) {
        return buttonViewModelFactory(
            size = buttonSize?.transformedSize,
            maxLines = buttonMaxLinesStyle?.transformedMaxLinesStyle,
            isEnabled = isEnabled,
        )
    }
}
```

```
val result = ButtonViewModel()
    val resolvedSize = size ?: result.size
    val resolvedMaxLines = maxLines ?: result.maxLines
    val resolvedIsEnabled = isEnabled ?: result.isEnabled

    return result.copy(
        size = resolvedSize,
        maxLines = resolvedMaxLines,
        isEnabled = resolvedIsEnabled
    )
}
```

KSP
PROCESSOR

```
override fun process(resolver: Resolver): List<KSAnnotated> {
    logger.info("DataProcessorSymbolProcessor processing started")

    getAnnotatedClasses(resolver)
        .map { DataFactoryFunTemplate(logger, it) }
        .forEach { template ->
            val output = codeGenerator.createNewFile(
                dependencies = Dependencies(
                    aggregating = true,
                    sources = arrayOf(template.sourceFile),
                ),
                packageName = template.packageName,
                fileName = template.fileName,
            )

            PrintWriter(output).use { writer ->
                writer.write(template.generateCode())
            }
        }

    return emptyList()
}
```

```
fun generateCode(): String {
    return buildString {
        appendLine("package $packageName")
        appendLine()
        if (notPublicParametersWithDefault.isNotEmpty()) {
            appendNotPublicFieldNames()
            appendLine()
        }
        appendFactoryFun()

        // remove final new line
        deleteAt(length - 1)
    }
}
```

RECAP

Сейчас уже сразу стоит думать
о возможности использования
KSP (не KAPT)

Мы подумали об этом заранее
и написали свою автоматизацию
через KSP

DEPENDENCY ANALYSIS PLUGIN

Detect unused and misused dependencies

The Dependency Analysis Gradle Plugin (DAGP, née Dependency Analysis Android Gradle Plugin) detects the following:

1. Unused dependencies.
2. Used transitive dependencies (which you may want to declare directly).
3. Dependencies declared on the wrong configuration (`api` vs `implementation` vs `compileOnly`, etc.).

As a side effect, the plugin can also tell you your project's ABI, and produces graphviz files representing various views of your dependency graph, among other things. These side effects are currently mostly undocumented internal behaviors, but they may be interesting for some advanced users.

Build health

In addition to the dependency-related advice (see above), DAGP provides other advice to help maintain your "build health." This includes the detection of:

1. Unnecessary plugins (currently only `kapt`).
2. Subprojects ("modules") that unnecessarily use the Android plugin, and could instead by "normal" JVM libraries.

https://github.com/autonomousapps/dependency-analysis-android-gradle-plugin

RECAP

Ваш даггер скажет вам спасибо (2.44). strictSuperficialValidation

Поможет избежать случаев, когда при подключении библиотеки приходится добавлять еще зависимости, чтобы проект собрался

ОШИБКА 4:
~~НЕ МОНИТРОРИТЬ НОВЫЕ ТЕХНОЛОГИИ~~

РЕШЕНИЕ 4:
МОНИТРОРИТЬ НОВЫЕ ТЕХНОЛОГИИ

ОШИБКА 5:

```
override fun onBoundsChange(bounds: Rect) {
    super.onBoundsChange(bounds)

    with(bounds) {
        deltaX = width() / HALF_SIZE
        deltaY = height() / HALF_SIZE
        maxRadius = minOf(deltaX, deltaY)
        correctedRadius = maxRadius * RADIUS_COEF
        arcAngle = (asin(maxRadius * ARC_COEF / (correctedRadius * 3)) / Math.PI * 180).toFloat()

        var sizeCoefficient = 1f
        if (isBigIcon) {
            arcAngleCrop = arcAngle * CROP_ARC_ANGLE_BIG
            sizeCoefficient = BIG_TO_MEDIUM_SIZE_COEF
            innerCroppingOffset = width() * CROP_INNER_RADIUS_OFFSET_BIG_COEF
        } else {
            arcAngleCrop = arcAngle * CROP_ARC_ANGLE
            innerCroppingOffset = width() * CROP_INNER_RADIUS_OFFSET_COEF
        }
        outerCroppingRadius = width() * CROP_OUTER_RADIUS_COEF * sizeCoefficient
        innerCroppingRadius = width() * CROP_INNER_RADIUS_COEF * sizeCoefficient
        calculateSuperEllipsePath()
        rect.set(0f, 0f, width().toFloat(), height().toFloat())
    }
    val shaderMatrix = Matrix()
    shaderMatrix.setRectToRect(bitmapRect, rect, Matrix.ScaleToFit.CENTER)
    bitmapShader.setLocalMatrix(shaderMatrix)
}
```

```
private fun calculateRightBottomArcCropped() {
    val pathMeasure = PathMeasure(path, false)
    val pathEnd = floatArrayOf(0f, 0f)
    pathMeasure.getPosTan(pathMeasure.length, pathEnd, null)
    val (x, y) = pathEnd

    val outerRect = RectF(
        x - 2 * outerCroppingRadius,
        y - outerCroppingRadius,
        x,
        y + outerCroppingRadius
    )
    path.arcTo(outerRect, 0f, 90f)
    val innerRect = RectF(
        outerRect.centerX() - innerCroppingRadius,
        outerRect.bottom,
        outerRect.centerX() + innerCroppingRadius,
        outerRect.bottom + 2 * innerCroppingRadius
    )
    innerRect.offset(-innerCroppingOffset, 0f)
    path.arcTo(innerRect, 270f, -90f)
    with(outerRect) {
        left = innerRect.left - 2 * outerCroppingRadius
        top = innerRect.centerY() - outerCroppingRadius
        right = innerRect.left
        bottom = innerRect.centerY() + outerCroppingRadius
    }
    outerRect.offset(0f, innerCroppingOffset)
    path.arcTo(outerRect, 0f, 90f)
}
```

RECAP

Очень сложно понять,
что происходит при последующих доработках

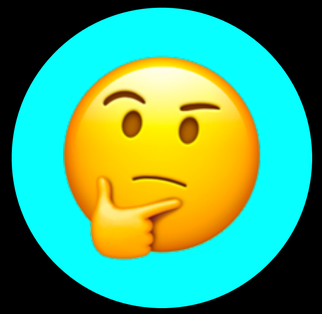
В подобных кейсах не надо бояться просить покрывать код javadoc

ОШИБКА 5:
~~ОТСУТСТВИЕ ДОКУМЕНТАЦИИ~~

РЕШЕНИЕ 5:
ПРИСУТСТВИЕ ДОКУМЕНТАЦИИ

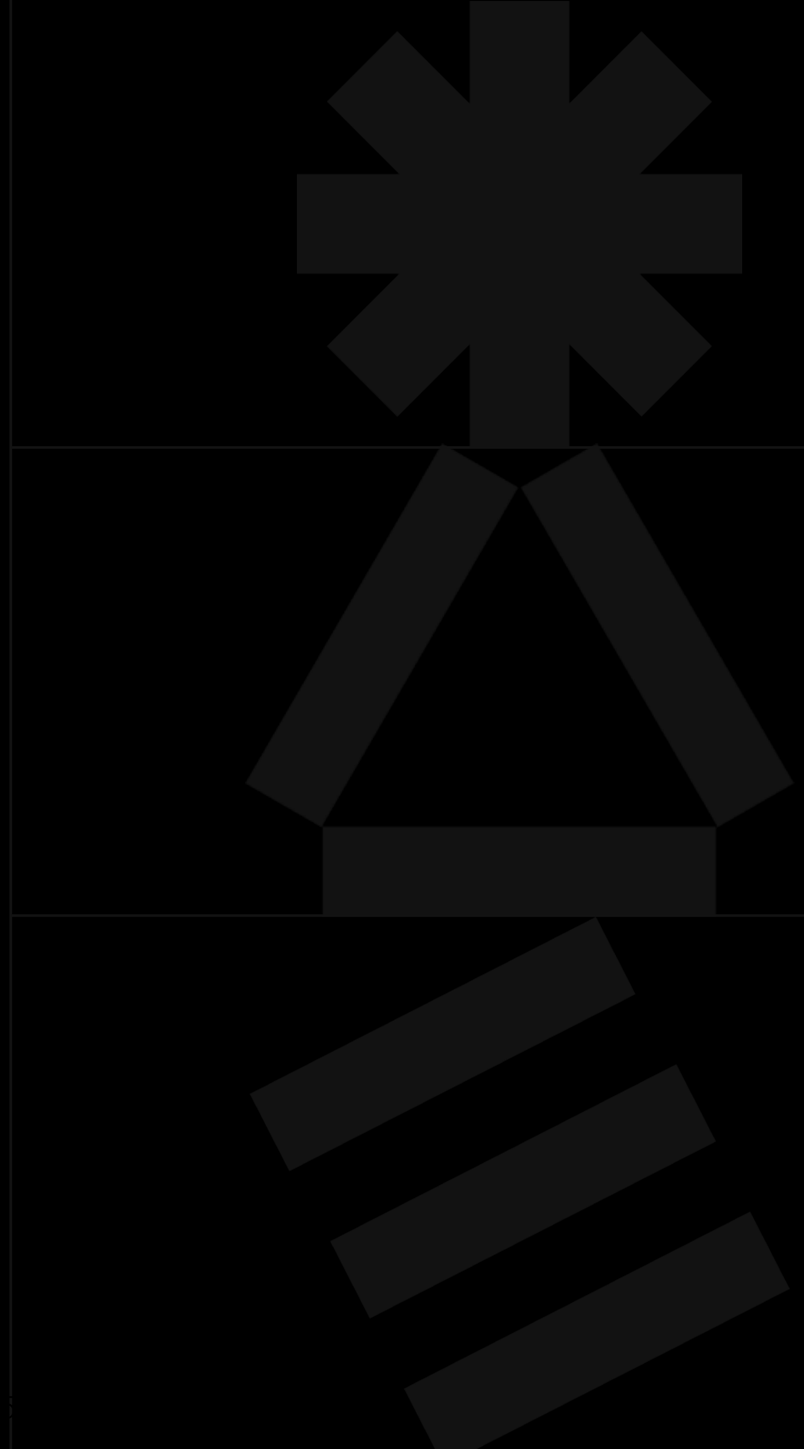
ОШИБКА 6:

ОТКРЫВАЮ Я КАК-ТО УТРОМ РАБОЧИЙ ЧАТ:

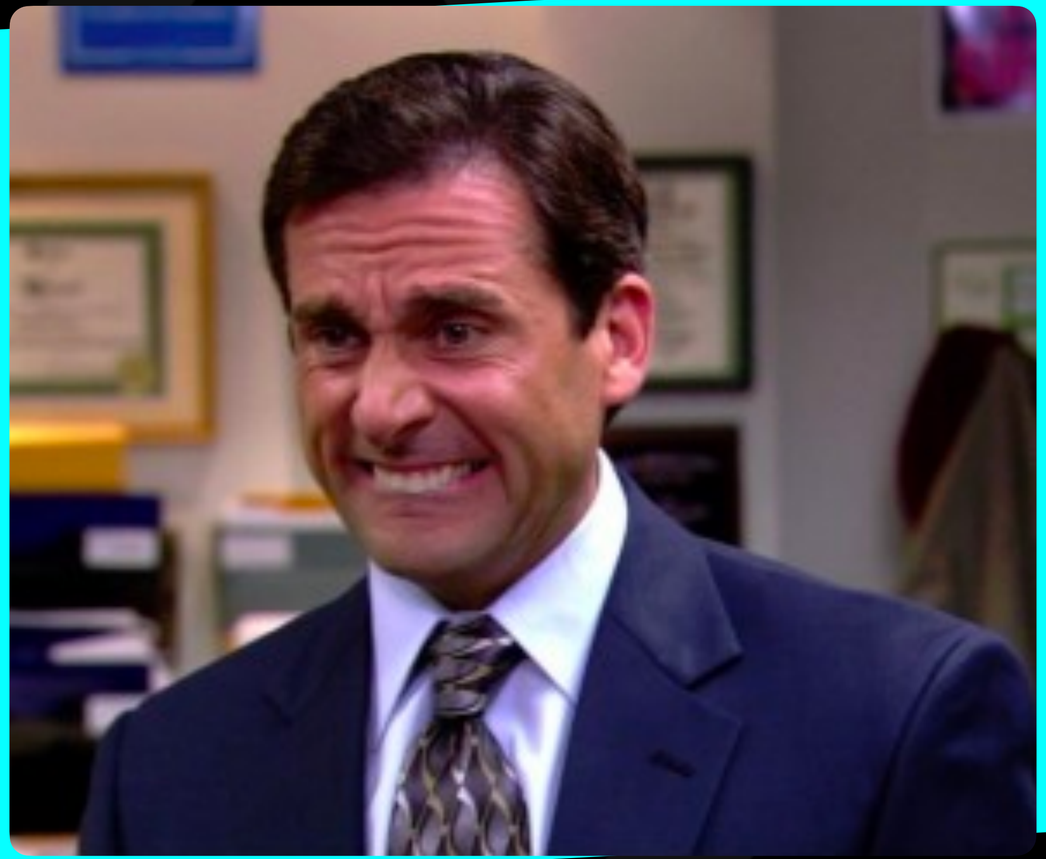


Хотел узнать, почему сделано
именно так, что без этого
поля метрики не шлются
и было ли какое-то
уведомление для мидловиков,
что метрики отвалятся,
если это поле не слать?

БЫВАЛО
И ТАКОЕ:




вот актуалочка 14:40
мы монорепу внедрили 14:40
да блед 14:40
этого я не знал, конечно) 14:40
ахахаххаха 14:40
Я по старинке 14:40
https://git.moscow.alfaintra.net/projects/ANDROID/repos/brandbook/browse
14:41



PROBLEMS

Нет единой системы оповещения о важных событиях в api (changelog работает не всегда, а сарафанное радио не работает почти никогда)

Это приводит к проблемам, на которые командам приходится тратить свое время, а потом они узнают, что это не баг, а фича нового api

RECAP

Теперь у нас на особенно важные вещи есть репозиторий с json схемами, чтобы туда замержиться нужно по 1 апруву от каждой платформы

Есть общие рабочие каналы по каждой библиотеке, куда выносятся новости об обновлениях

В планах автоматизация оповещений с помощью бота (но тут главное не попасть в ситуацию с анекдотом про волков)

ОШИБКА 6:
~~ОТСУТСТВИЕ СТАБИЛЬНОГО КАНАЛА ОПОВЕЩЕНИЙ~~

РЕШЕНИЕ 6:
ПРИСУТСТВИЕ СТАБИЛЬНОГО КАНАЛА ОПОВЕЩЕНИЙ

EXTRA BONUS

INFRASTRUCTURE

Всегда продумывайте как будет публиковаться ваша библиотека. Например если это maven publication то все зависимости должны быть описаны в pom файле.

Хотите многомодульность в библиотеке ? Тоже придется подумать

Чем стабильнее CI тем меньше психопатов в команде

```
<groupId>ru.alfabank.android</groupId>
<artifactId>brandbook</artifactId>
<version>1.999.999</version>
<packaging>aar</packaging>
<dependencies>
  <dependency>
    <groupId>com.google.dagger</groupId>
    <artifactId>dagger</artifactId>
    <version>2.44.2</version>
  </dependency>
  <dependency>
    <groupId>com.google.android.renderscript</groupId>
    <artifactId>renderscript-toolkit</artifactId>
    <version>1.0</version>
  </dependency>
```

НАШИ ВЫВОДЫ:

- Закрывать всё, что возможно по максимуму
- Стараться не использовать (data, sealed и т.д) классы и даже если используем смотреть, как они себя ведут в рамках бинарных изменений
- Улыбаться и махать

ПОЛЕЗНЫЕ ИСТОЧНИКИ

https://jakewharton.com/public-api-challenges-in-kotlin/

https://docs.gradle.org/current/userguide/dependency_management.html#controlling_transitive_dependencies

https://docs.oracle.com/javase/specs/jls/se7/html/jls-13.html

https://youtube.com/@jjohannes

https://youtu.be/g1Pqyskjx8M

https://docs.oracle.com/javase/specs/jvms/se20/html/jvms-4.html#jvms-4.4

https://youtu.be/2wkEX6MCxJs

https://youtu.be/aqQT3J16OxI


WITH GREAT POWER
COMES GREAT RESPONSIBILITY.

СПАСИБО

Абакар Магомедов

Android-Techlead
Альфа-Банк

@Abocha

Alfa Digital