GDG Izhevsk

GDG Mobile Meetup

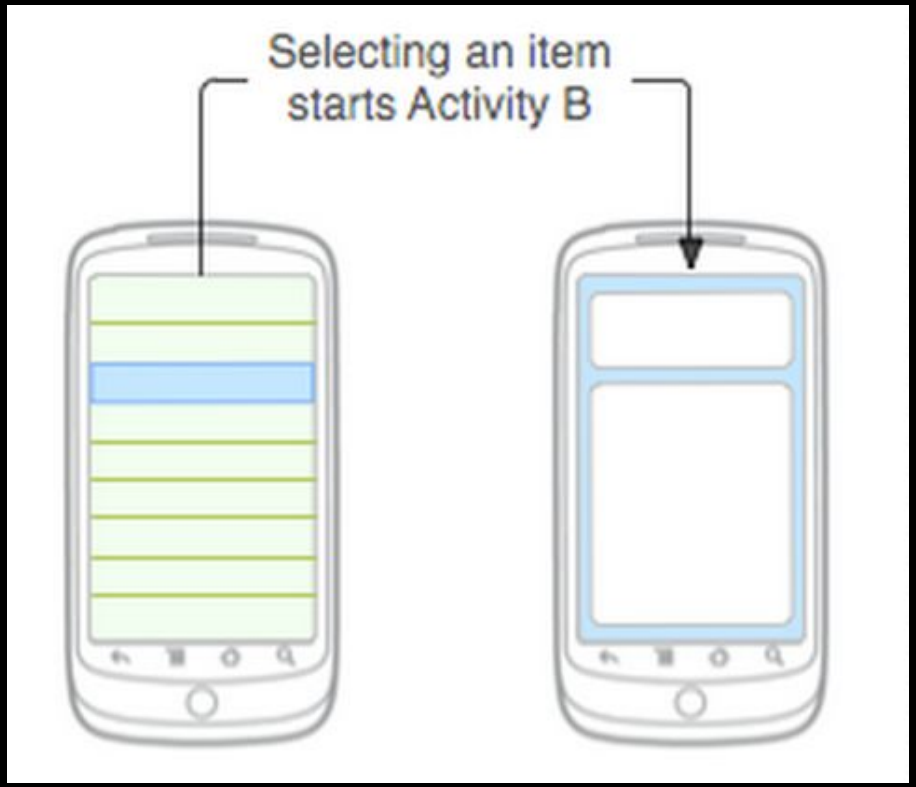
Izhevsk, 20/02

GDG Izhevsk

Навигация в Android с Kotlin

Упоров Дмитрий
Руководитель отдела мобильной разработки, ЦВТ

Master-details



GDG Izhevsk

Java - Junior way

```
public void itemChosen(SomeModel model) {
    Intent intent = new Intent( packageContext: this, DetailsActivity.class);
    intent.putExtra(DetailsActivity.MODEL_KEY, model);
    startActivity(intent);
}
```

Минусы:

- Дублирование, многословность
- Неочевиден контракт передачи данных

GDG Izhevsk

Java - The way: Factory-method

```
public class DetailsActivity extends AppCompatActivity {

    private static final String MODEL_KEY = "modelKey";

    public static void start(Context context, SomeModel model) {
        Intent intent = new Intent(context, DetailsActivity.class);
        intent.putExtra(MODEL_KEY, model);
        context.startActivity(intent);
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        SomeModel model = (SomeModel) getIntent().getSerializableExtra(MODEL_KEY);
    }
}
```

GDG Izhevsk

Java - The way

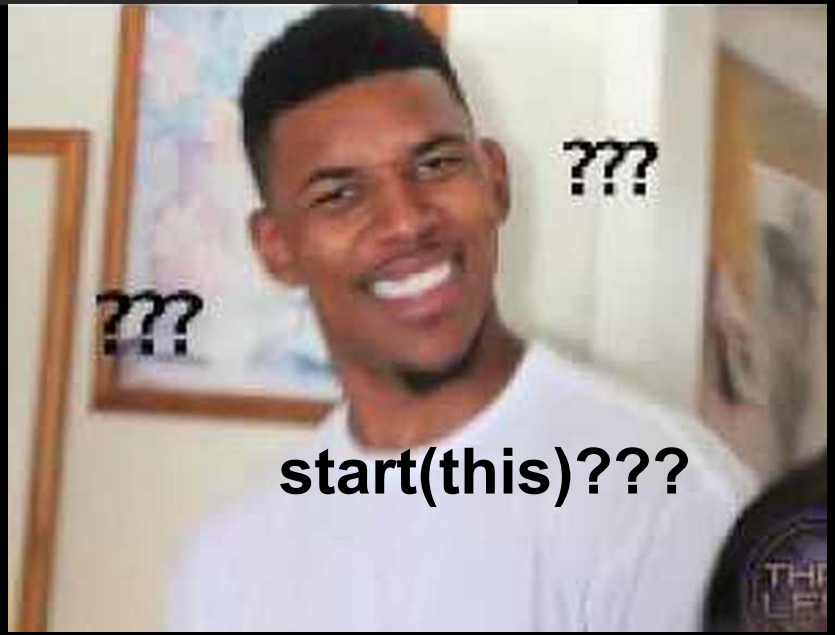
```
public void itemChosen(SomeModel model) {
    DetailsActivity.start( context: this, model);
}
```

Плюсы:

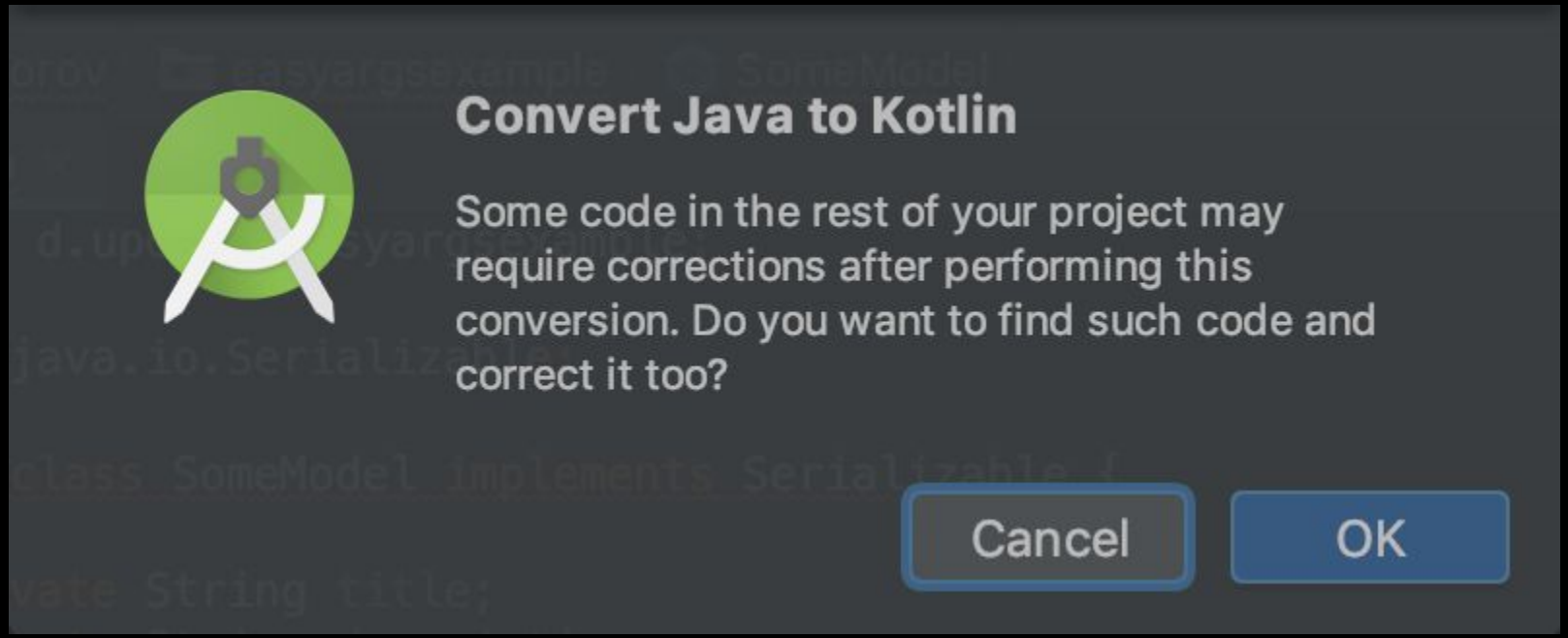
- Лаконичность вызова

Минусы:

- Неочевидность
- Boilerplate code
- Нагрузка статикой, нарушение SR



Java -> Kotlin


Convert Java to Kotlin
Some code in the rest of your project may require corrections after performing this conversion. Do you want to find such code and correct it too?
Cancel
OK

Java -> Kotlin

```kotlin
class DetailsActivity : AppCompatActivity() {

    companion object {
        private const val MODEL_KEY = "modelKey"

        fun start(context: Context, model: SomeModel) {
            val intent = Intent(context, DetailsActivity::class.java)
            intent.putExtra(MODEL_KEY, model)
            context.startActivity(intent)
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val model : SomeModel = intent.getSerializableExtra(MODEL_KEY) as SomeModel
    }
}
```

GDG Izhevsk

Kotlin - a way

```
fun Context.startDetailsActivity(model: SomeModel) {
    Intent( packageContext: this, DetailsActivity::class.java)
        .apply { putExtra(MODEL_KEY, model) }
        .also(::startActivity)
}
```

```
fun itemChosen(model: SomeModel) {
    startDetailsActivity(model)
}
```

- ~~Неочевидность~~
- Boilerplate code
- Нагрузка статикой, нарушение SR

GDG Izhevsk

Kotlin - better way

```kotlin
private const val MODEL_KEY = "modelKey"

fun Context.startDetailsActivity(model: SomeModel) {
    Intent( packageContext: this, DetailsActivity::class.java)
        .apply { putExtra(MODEL_KEY, model) }
        .also(::startActivity)
}

class DetailsActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val model : SomeModel = intent.getSerializableExtra(MODEL_KEY) as SomeModel
    }
}
```

- Boilerplate code
- ~~Нагрузка статикой, нарушение SR~~

GDG Izhevsk

Kotlin - better way

```
fun Context.startDetailsActivity() {
    Intent(packageContext: this, DetailsActivity::class.java)
        .also(::startActivity)
}
```

GDG Izhevsk

Kotlin - better way

```
fun Context.start(clazz: Class<out AppCompatActivity>) {
    Intent( packageContext: this, clazz)
        .also(::startActivity)
}
```

```
start(DetailsActivity::class.java)
```

GDG Izhevsk

Kotlin - The best way

```
inline fun <reified T : AppCompatActivity> Context.start() {
    intent( packageContext: this, T::class.java)
        .also(::startActivity)
}
```

```
start<DetailsActivity>()
```

Decompiled byte-code:

```
public final void itemChosen() {
   Intent var2 = new Intent( packageContext: this, DetailsActivity.class);
   ((Context)this).startActivity(var2);
}
```

GDG Izhevsk

GDG Izhevsk


Parcelable
GDG Izhevsk

Model wrapping - pervy' blin komom

Обертка над каждой моделью:

```
private const val SOME_MODEL_KEY = "SOME_MODEL_KEY"

infix fun Intent.with(oneModel: SomeModel): Intent = putExtra(SOME_MODEL_KEY, oneModel)

fun Intent.getSomeModelOrNull(): SomeModel? = getSerializableExtra(SOME_MODEL_KEY) as? SomeModel
```

Плюсы:

- Инкапсуляция
- Читабельность

Минусы:

- Boilerplate
- Несколько объектов одного типа

GDG Izhevsk

Model wrapping - pervy' blin komom

```
inline fun <reified T : AppCompatActivity> Context.start(vararg items: Any?) {
    startActivity(Intent( packageContext: this, T::class.java).apply { this: Intent
        for (item :Any? in items) {
            when (item) {
                null -> { }
                is SomeModel -> this with item
                is AnotherModel -> this with item
                else -> throw IllegalArgumentException("Not supported type of item $item")
            }
        }
    })
}
```

Плюсы:

- Единая точка обработки
- Расширяемость (ха-ха)

Минусы:

- Необходимость расширения
- RuntimeException
- Неоформляемо

GDG Izhevsk

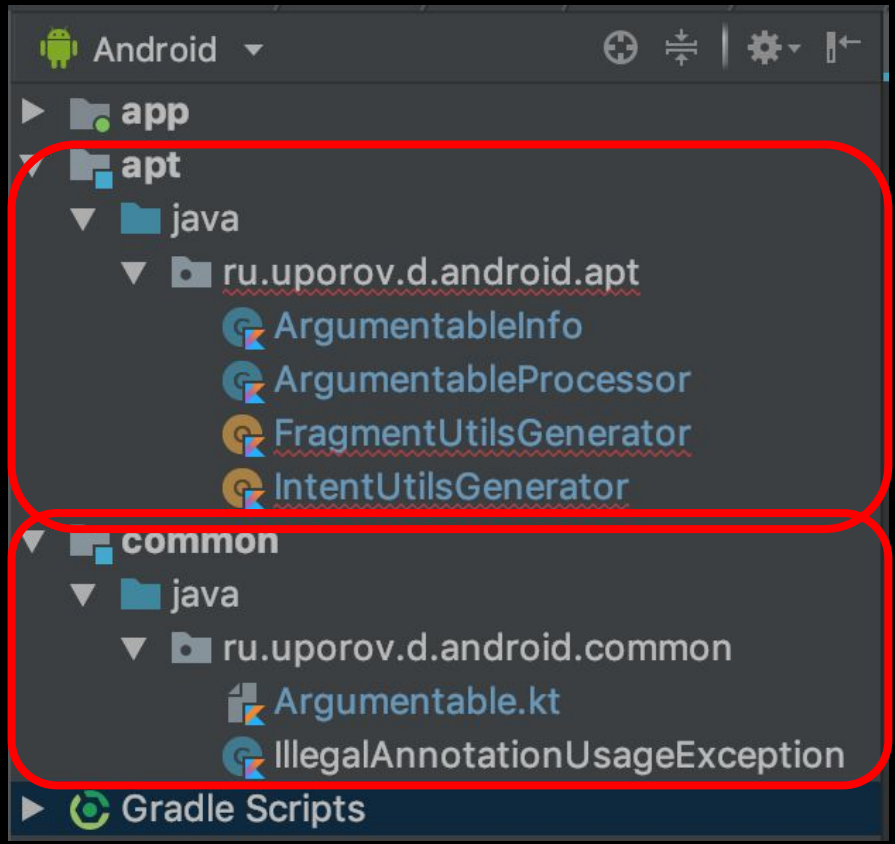

GDG Izhevsk


GDG Izhevsk

КОДОГЕНЕРАЦИЯ!!1! >:D


GDG Izhevsk

Model wrapping - Wrong way



GDG Izhevsk

Annotation processor

```kotlin
@AutoService(Processor::class) // For registering the service
@SupportedSourceVersion(SourceVersion.RELEASE_8) // to support Java 8
@SupportedOptions(ArgumentableProcessor.KAPT_KOTLIN_GENERATED_OPTION_NAME)
class ArgumentableProcessor : AbstractProcessor() {

    companion object {
        const val KAPT_KOTLIN_GENERATED_OPTION_NAME = "kapt.kotlin.generated"
        const val PACKAGE = "d.uporov.argumentable"
    }

    override fun getSupportedAnnotationTypes(): MutableSet<String> {
        return mutableSetOf(Argumentable::class.java.name)
    }

    override fun getSupportedSourceVersion(): SourceVersion {
        return SourceVersion.latest()
    }

    override fun process(set: MutableSet<out TypeElement>?, roundEnvironment: RoundEnvironment): Boolean {
        roundEnvironment.getElementsAnnotatedWith(Argumentable::class.java)
            .map { it: Element!
                if (it.kind == ElementKind.CLASS)  ^map it
                else throw IllegalAnnotationUsageException(Argumentable::class)
            }
            .map { it: Element!
                if (it is Symbol.ClassSymbol)  ^map it
                else throw IllegalAnnotationUsageException(Argumentable::class)
            }
            // needed to skip processing round
            .ifEmpty { return false }
            .toArgumentableInfoList()
            .also { it: List<ArgumentableInfo>
                generateIntentUtils(PACKAGE, it).write()
                generateFragmentUtils(PACKAGE, it).write()
            }
        return true
    }

    private fun List<Symbol.ClassSymbol>.toArgumentableInfoList(): List<ArgumentableInfo> {
        val list : MutableList<ArgumentableInfo>  = mutableListOf<ArgumentableInfo>()
        forEach { element ->
            var argumentName = ""
            var type = ArgumentableType.SERIALIZABLE
            for (annotation : Attribute.Compound!  in element.annotationMirrors) {
                for (pair : Pair<Symbol.MethodSymbol!, Attribute!>!  in annotation.values) {
                    when (pair.fst.simpleName.toString()) {
                        "argumentName" -> argumentName = pair.snd.value as String
//                          "argumentType" -> type = pair.snd.value as ArgumentableType
                    }
                }
            }
            element.toArgumentableInfo(argumentName, type).let(list::add)
        }
        return list
    }

    private fun Symbol.ClassSymbol.toArgumentableInfo(argumentName: String, type: ArgumentableType): ArgumentableInfo =
        asClassName().let { it: ClassName
            ArgumentableInfo(it, type, if (argumentName.isEmpty()) it.simpleName else argumentName)
        }

    private fun FileSpec.write() = writeTo(File(processingEnv.options[KAPT_KOTLIN_GENERATED_OPTION_NAME]))
}
```

Model wrapping - Wrong way

```
@Argumentable(argumentName = "special")
class SomeModel(val name: String) : Serializable
```

```
private const val SOME_MODEL_KEY = "SOME_MODEL_KEY"

infix fun Intent.with(oneModel: SomeModel): Intent = putExtra(SOME_MODEL_KEY, oneModel)

fun Intent.getSomeModelOrNull(): SomeModel? = getSerializableExtra(SOME_MODEL_KEY) as? SomeModel
```

```
inline fun <reified T : AppCompatActivity> Context.start(vararg items: Any?) {
    startActivity(Intent( packageContext: this, T::class.java).apply { this: Intent
        for (item : Any? in items) {
            when (item) {
                null -> { }
                is SomeModel -> this with item
                is AnotherModel -> this with item
                else -> throw IllegalArgumentException("Not supported type of item $item")
            }
        }
    })
}
```

GDG Izhevsk

Java Mirror API

GDG Izhevsk

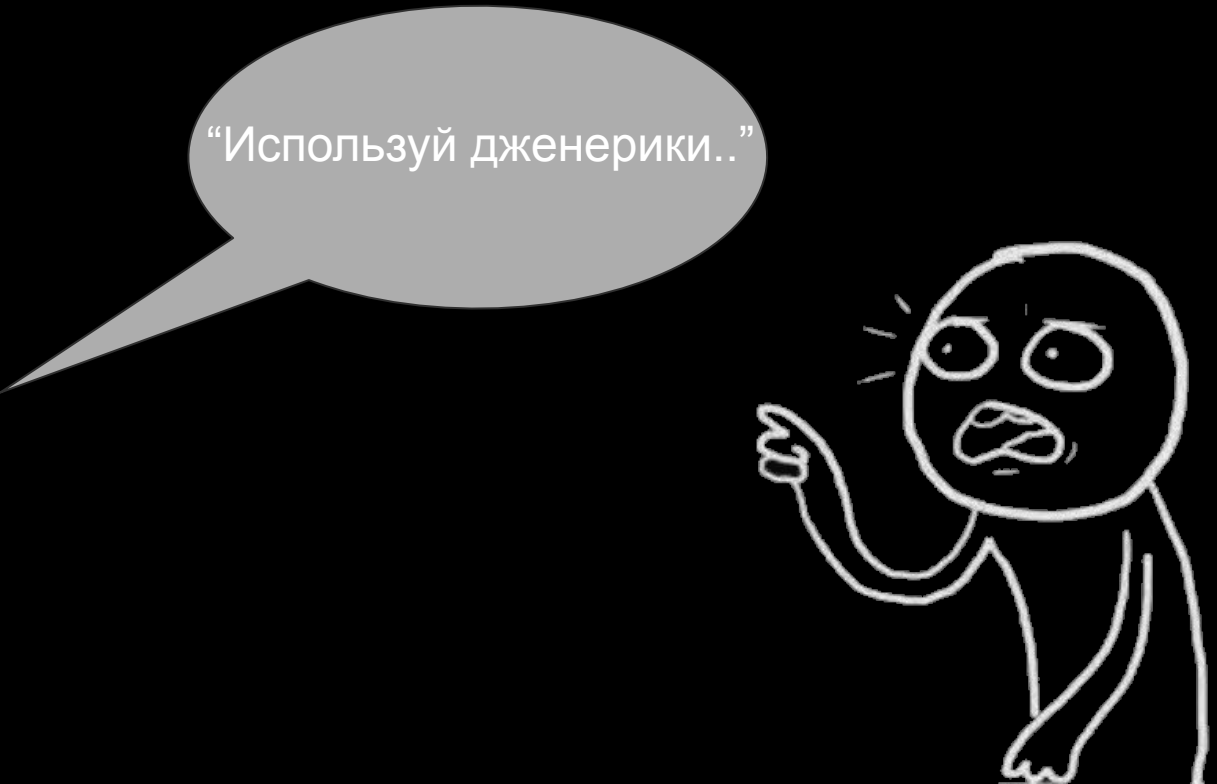
APT == MirrorAPI != Reflection

- Нет информации о реализованных интерфейсах

```
@Target(AnnotationTarget.CLASS)
@Retention(AnnotationRetention.SOURCE)
annotation class Argumentable<T>(
    val argumentName: String = "",
    val argumentType: KClass<T>
) where T : Serializable, T : Parcelable
```

- Нет ограничения реализации “Или то, или другое”
- Нет гарантии, что пользователь подставит в аннотацию свой класс
- Невозможно явно сказать пользователю, что мы хотим от него реализации интерфейса


GDG Izhevsk



GDG Izhevsk

Kotlin - Wrapped model getting

```
inline fun <reified T> Intent.getArgOrNull(byKey: String = T::class.java.simpleName): T? {
    return when {
        T::class.isParcelable() -> getParcelableExtra<Parcelable>(byKey) as? T
        T::class.isSerializable() -> getSerializableExtra(byKey) as? T
        else -> throw IllegalArgumentException("${Intent::class} cannot have arguments by type ${T::class}")
    }
}
```

```
fun KClass<*>.isSerializable() = Serializable::class.java.isAssignableFrom(this.java)

fun KClass<*>.isParcelable() = Parcelable::class.java.isAssignableFrom(this.java)
```

```
intent.getArgOrNull<SomeModel>()
```

- RuntimeException

GDG Izhevsk

There is way to avoid RuntimeException?

```
inline fun <reified T : Serializable> Intent.getArgOrNull(): T? {
    return getSerializableExtra(T::class.java.simpleName) as? T
}

inline fun <reified T : Parcelable> Intent.getArgOrNull(): T? {
    return getParcelableExtra(T::class.java.simpleName) as? T
}
```

No way, but...

Input

Any?

Parcelable
Serializable

GDG Izhevsk

Model wrapping - Kotlin better way

```
inline fun <reified T : AppCompatActivity> Context.createIntent(vararg items: Any?): Intent {
    return Intent( packageContext: this, T::class.java).apply { this: Intent
        for (item : Any? in items) {
            when (item) {
                null -> Unit
                is Serializable -> this with item
                is Parcelable -> this with item
                else -> throw IllegalArgumentException(
                    "Class ${item::class} must implement " +
                            Parcelable::class + " or " + Serializable::class
                )
            }
        }
}

infix fun <T : Parcelable> Intent.with(parcelable: T): Intent = apply { this: Intent
    putExtra(parcelable::class.java.simpleName, parcelable)
}

infix fun <T : Serializable> Intent.with(serializable: T): Intent = apply { this: Intent
    putExtra(serializable::class.java.simpleName, serializable)
}
```

GDG Izhevsk

How to filter Serializable or Parcelable?

- Перегрузка метода == ограничение всех параметров одним интерфейсом
- Ограничения реализации “Или то, или другое” (невозможно)
- 2 массива: Serializable и Parcelable. Ограниченный порядок аргументов

Model wrapping - Kotlin the best way

```kotlin
data class Argument<T> internal constructor(val arg: T)

fun <T : Parcelable> T.asArg() = Argument( arg: this)

fun <T : Serializable> T.asArg() = Argument( arg: this)
```

GDG Izhevsk

Model wrapping - Kotlin the best way

```
inline fun <reified T : AppCompatActivity> Context.createIntent(vararg items: Argument<*>) : Intent {
    return Intent( packageContext: this, T::class.java).apply { this: Intent
        for (item : Argument<*> in items) {
            when (item.arg) {
                is Serializable -> this with item.arg
                is Parcelable -> this with item.arg
                else -> Unit
            }
        }
    }
}
```

```
start<DetailsActivity>(SomeModel().asArg())
```

GDG Izhevsk

Improvements time

```
data class Argument<T> internal constructor(val arg: T)

fun <T : Parcelable> T.asArg() = Argument( arg: this)

fun <T : Serializable> T.asArg() = Argument( arg: this)
```

```
class SomeModel() : Serializable, Parcelable
```

```
74        SomeModel().asArg()
```

Cannot choose among the following candidates without completing type inference:

- **public fun** <T : Parcelable> SomeModel.asArg(): Argument<SomeModel> *defined in* com.htc_cs.vkurse
- **public fun** <T : Serializable> SomeModel.asArg(): Argument<SomeModel> *defined in* com.htc_cs.vkurse

Improvements time

```
(SomeModel() as Parcelable).asArg()
```

GDG Izhevsk

Improvements time

```
data class Argument<T> internal constructor(val arg: T)

fun <T : Parcelable> T.asArg() = Argument( arg: this)

fun <T : Serializable> T.asArg() = Argument( arg: this)

fun <T> T.asArg() where T : Serializable, T : Parcelable = Argument( arg: this)
```

```
data class Argument<T> internal constructor(val arg: T)

fun <T : Parcelable> T.asArg() = Argument( arg: this)

fun <T : Serializable> T.asArg() = Argument( arg: this)

fun <T> T.asArg() where T : Parcelable, T : Serializable = Argument( arg: this)
```

Improvements time

```
@NotNull
public static final Argument asArg(@NotNull Parcelable $receiver) {
    Intrinsics.checkParameterIsNotNull($receiver, "receiver$0");
    return new Argument($receiver);
}

@NotNull
public static final Argument asArg(@NotNull Serializable $receiver) {
    Intrinsics.checkParameterIsNotNull($receiver, "receiver$0");
    return new Argument($receiver);
}

@NotNull
public static final Argument asArg(@NotNull Serializable $receiver) {
    Intrinsics.checkParameterIsNotNull($receiver, "receiver$0");
    return new Argument($receiver);
}
```

GDG Izhevsk

Kotlin magic time

```
data class Argument<T> internal constructor(val arg: T)

fun <T : Parcelable> T.asArg() = Argument( arg: this)

fun <T : Serializable> T.asArg() = Argument( arg: this)

fun <T> T.asArg() where T : Any?, T : Parcelable, T : Serializable = Argument( arg: this)
```

```
data class Argument<T> internal constructor(val arg: T)

fun <T : Parcelable> T.asArg() = Argument( arg: this)

fun <T : Serializable> T.asArg() = Argument( arg: this)

fun <T> T.asArg() where T : Parcelable, T : Serializable, T : Any? = Argument( arg: this)
```

Improvements time

```
@NotNull
public static final Argument asArg(@NotNull Parcelable $receiver) {
    Intrinsics.checkParameterIsNotNull($receiver, "receiver$0");
    return new Argument($receiver);
}

@NotNull
public static final Argument asArg(@NotNull Serializable $receiver) {
    Intrinsics.checkParameterIsNotNull($receiver, "receiver$0");
    return new Argument($receiver);
}

@NotNull
public static final Argument asArg(Object $receiver) {
    return new Argument($receiver);
}
```

GDG Izhevsk

Success!

```
data class Argument<T> internal constructor(val arg: T)

fun <T : Parcelable> T.asArg() = Argument( arg: this)

fun <T : Serializable> T.asArg() = Argument( arg: this)

fun <T> T.asArg() where T : Any?, T : Parcelable, T : Serializable = Argument( arg: this)
```

GDG Izhevsk

Improvements time 2

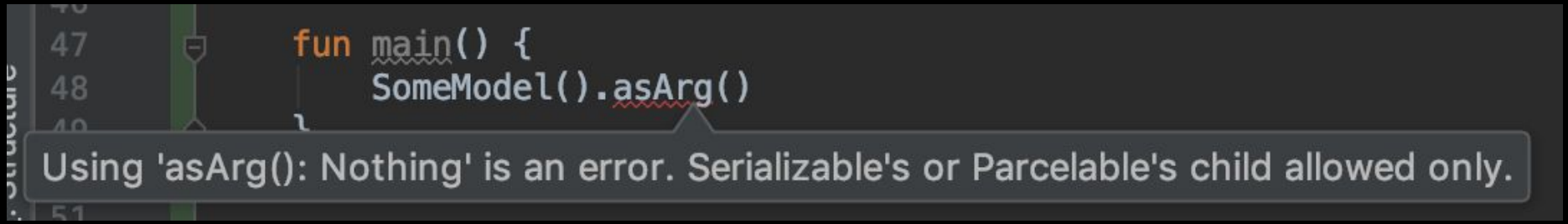
```
class SomeModel

fun main() {
    SomeModel().asArg()
}
```

GDG Izhevsk

Improvements time 2

```
@Suppress( ...names: "DeprecatedCallableAddReplaceWith")
@Deprecated(
    message = "Serializable's or Parcelable's child allowed only.",
    level = DeprecationLevel.ERROR
)
fun Any?.asArg(): Nothing = throw IllegalArgumentException()
```

Success 2!



GDG Izhevsk

Improvements time 3 - ArgumentWithKey

```
data class ArgumentWithKey<T> internal constructor(val arg: T, val key: String)

fun <T : Parcelable> T.asArg(key: String = this::class.java.simpleName) = ArgumentWithKey( arg: this, key)

fun <T : Serializable> T.asArg(key: String = this::class.java.simpleName) = ArgumentWithKey( arg: this, key)
```

```
fun <T> ArgumentWithKey<T>.asIntent() = Intent().apply {
    when (arg) {
        is Serializable -> putExtra(key, arg)
        is Parcelable -> putExtra(key, arg)
    }
}
```

GDG Izhevsk

Improvements time 4: Intent.() -> Unit

```
Context.start(intentSettings: Intent.() -> Unit = {})
```

```
start<DetailsActivity> {  this: Intent
    flags = FLAG_ACTIVITY_SINGLE_TOP
}
```

Improvements time 5: Other components

Fragment way:

```
newInstance<DetailsFragment>(SomeModel().asArg())
```

Service way:

```
start<SomeService>()
```

to be continued..

GDG Izhevsk

easyargs

Download 0.0.3

With this small library you can easy navigate to activity or fragment. Convenient API allows you to put arguments (into Intent or Bundle) and to get them later.

```
class MasterActivity: AppCompatActivity {

  fun onItemChosen(item: Model) {
    start<DetailsActivity>(item.asArg())
  }
}

class DetailsActivity: AppCompatActivity {

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    val item: Model = getArg()
  }
}
```

Download

Project `build.gradle`

```
buildscript {
  repositories {
    jcenter()
  }
}
```

Module `build.gradle`

```
dependencies {
  implementation 'com.github.udy18rus:easyargs:$easyargs_version'
}
```

GDG Izhevsk

Спасибо за внимание!

Упоров Дмитрий
Руководитель отдела мобильной разработки, ЦВТ