Your first multiplatform Kotlin project

About me

Marcin Moskała
@marcinmoskala
marcinmoskala@gmail.com

Trainer, consultant, author of
Android Development in Kotlin
Founder of Kt. Academy
www.kt.academy

Goal


Effective
multiplatform
Kotlin
development
by Marcin Moskała


How?

```
fun main(args: Array<String>) {
    println("Hello, World")
}
```

```
public final class TestKt {


  // access flags 0x19
  public final static main([Ljava/lang/String;)V
    @Lorg/jetbrains/annotations/NotNull;() // in
   L0
    ALOAD 0
    LDC "args"
    INVOKESTATIC kotlin/jvm/internal/Intrinsics.
   L1
    LINENUMBER 5 L1
    LDC "Hello, World"
    ASTORE 1
   L2
    GETSTATIC java/lang/System.out : Ljava/io/Pr
    ALOAD 1
    INVOKEVIRTUAL java/io/PrintStream.println (L
   L3
   L4
    LINENUMBER 6 L4
    RETURN
   L5
    LOCALVARIABLE args [Ljava/lang/String; L0 L5
    MAXSTACK = 2
    MAXLOCALS = 2
```

Kotlin/JVM

```
kotlin.kotlin.io.output.flush();
if (typeof kotlin === 'undefined
  throw new Error("Error loading
}
var moduleId = function (_, Kotl
  'use strict';
  var println = Kotlin.kotlin.io
  function main(args) {
    println('Hello, World');
  }
  _.main_kand9s$ = main;
  main([]);
  Kotlin.defineModule('moduleId'
  return _;
}(typeof moduleId === 'undefined

kotlin.kotlin.io.output.buffer;
```

Kotlin/JS

```
<kfun:main(kotlin.Array<kotlin.String>)>:
push   %rbp
mov    %rsp,%rbp
mov    $0x4491b0,%edi
callq  429210 <Kotlin_io_Console_println>
pop    %rbp
retq

<EntryPointSelector>:
push   %rax
callq  409ea0 <kfun:main(kotlin.Array<kotlin
pop    %rax
retq
nopl   0x0(%rax,%rax,1)


<kfun:kotlin.Unit.toString()Reference>:
push   %rbp
mov    %rsp,%rbp
mov    %rsi,%rax
mov    $0x4491f0,%esi
mov    %rax,%rdi
callq  4252f0 <UpdateReturnRef>
mov    $0x4491f0,%eax
pop    %rbp
retq
nopl   0x0(%rax,%rax,1)
```

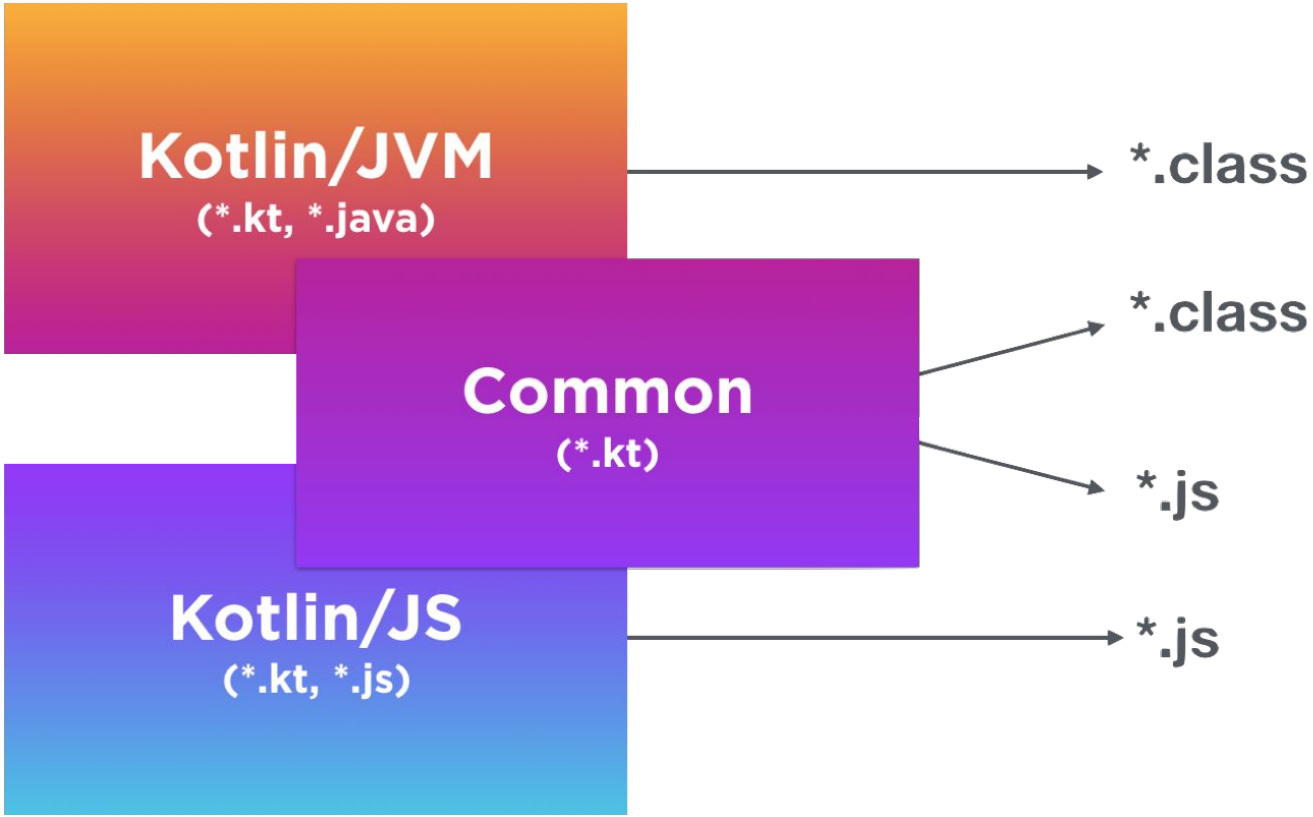
Kotlin/Native

Kotlin/JVM

Kotlin/Native

Kotlin/JS

Kotlin/JVM


Kotlin/JVM
(*.kt, *.java)
Common
(*.kt)
Kotlin/JS
(*.kt, *.js)
*.class
*.class
*.js
*.js

Plan

- Make your first multiplatform project
- Build MPP with shared logic
- MVVM
- View Binding

Make your first multiplatform project

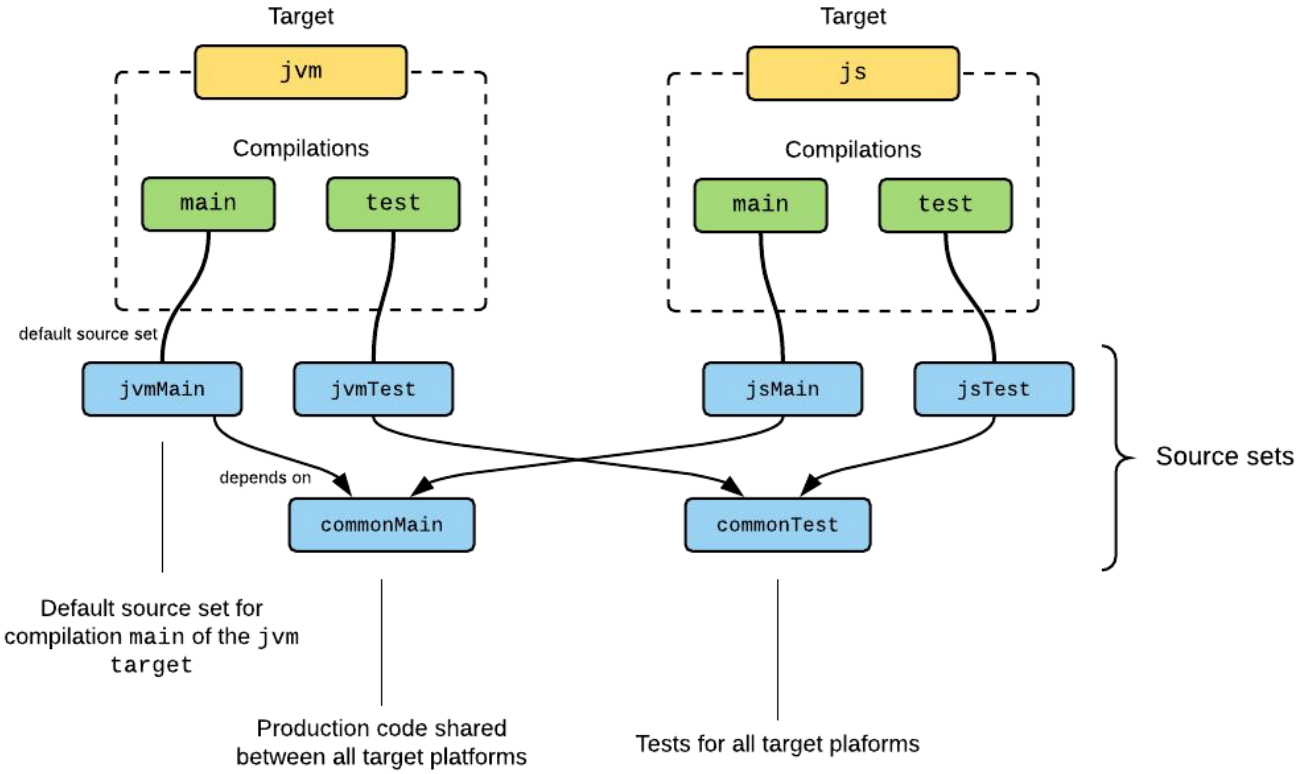
Building Multiplatform Projects with Gradle

Edit Page

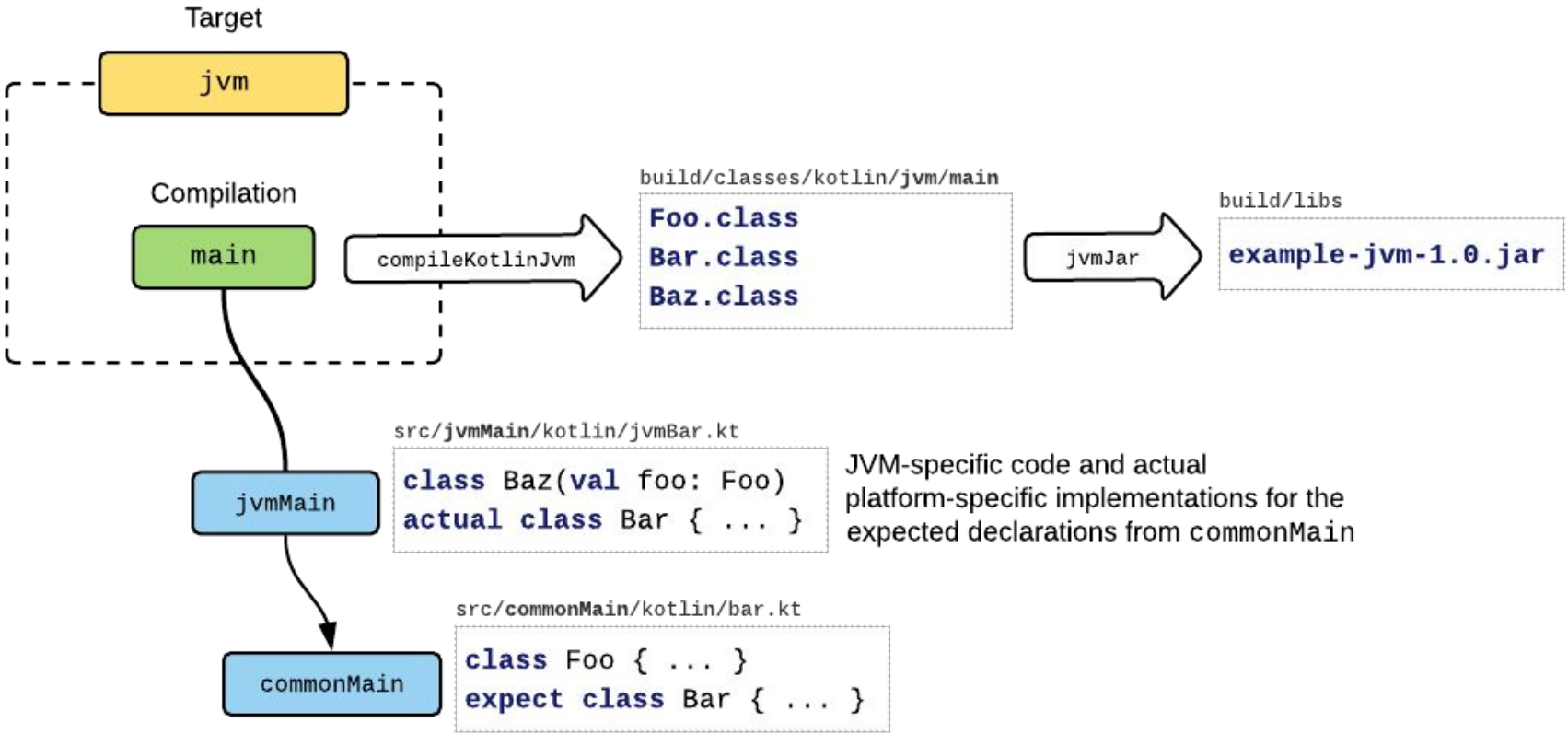
⚠ Multiplatform projects are an experimental feature in Kotlin 1.2 and 1.3. All of the language and tooling features described in this document are subject to change in future Kotlin versions.

This document explains the structure of Kotlin multiplatform projects and describes how those are configured and built using Gradle.

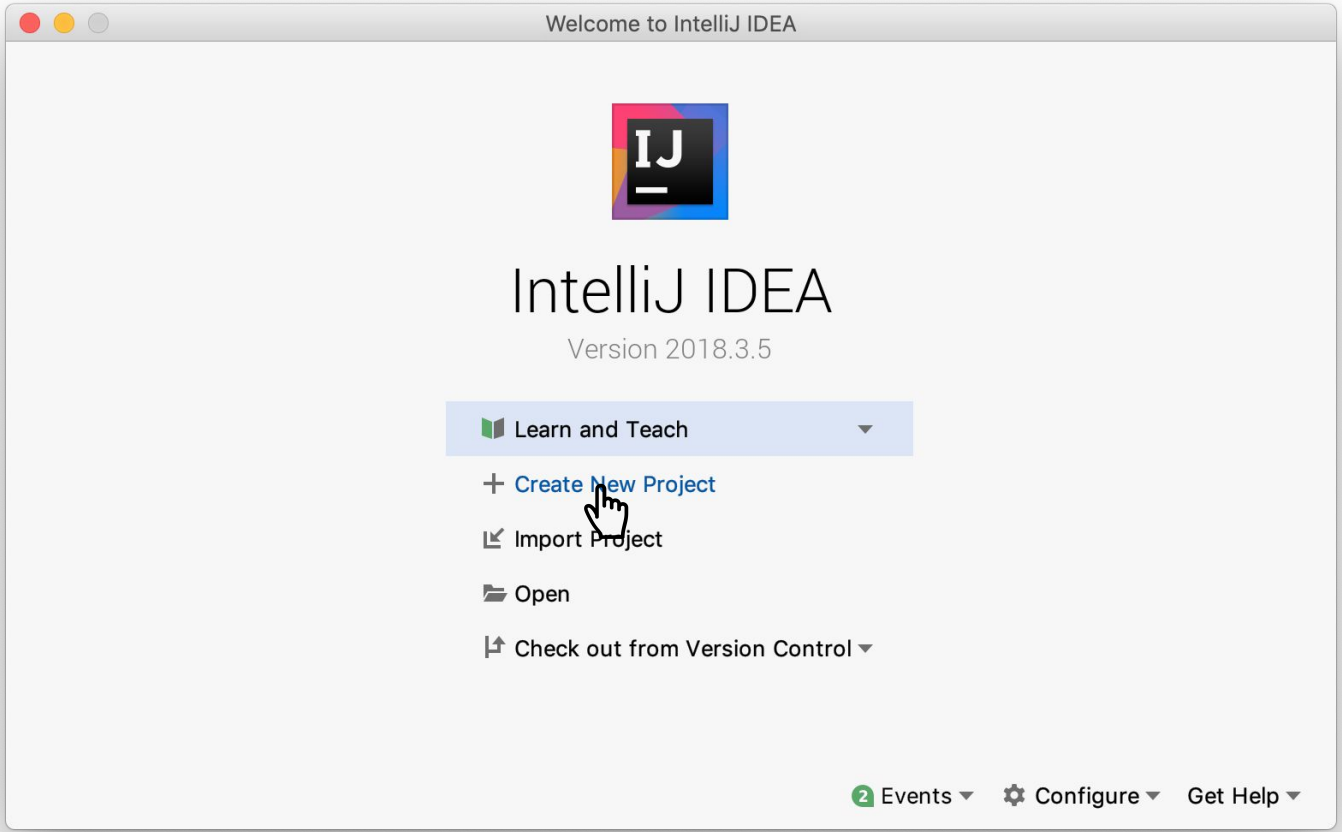
Table of Contents

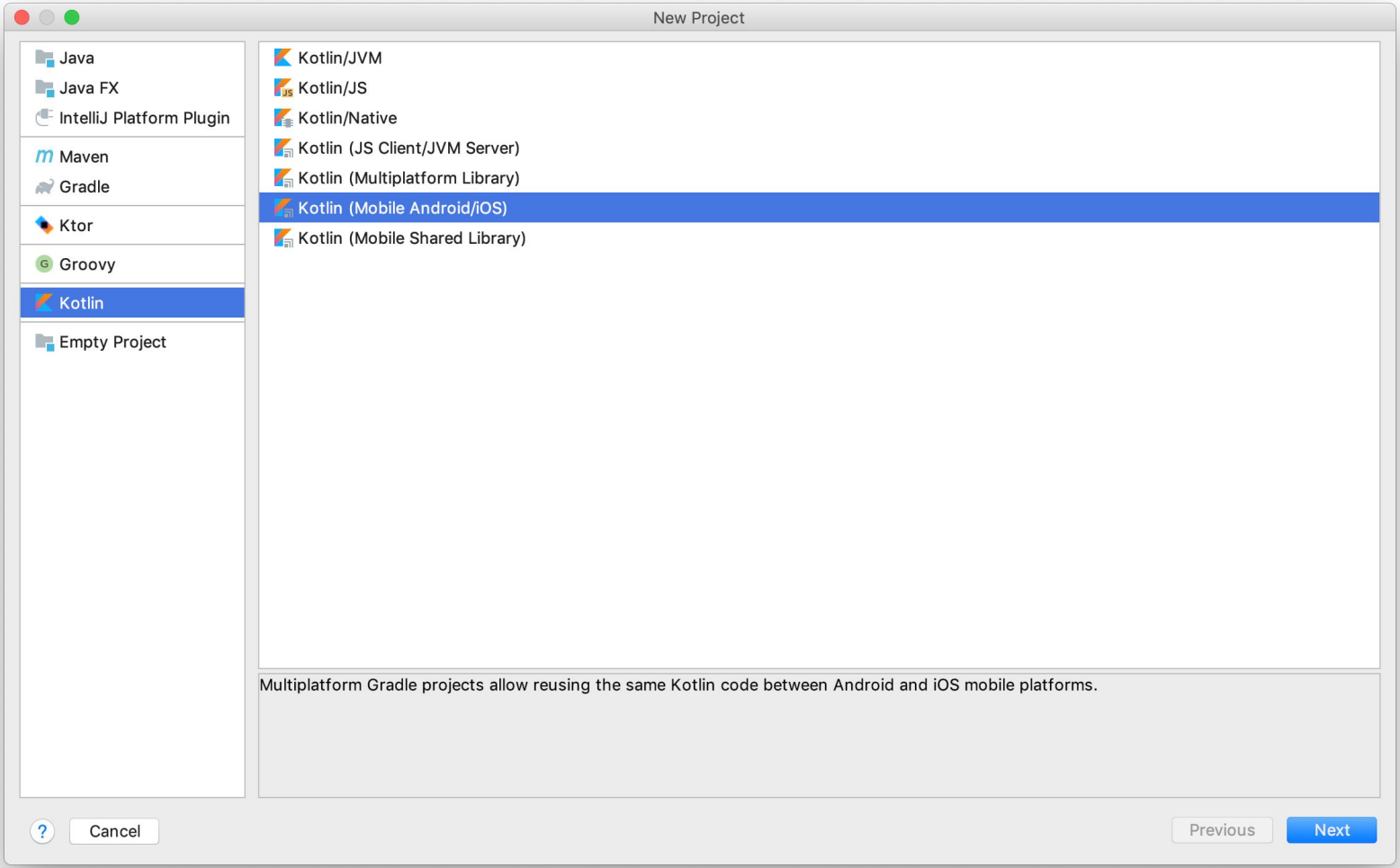


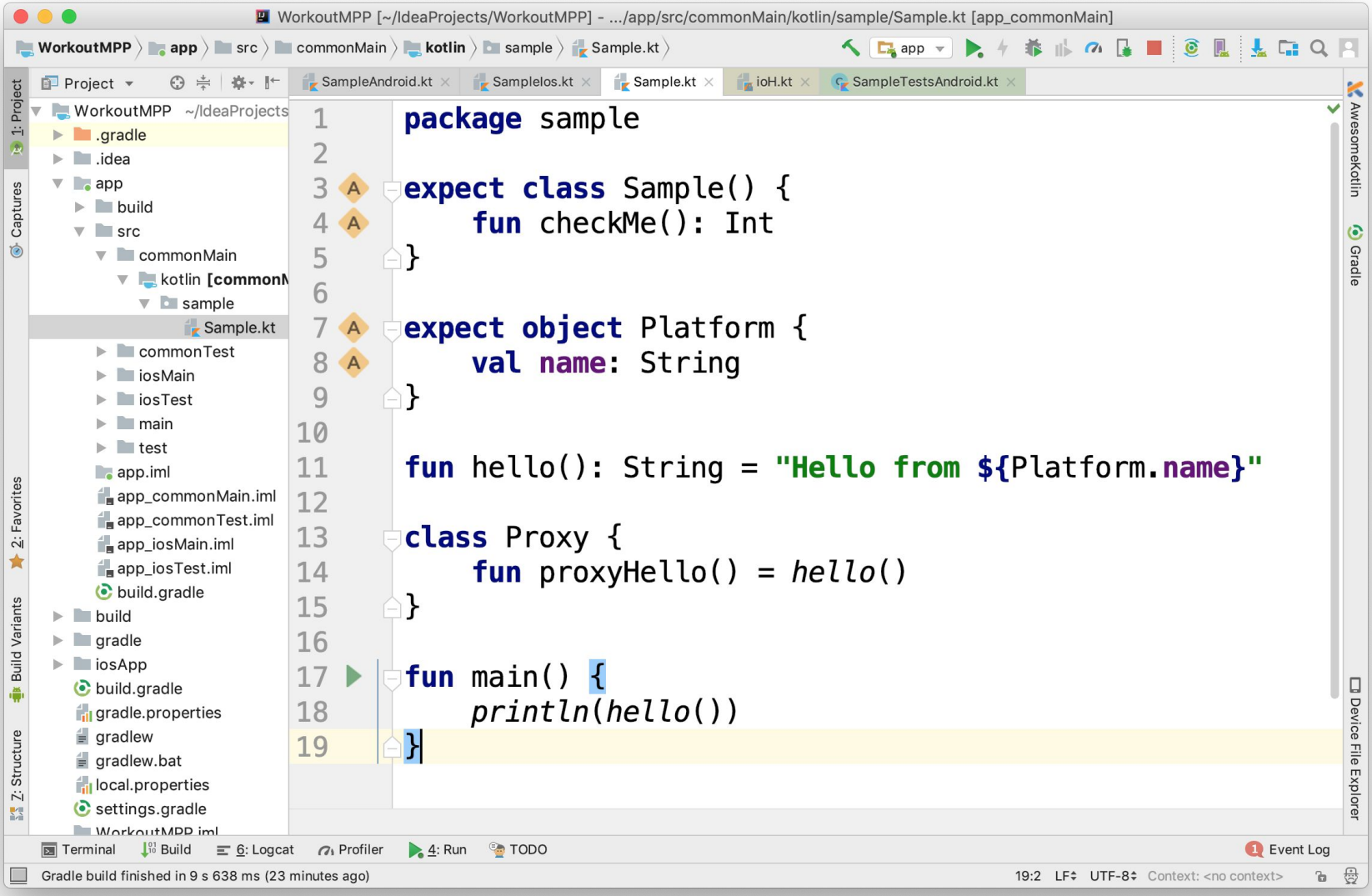
https://kotlinlang.org/docs/reference/building-mpp-with-gradle.html

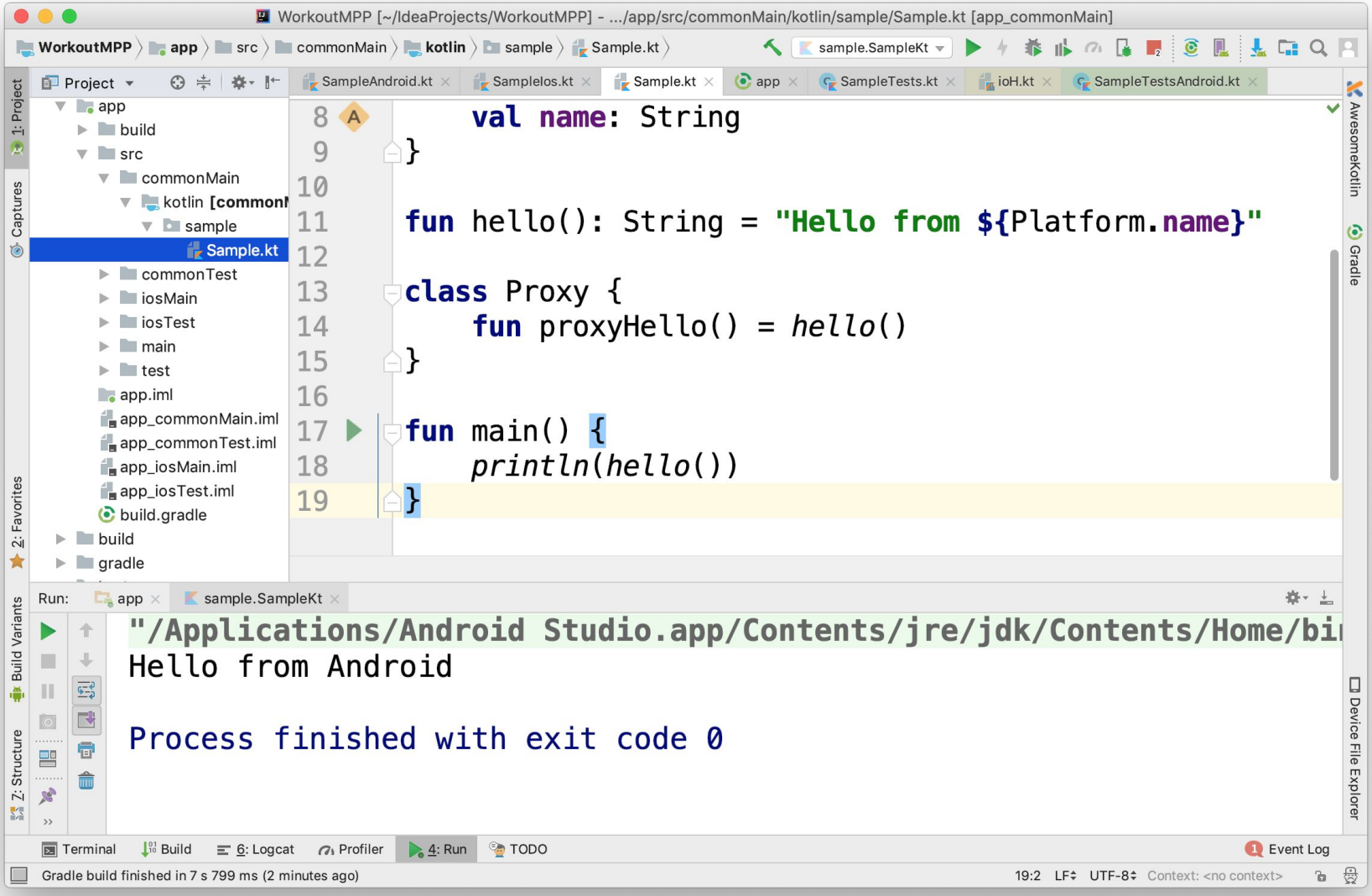


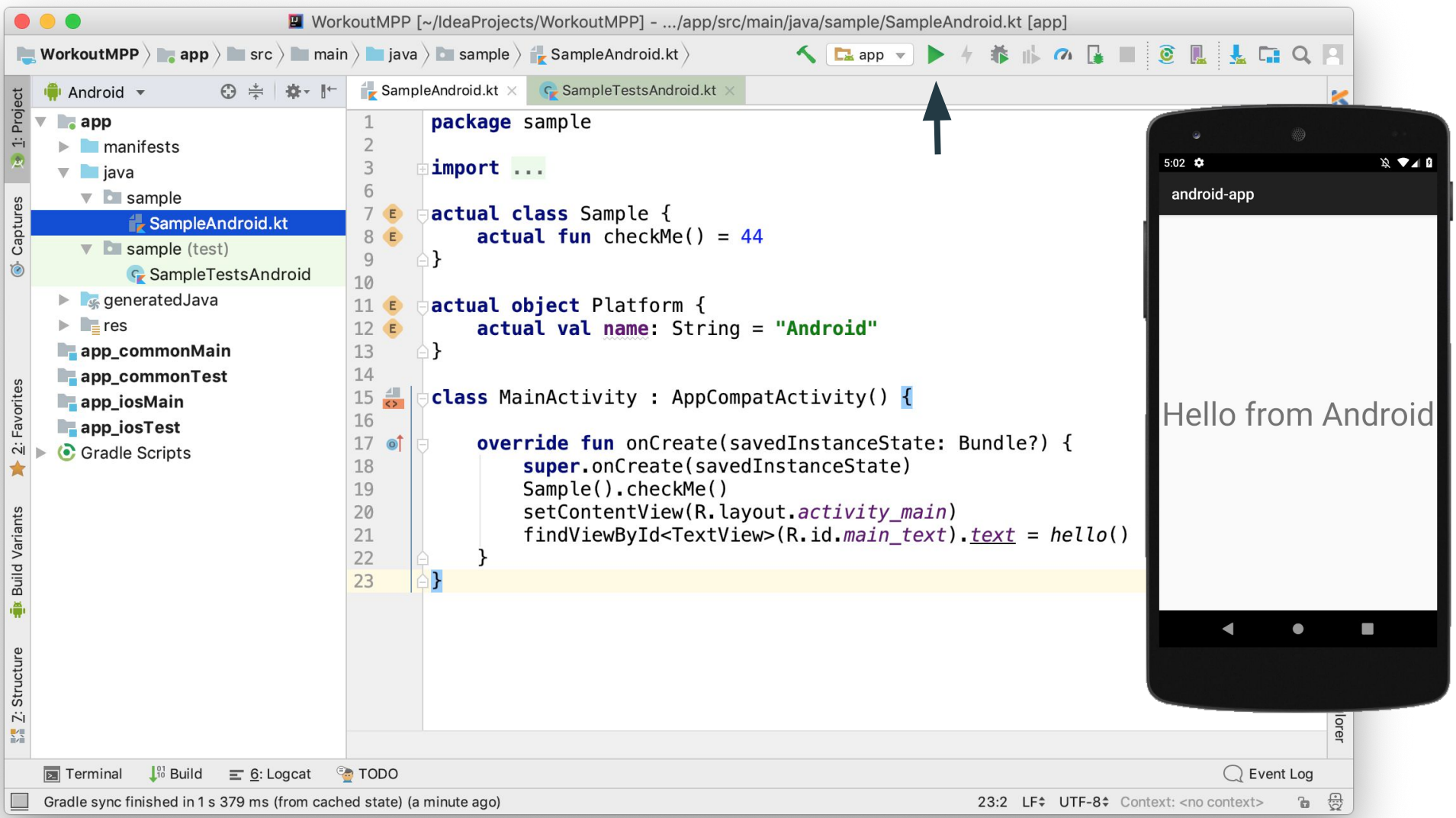
https://kotlinlang.org/docs/reference/building-mpp-with-gradle.html


Welcome to IntelliJ IDEA
IntelliJ IDEA
Version 2018.3.5
Learn and Teach
Create New Project
Import Project
Open
Check out from Version Control
2 Events
Configure
Get Help


New Project
Java
Java FX
IntelliJ Platform Plugin
Maven
Gradle
Ktor
Groovy
Kotlin
Empty Project
Kotlin/JVM
Kotlin/JS
Kotlin/Native
Kotlin (JS Client/JVM Server)
Kotlin (Multiplatform Library)
Kotlin (Mobile Android/iOS)
Kotlin (Mobile Shared Library)
Multiplatform Gradle projects allow reusing the same Kotlin code between Android and iOS mobile platforms.
Cancel
Previous
Next

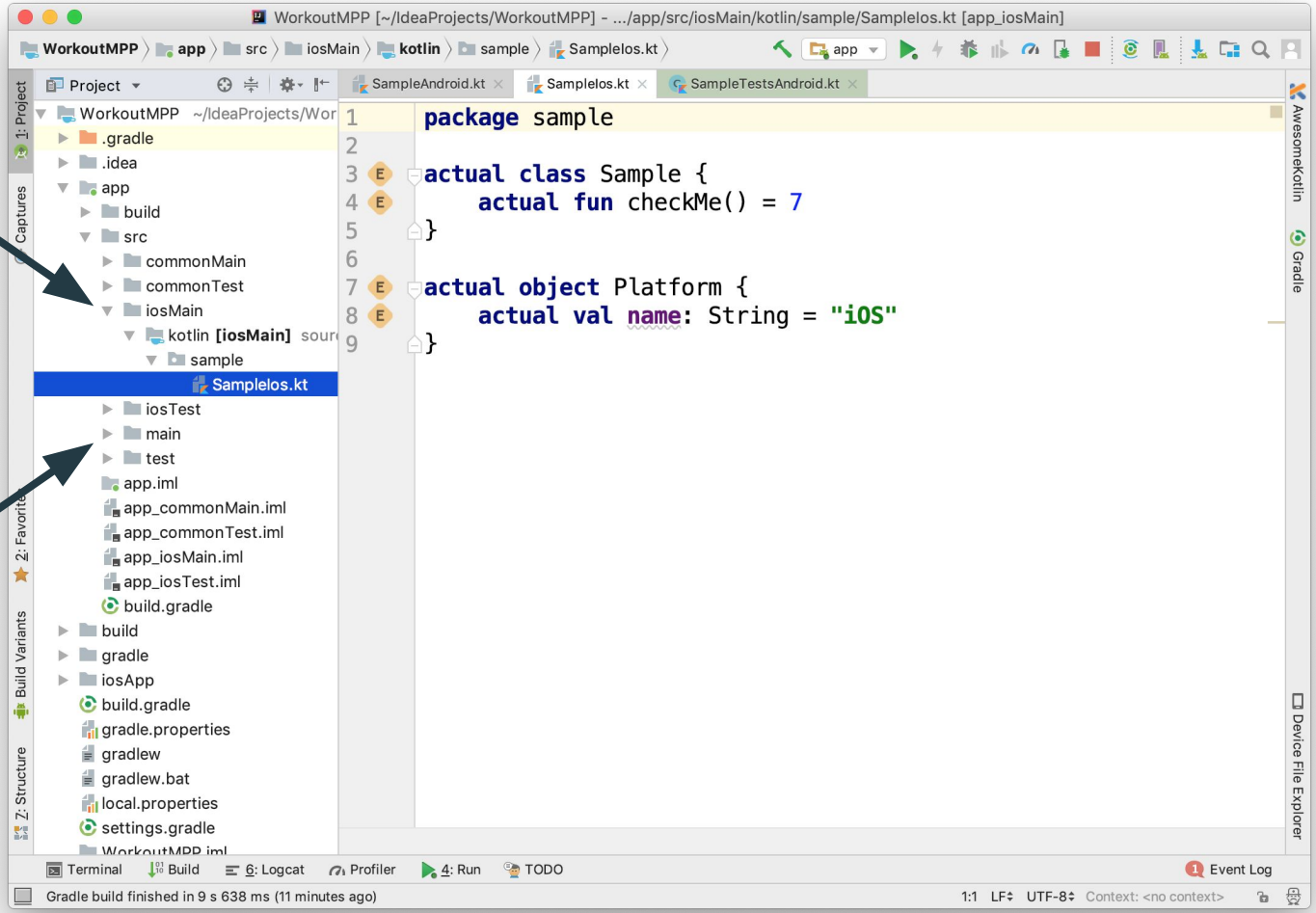

WorkoutMPP [~/IdeaProjects/WorkoutMPP] - .../app/src/commonMain/kotlin/sample/Sample.kt [app_commonMain]
WorkoutMPP app src commonMain kotlin sample Sample.kt
app
Project
SampleAndroid.kt
SampleIos.kt
Sample.kt
ioH.kt
SampleTestsAndroid.kt
WorkoutMPP ~/IdeaProjects
.gradle
.idea
app
build
src
commonMain
kotlin [commonM
sample
Sample.kt
commonTest
iosMain
iosTest
main
test
app.iml
app_commonMain.iml
app_commonTest.iml
app_iosMain.iml
app_iosTest.iml
build.gradle
build
gradle
iosApp
build.gradle
gradle.properties
gradlew
gradlew.bat
local.properties
settings.gradle
WorkoutMPP.iml
1 package sample
2
3 expect class Sample() {
4 fun checkMe(): Int
5 }
6
7 expect object Platform {
8 val name: String
9 }
10
11 fun hello(): String = "Hello from ${Platform.name}"
12
13 class Proxy {
14 fun proxyHello() = hello()
15 }
16
17 fun main() {
18 println(hello())
19 }
1: Project
Captures
2: Favorites
Build Variants
7: Structure
AwesomeKotlin
Gradle
Device File Explorer
Terminal
Build
6: Logcat
Profiler
4: Run
TODO
Event Log
Gradle build finished in 9 s 638 ms (23 minutes ago)
19:2 LF UTF-8 Context: <no context>

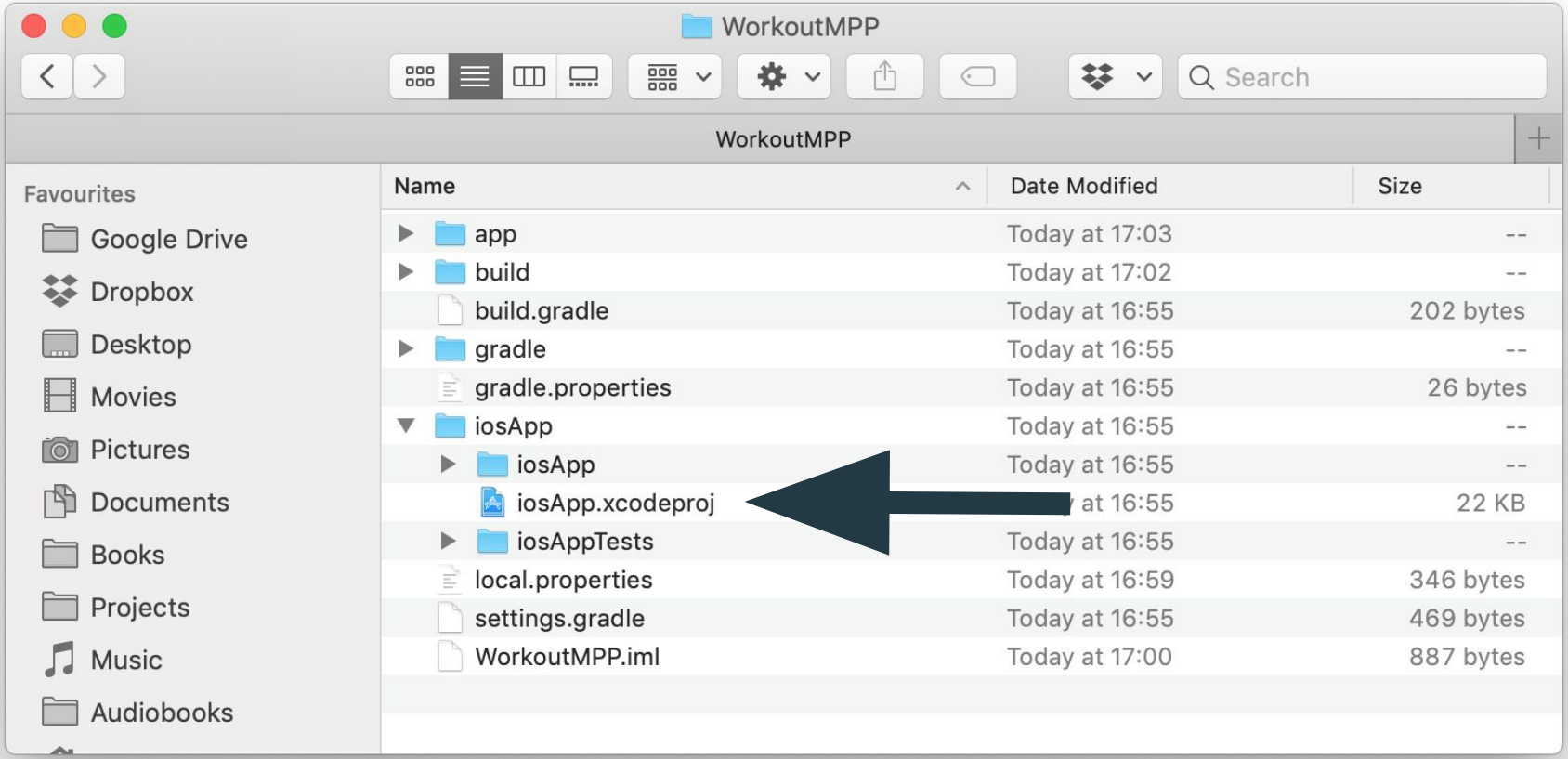

WorkoutMPP [~/IdeaProjects/WorkoutMPP] - .../app/src/commonMain/kotlin/sample/Sample.kt [app_commonMain]
WorkoutMPP app src commonMain kotlin sample Sample.kt
sample.SampleKt
Project
app
build
src
commonMain
kotlin [commonI
sample
Sample.kt
commonTest
iosMain
iosTest
main
test
app.iml
app_commonMain.iml
app_commonTest.iml
app_iosMain.iml
app_iosTest.iml
build.gradle
build
gradle
SampleAndroid.kt
SampleIos.kt
Sample.kt
app
SampleTests.kt
ioH.kt
SampleTestsAndroid.kt
8 A val name: String
9 }
10
11 fun hello(): String = "Hello from ${Platform.name}"
12
13 class Proxy {
14 fun proxyHello() = hello()
15 }
16
17 fun main() {
18 println(hello())
19 }
Run: app
sample.SampleKt
"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bi
Hello from Android
Process finished with exit code 0
1: Project
Captures
2: Favorites
Build Variants
Z: Structure
AwesomeKotlin
Gradle
Device File Explorer
Terminal
Build
6: Logcat
Profiler
4: Run
TODO
Event Log
Gradle build finished in 7 s 799 ms (2 minutes ago)
19:2 LF UTF-8 Context: <no context>


WorkoutMPP [~/IdeaProjects/WorkoutMPP] - .../app/src/main/java/sample/SampleAndroid.kt [app]
WorkoutMPP
app
src
main
java
sample
SampleAndroid.kt
app
Android
SampleAndroid.kt
SampleTestsAndroid.kt
app
manifests
java
sample
SampleAndroid.kt
sample (test)
SampleTestsAndroid
generatedJava
res
app_commonMain
app_commonTest
app_iosMain
app_iosTest
Gradle Scripts
1: Project
Captures
2: Favorites
Build Variants
Z: Structure
package sample
import ...
actual class Sample {
actual fun checkMe() = 44
}
actual object Platform {
actual val name: String = "Android"
}
class MainActivity : AppCompatActivity() {
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
Sample().checkMe()
setContentView(R.layout.activity_main)
findViewById<TextView>(R.id.main_text).text = hello()
}
}
5:02
android-app
Hello from Android
Terminal
Build
6: Logcat
TODO
Event Log
Gradle sync finished in 1 s 379 ms (from cached state) (a minute ago)
23:2
LF
UTF-8
Context: <no context>

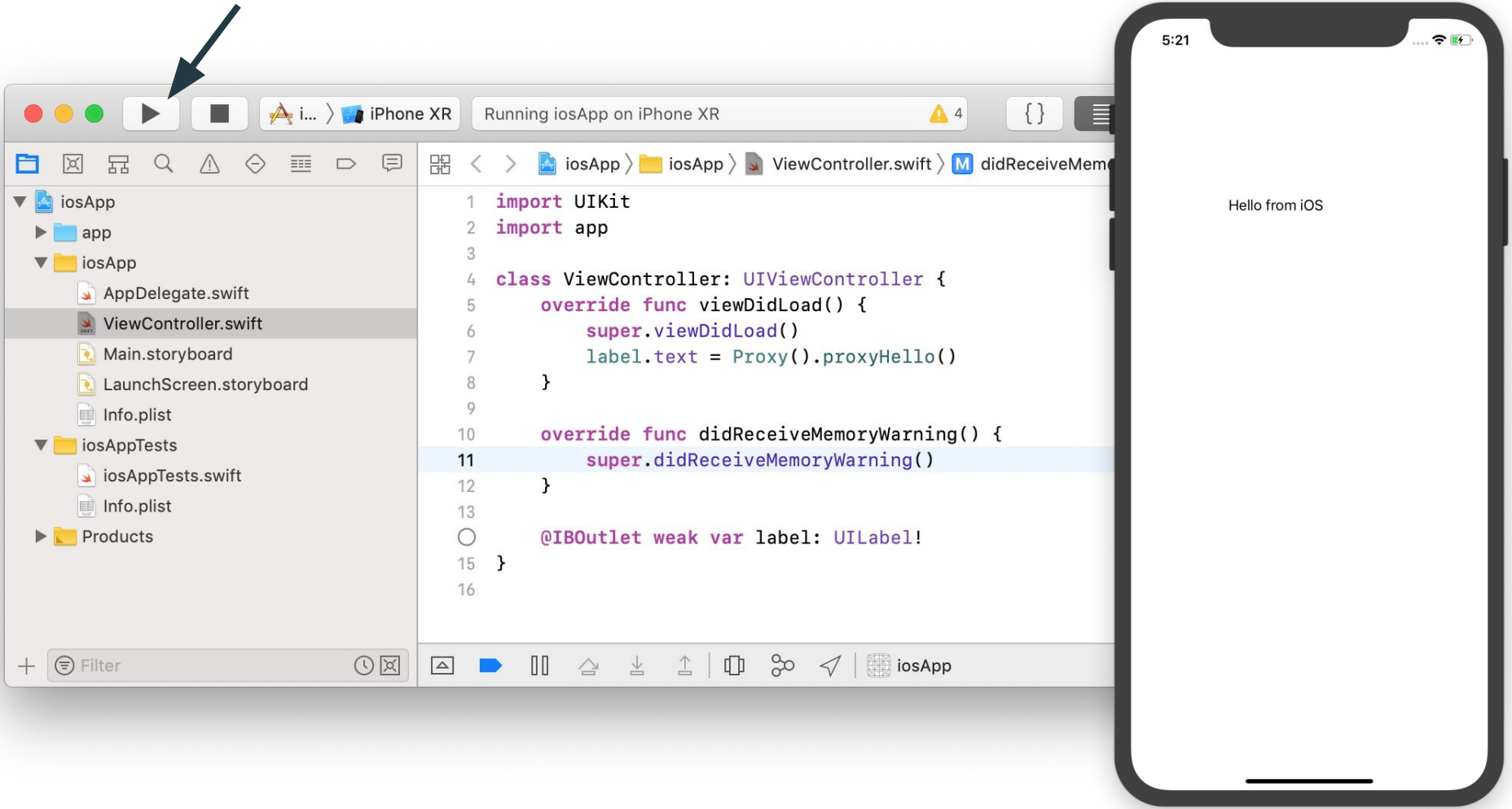
iOS

Android


WorkoutMPP [~/IdeaProjects/WorkoutMPP] - .../app/src/iosMain/kotlin/sample/Samplelos.kt [app_iosMain]
WorkoutMPP
app
src
iosMain
kotlin
sample
Samplelos.kt
SampleAndroid.kt
Samplelos.kt
SampleTestsAndroid.kt
Project
WorkoutMPP ~/IdeaProjects/Wor
.gradle
.idea
app
build
src
commonMain
commonTest
iosMain
kotlin [iosMain] sour
sample
Samplelos.kt
iosTest
main
test
app.iml
app_commonMain.iml
app_commonTest.iml
app_iosMain.iml
app_iosTest.iml
build.gradle
build
gradle
iosApp
build.gradle
gradle.properties
gradlew
gradlew.bat
local.properties
settings.gradle
WorkoutMPP.iml
1 package sample
2
3 actual class Sample {
4 actual fun checkMe() = 7
5 }
6
7 actual object Platform {
8 actual val name: String = "iOS"
9 }
1: Project
Captures
2: Favorites
Build Variants
Z: Structure
AwesomeKotlin
Gradle
Device File Explorer
Terminal
Build
6: Logcat
Profiler
4: Run
TODO
Event Log
Gradle build finished in 9 s 638 ms (11 minutes ago)
1:1
LF
UTF-8
Context: <no context>

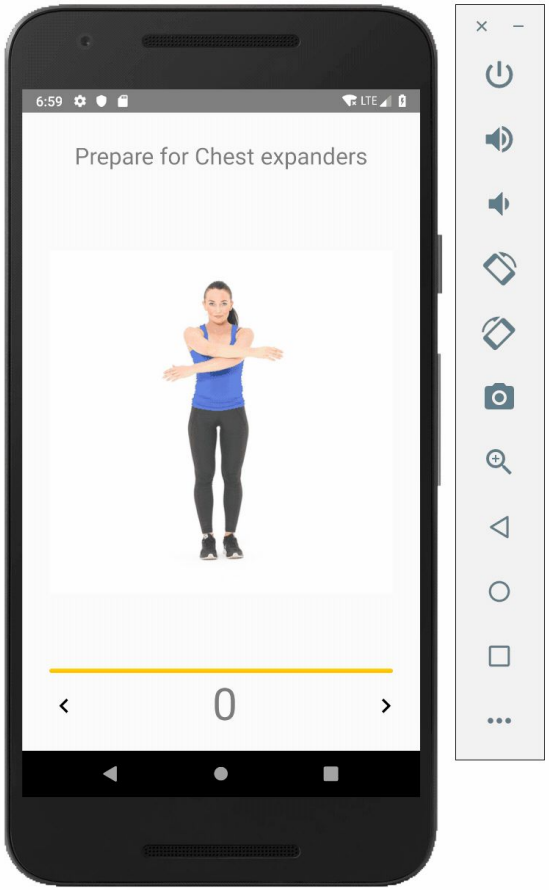

WorkoutMPP
Search
WorkoutMPP
Favourites
Google Drive
Dropbox
Desktop
Movies
Pictures
Documents
Books
Projects
Music
Audiobooks
Name
Date Modified
Size
app
Today at 17:03
--
build
Today at 17:02
--
build.gradle
Today at 16:55
202 bytes
gradle
Today at 16:55
--
gradle.properties
Today at 16:55
26 bytes
iosApp
Today at 16:55
--
iosApp
Today at 16:55
--
iosApp.xcodeproj
at 16:55
22 KB
iosAppTests
Today at 16:55
--
local.properties
Today at 16:59
346 bytes
settings.gradle
Today at 16:55
469 bytes
WorkoutMPP.iml
Today at 17:00
887 bytes

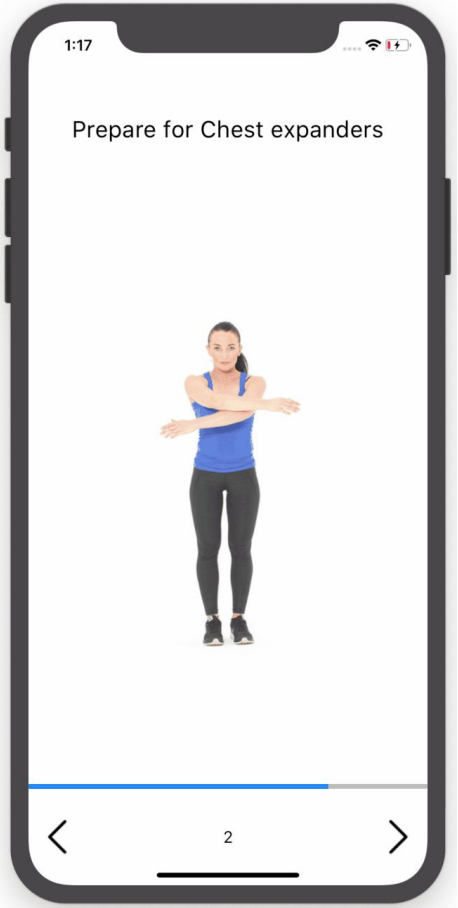
```
MacBook-Pro-Marcin:WorkoutMPP marcin$ gradle wrapper

> Configure project :app
Kotlin Multiplatform Projects are an experimental feature.

BUILD SUCCESSFUL in 1s
1 actionable task: 1 executed
```


i... 〉 iPhone XR
Running iosApp on iPhone XR
4
iosApp 〉 iosApp 〉 ViewController.swift 〉 M didReceiveMem
iosApp
app
iosApp
AppDelegate.swift
ViewController.swift
Main.storyboard
LaunchScreen.storyboard
Info.plist
iosAppTests
iosAppTests.swift
Info.plist
Products
Filter
1 import UIKit
2 import app
3
4 class ViewController: UIViewController {
5 override func viewDidLoad() {
6 super.viewDidLoad()
7 label.text = Proxy().proxyHello()
8 }
9
10 override func didReceiveMemoryWarning() {
11 super.didReceiveMemoryWarning()
12 }
13
15 @IBOutlet weak var label: UILabel!
15 }
16
iosApp
5:21
Hello from iOS

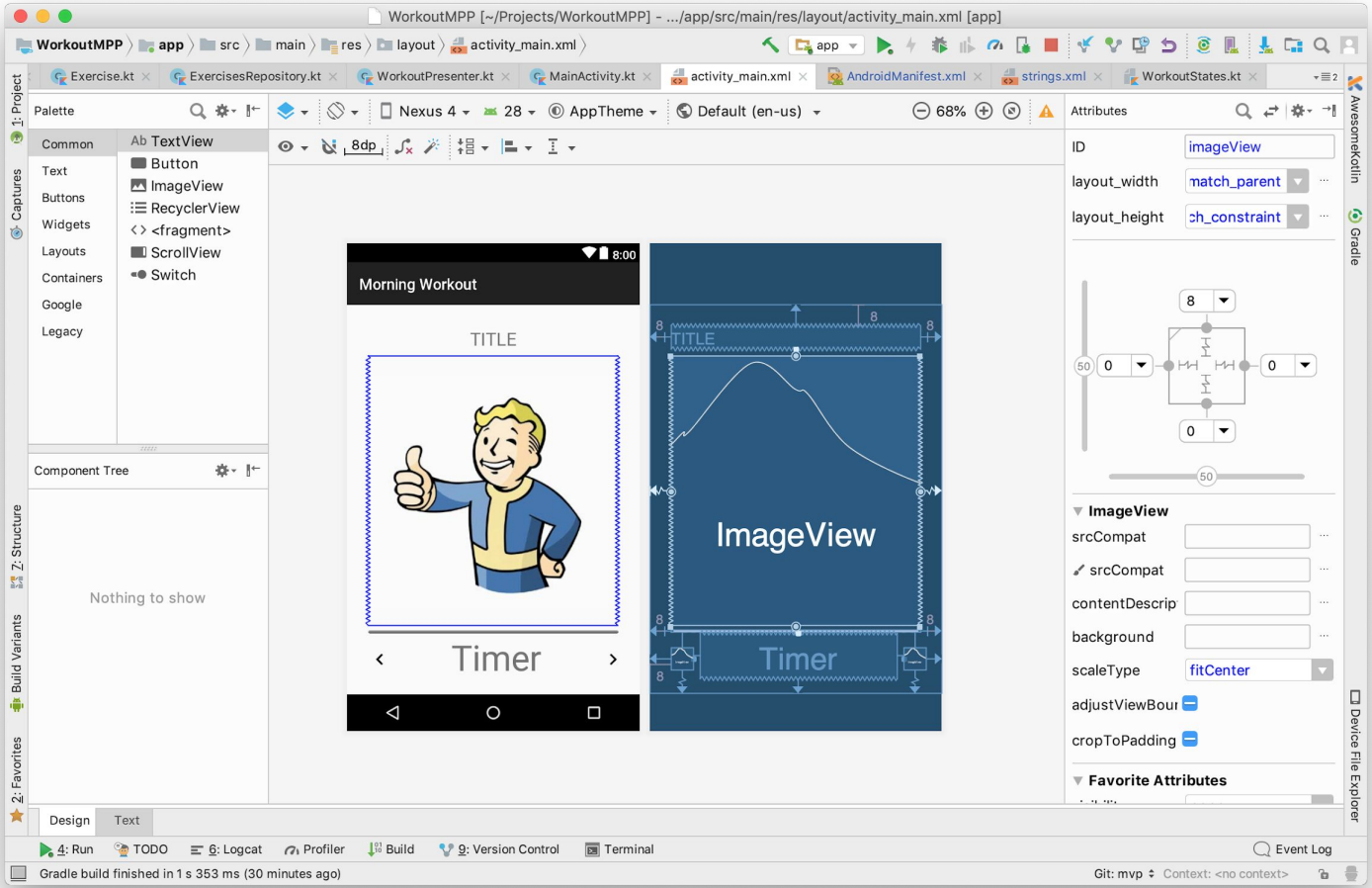
Make your first multiplatform project

Build MPP with shared logic

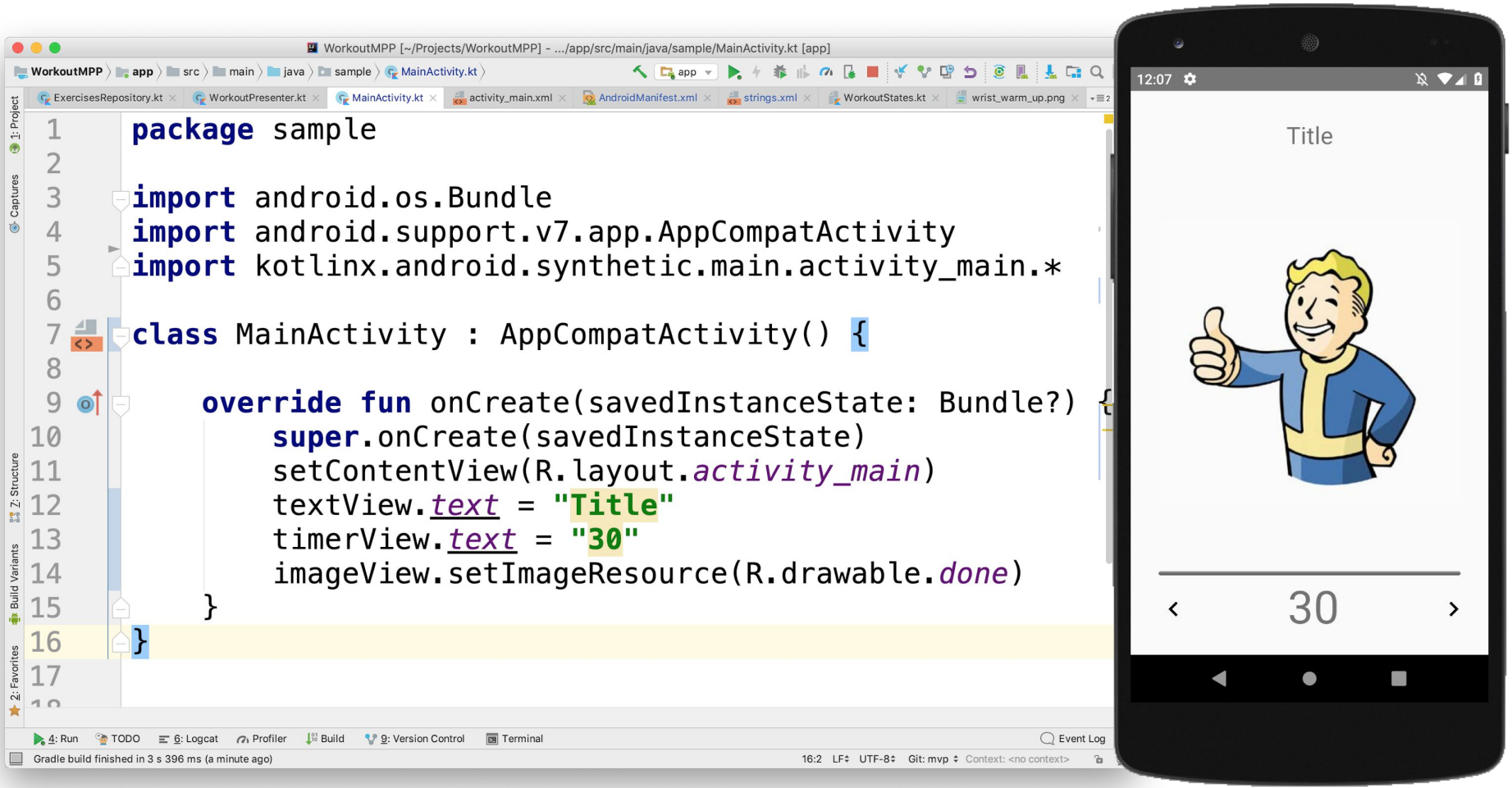

6:59
Prepare for Chest expanders
0


1:17
Prepare for Chest expanders
2

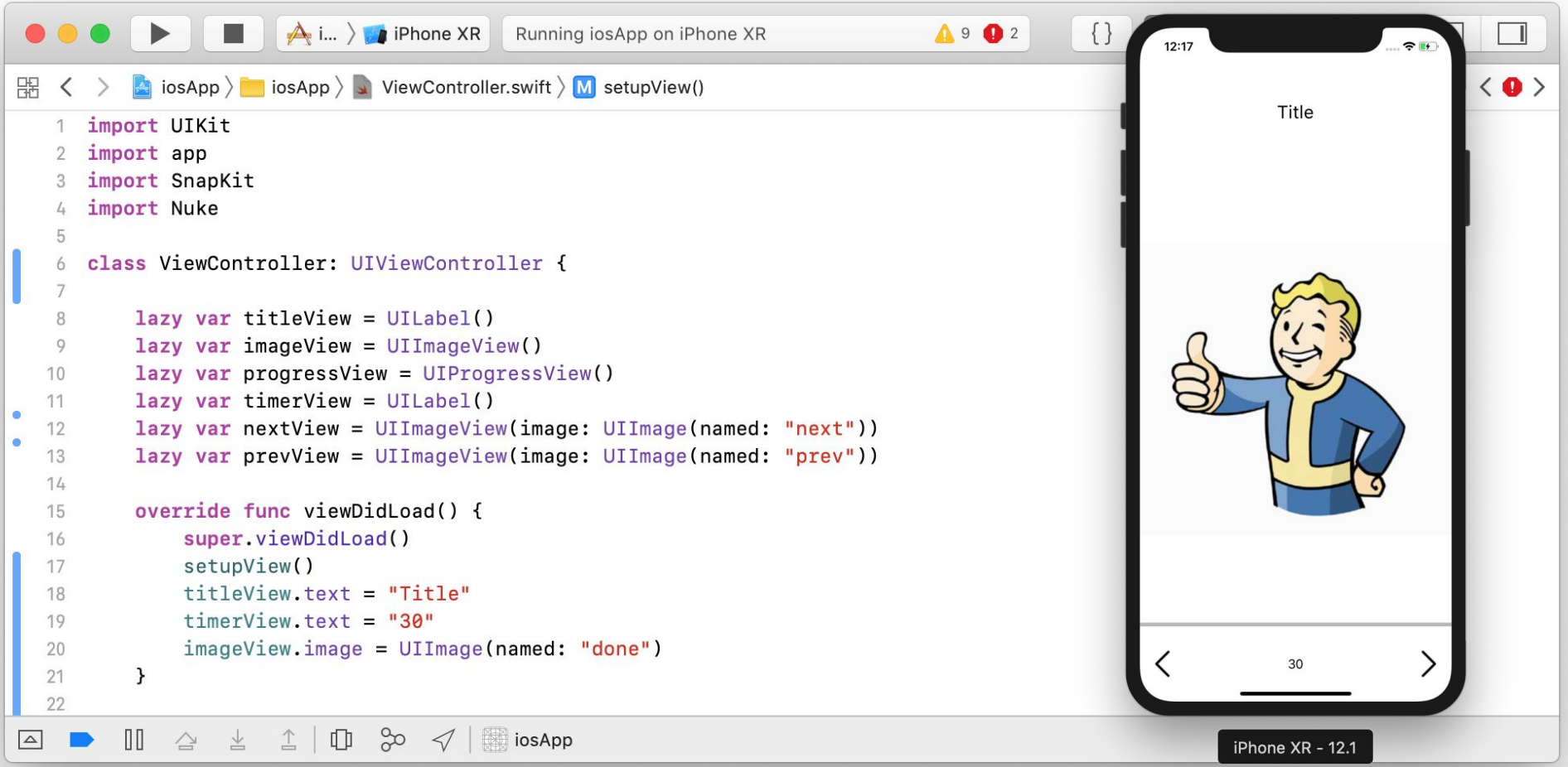
Step 1: Make views



https://github.com/MarcinMoskala/WorkoutMPP

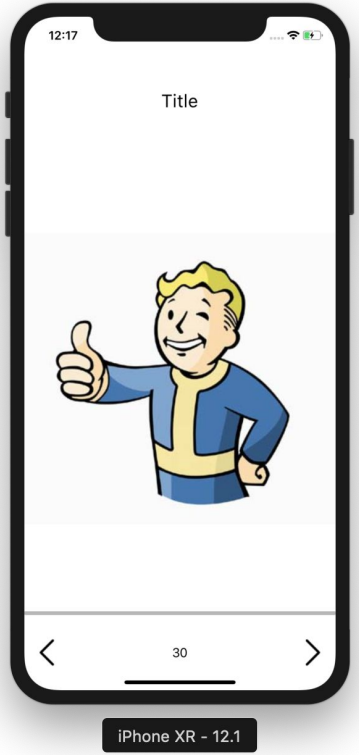


https://github.com/MarcinMoskala/WorkoutMPP



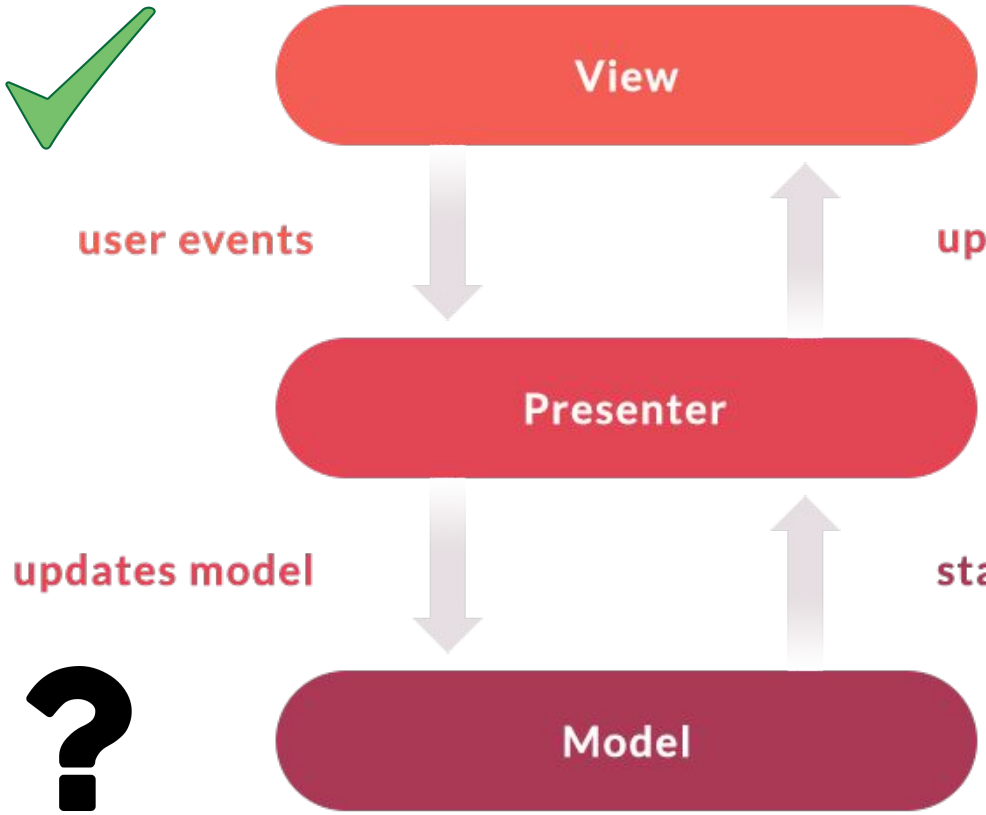
https://github.com/MarcinMoskala/WorkoutMPP

```
23      fileprivate func setupView() {
24          view.backgroundColor = UIColor.white
25
26          titleView.textAlignment = .center
27          titleView.font = UIFont(name: titleView.font.fontName, size: 24)
28          view.addSubview(titleView)
29
30          titleView.snp.makeConstraints { make in
31              make.right.left.equalToSuperview()
32              make.margins.equalTo(60)
33              make.height.equalTo(120)
34              make.margins.equalTo(16)
35          }
36
37          imageView.contentMode = .scaleAspectFit
38          view.addSubview(imageView)
39
40          imageView.snp.makeConstraints { make in
41              make.right.left.equalToSuperview()
42              make.top.equalTo(titleView.snp.bottom)
43              make.bottom.equalTo(progressView.snp.top)
44          }
```



https://github.com/SnapKit/SnapKit

Step 2: Implement logic



Presentation logic
How app looks like?

Business logic
How app behaves?

Other components
repositories, API, databases etc.

```
interface Speaker {
    fun speak(text: String)
}

interface Timer {
    fun start(
        seconds: Int,
        onTick: (secLeft: Int)->Unit,
        onFinish: ()->Unit
    )
    fun stop()
}
```

https://github.com/MarcinMoskala/WorkoutMPP

common

```
interface Speaker
interface Timer
```

Android

```
class AndroidSpeaker: Speaker
class AndroidTimer: Timer
```

iOS

```
class iOSSpeaker: Speaker
class iOSTimer: Timer
```

Android

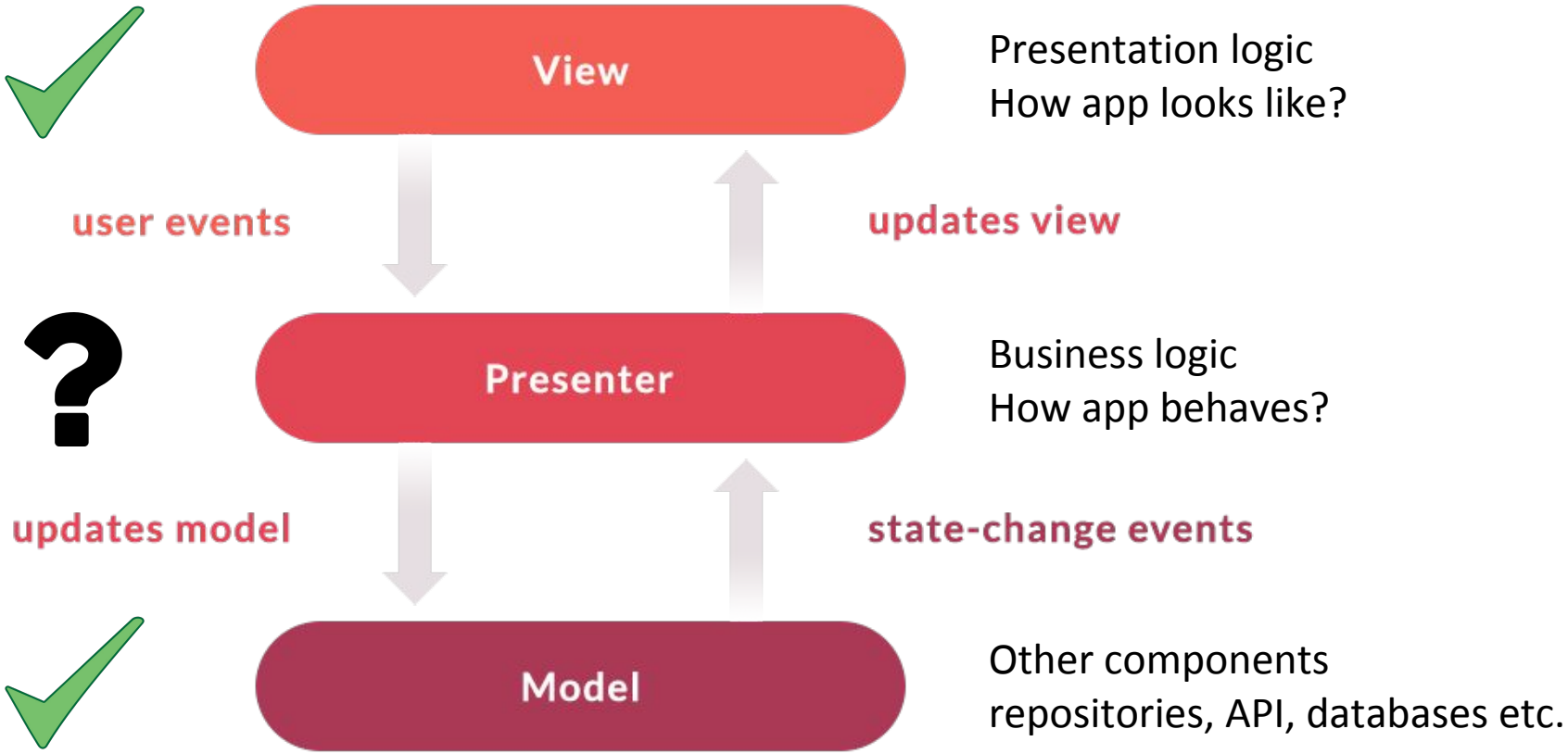
```
class AndroidSpeaker(context: Context) : Speaker {

    private var tts = TextToSpeech(context, null)

    override fun speak(text: String) {
        tts.speak(text, TextToSpeech.QUEUE_FLUSH, null, null)
    }
}
```

https://github.com/MarcinMoskala/WorkoutMPP

iOS

```
class iOSSpeaker: Speaker {
    private let synthesizer = AVSpeechSynthesizer()

    func speak(text: String) {
        synthesizer.stopSpeaking(at: .word)
        synthesizer.speak(AVSpeechUtterance(string: text))
    }
}
```

https://github.com/MarcinMoskala/WorkoutMPP


View
Presentation logic
How app looks like?
user events
updates view
Presenter
Business logic
How app behaves?
updates model
state-change events
Model
Other components
repositories, API, databases etc.

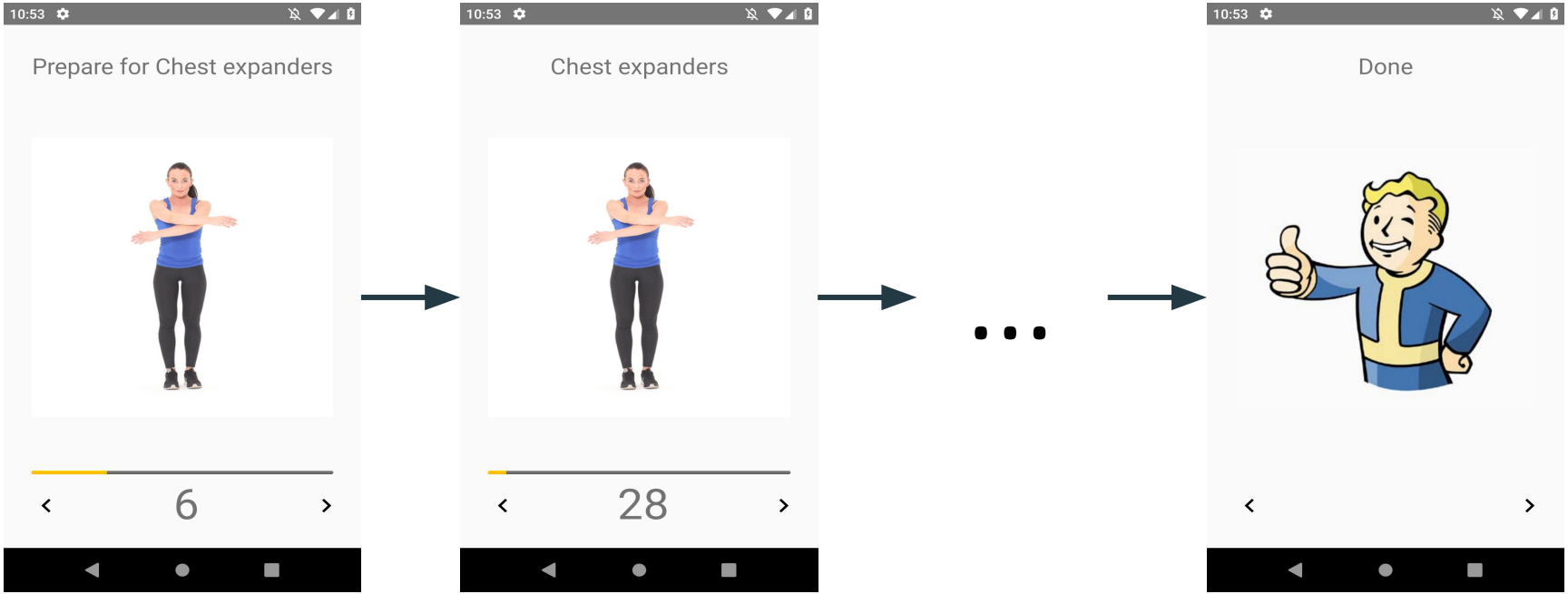
Presenter contract

```
class WorkoutPresenter(
    private val view: WorkoutView,
    private val timer: Timer,
    private val speaker: Speaker
) {
    fun onStart() { /*...*/ }
    fun onNext() { /*...*/ }
    fun onPrevious() { /*...*/ }
}

interface WorkoutView {
    fun setUpWorkoutDisplay(title: String, imgApiName: String)
    fun updateTimer(secLeft: Int, progress: Int)
    fun hideTimer()
}
```

https://github.com/MarcinMoskala/WorkoutMPP

```kotlin
data class Exercise(
    val imgUrlName: String,
    val nameText: String,
    val time: Int = 30
)

fun getExercises(): List<Exercise> = listOf(
    Exercise("chest_expander.png", "Chest expanders"),
    Exercise("rotating_toe_touches.jpg",
"Rotating Toe Touches"),
    Exercise("hips_rotation.png", "Hips rotation",
time = 20),
    //...
)
```

https://github.com/MarcinMoskala/WorkoutMPP


10:53
Prepare for Chest expanders
6
10:53
Chest expanders
28
10:53
Done
Prepare
Exercise
...
Done

```
sealed class WorkoutState

class PrepareState(val exercise: Exercise) : WorkoutState()

class ExerciseState(val exercise: Exercise) : WorkoutState()

object DoneState : WorkoutState()

fun List<Exercise>.toStates(): List<WorkoutState> =
    flatMap { exercise ->
      listOf(PrepareState(exercise), ExerciseState(exercise))
    } + DoneState
```

https://github.com/MarcinMoskala/WorkoutMPP

```
class WorkoutPresenter( /*...*/ ) {
    private val states: List<WorkoutState> = getExercises().toStates()
    private var state: WorkoutState = states.first()

    fun onStart() {
        showState(state)
    }

    fun onNext() {
        val nextState = states.getOrNull(states.indexOf(state) + 1)
        if (nextState != null) {
            state = nextState
            showState(state)
        }
    }

    private fun showState(state: WorkoutState) {}

    /*...*/
```

https://github.com/MarcinMoskala/WorkoutMPP

```
fun onNext() {
    val nextState = states.getOrNull(states.indexOf(state) + 1)
    if (nextState != null) {
        state = nextState
        showState(state)
    }
}

fun onPrevious() {
    val nextState = states.getOrNull(states.indexOf(state) - 1)
    if (nextState != null) {
        state = nextState
        showState(state)
    }
}
```

https://github.com/MarcinMoskala/WorkoutMPP

```
private fun showState(state: WorkoutState) {
    val titleText = when (state) {
        is PrepareState -> "Prepare for " + state.exercise.nameText
        is ExerciseState -> state.exercise.nameText
        is DoneState -> "Done"
    }
    val imgApiName = when (state) {
        is PrepareState -> state.exercise.imgUrlName
        is ExerciseState -> state.exercise.imgUrlName
        is DoneState -> "done.jpg"
    }
    view.setUpWorkoutDisplay(titleText, imgApiName)
    speaker.speak(titleText)
    setUpTimer(state)
}
```

https://github.com/MarcinMoskala/WorkoutMPP

```
private fun setUpTimer(state: WorkoutState) {
    val durationSec = when (state) {
        DoneState -> {
            view.hideTimer()
            timer.stop()
            return
        }
        is ExerciseState -> state.exercise.time
        is PrepareState -> EXERCISE_PREPARE_TIME
    }

    timer.start(
        durationSec,
        onTick = { secLeft ->
            val progress = 100 * (durationSec - secLeft) / durationSec
            view.updateTimer(secLeft = secLeft, progress = progress)
        },
        onFinish = this::onNext
    )
}
```

https://github.com/MarcinMoskala/WorkoutMPP

Step 3: Use presenter

```
class MainActivity : AppCompatActivity(), WorkoutView {

    private val presenter by lazy {
        WorkoutPresenter(this, AndroidTimer(), AndroidSpeaker(this))
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        nextButton.setOnClickListener { presenter.onNext() }
        prevButton.setOnClickListener { presenter.onPrevious() }
        presenter.onStart()
    }

    //...
}
```

https://github.com/MarcinMoskala/WorkoutMPP

```
override fun setUpWorkoutDisplay(title: String, imgApiName: String) {
    textView.text = title
    imageView.loadImage(url = "$BASE_URL/images/$imgApiName")
}

override fun hideTimer() {
    timerView.text = ""
    progressBar.progress = 0
}

override fun updateTimer(secLeft: Int, progress: Int) {
    timerView.text = "$secLeft"
    progressBar.progress = progress
}
```

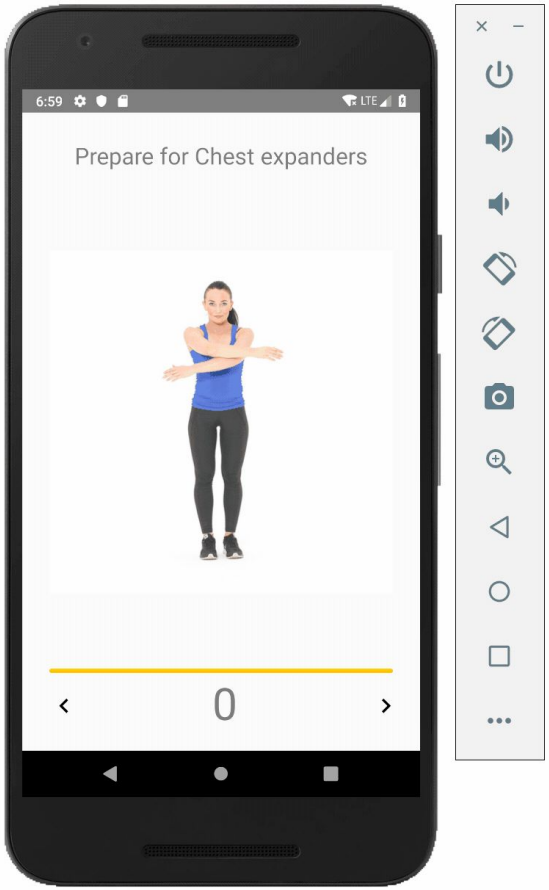
https://github.com/MarcinMoskala/WorkoutMPP

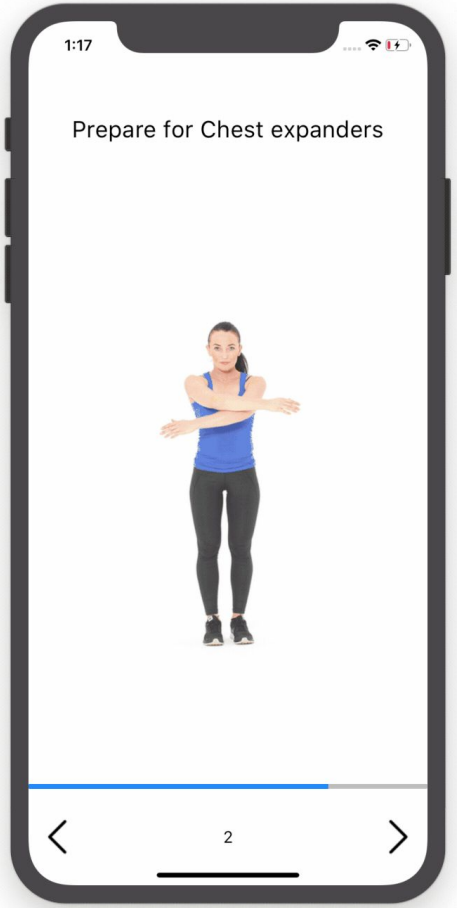
```
class ViewController: UIViewController, WorkoutView {

    private lazy var presenter = WorkoutPresenter(view: self,
        timer: iOSTimer(), speaker: iOSSpeaker())

    // UI views

    override func viewDidLoad() {
        super.viewDidLoad()
        setupView()
        nextView.addTapGestureRecognizer { self.presenter.onNext() }
        prevView.addTapGestureRecognizer { self.presenter.onPrevious() }
        presenter.onStart()
    }

    //...
}
```

https://github.com/MarcinMoskala/WorkoutMPP

```
func setUpWorkoutDisplay(title: String, imgApiName: String) {
    titleView.text = title
    imageView.loadImage(url: BASE_URL + "/images/" + imgApiName)
}

func hideTimer() {
    timerView.text = ""
    progressView.progress = 0
}

func updateTimer(secLeft: Int32, progress: Int32) {
    timerView.text = String(secLeft)
    progressView.progress = Float(progress) / 100
}
```

https://github.com/MarcinMoskala/WorkoutMPP


6:59
Prepare for Chest expanders
0


1:17
Prepare for Chest expanders
2

```
@Test
fun `Check preparation start, first tick and change to next`() {
    val timer = FakeTimer()
    val speaker: Speaker = mockk(relaxed = true)
    val view: WorkoutView = mockk(relaxed = true)
    val presenter = WorkoutPresenter(view, timer, speaker)

    presenter.onStart()
    timer.onTick(prepTime - 1)
    presenter.onNext()

    val prepareText = "Prepare for " + firstExercise.nameText
    verify {
        view.setUpWorkoutDisplay(prepareText, firstExercise.imgUrlName)
        speaker.speak(prepareText)
        view.updateTimer(prepTime, 0)
        view.updateTimer(prepTime - 1, 100 / prepTime)

        view.setUpWorkoutDisplay(firstExercise.nameText, firstExercise.imgUrlName)
        speaker.speak(firstExercise.nameText)
        view.updateTimer(firstExercise.time, 0)
    }
}
```

https://github.com/MarcinMoskala/WorkoutMPP

Build MPP with shared logic

What's next?

- Common API (Ktor client)
- Multithreading (Coroutines)
- MVVM
- View Binding

Kotlin Conf application
https://github.com/
JetBrains/kotlinconf-app

MVVM

```
class WorkoutViewModel(
    private val view: WorkoutView,
    private val timer: Timer,
    private val speaker: Speaker
) {
    //...

    val titleProp = MutableProp("")
    val imgApiUrlProp = MutableProp("")
    val progressProp = MutableProp(0)
    val timerTextProp = MutableProp("")

    fun onStart() {
        showState()
    }

    //...
```

https://github.com/MarcinMoskala/WorkoutMPP

```
class MutableProp<T>(private var current: T) {
    private var listeners = listOf<(T)->Unit>()

    fun get(): T = current

    fun set(elem: T) {
        current = elem
        listeners.forEach { it(elem) }
    }

    fun addListener(listener: (T)->Unit) {
        listeners = listeners + listener
        listener(current)
    }
}
```

https://github.com/MarcinMoskala/WorkoutMPP

```
private fun showState(state: WorkoutState) {
    val titleText = when (state) {
        is PrepareState -> "Prepare for " + state.exercise.nameText
        is ExerciseState -> state.exercise.nameText
        is DoneState -> "Done"
    }
    val imgApiName = when (state) {
        is PrepareState -> state.exercise.imgUrlName
        is ExerciseState -> state.exercise.imgUrlName
        is DoneState -> "done.jpg"
    }
    view.setUpWorkoutDisplay(titleText, imgApiName)
    imgApiUrlProp.set(imgApiName)
    speaker.speak(titleText)
    setUpTimer(state)
}
```

https://github.com/MarcinMoskala/WorkoutMPP

```
view.setUpWorkoutDisplay(
   titleText, imgApiName)
```

```
titleProp.set(titleText)
imgApiUrlProp.set(imgApiName)
```

```
view.hideTimer()
```

```
timerTextProp.set("")
progressProp.set(0)
```

```
view.updateTimer(
    secLeft, progress)
```

```
timerTextProp.set("$secLeft")
progressProp.set(progress)
```

```
class MainActivity : AppCompatActivity() {
    private val viewModel by lazy { WorkoutViewModel(AndroidTimer(),
AndroidSpeaker(this)) }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        viewModel.apply {
            titleProp.addListener { titleView.text = it }
            timerTextProp.addListener { timerView.text = it }
            imgApiUrlProp.addListener {
                imageView.loadImage(url = "$BASE_URL/images/$it")
            }
            progressProp.addListener { progressBar.progress = it }
        }
        nextButton.setOnClickListener { viewModel.onNext() }
        prevButton.setOnClickListener { viewModel.onPrevious() }
        viewModel.onStart()
    }
}
```

https://github.com/MarcinMoskala/WorkoutMPP

Problem with Kotlin/Native

Any?

```
func setUpViewModel() {
    viewModel.titleProp.addListener { text in
        self.titleView.text = text as! String
    }
    //...
}
```

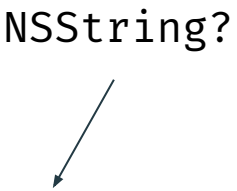
app/build.gradle

```
kotlin {
    iosX64("ios") {
        binaries {
            framework {
                freeCompilerArgs += "-Xobjc-generics"
            }
        }
    }
}
```

https://github.com/MarcinMoskala/WorkoutMPP

```
compilations.main {
    outputKinds("framework")
    extraOpts "-Xobjc-generics"
}
```

Problem with Kotlin/Native




```
func setUpViewModel() {
    viewModel.titleProp.addListener { (text) in
        self.titleView.text = text! as String
    }
    //...
}
```

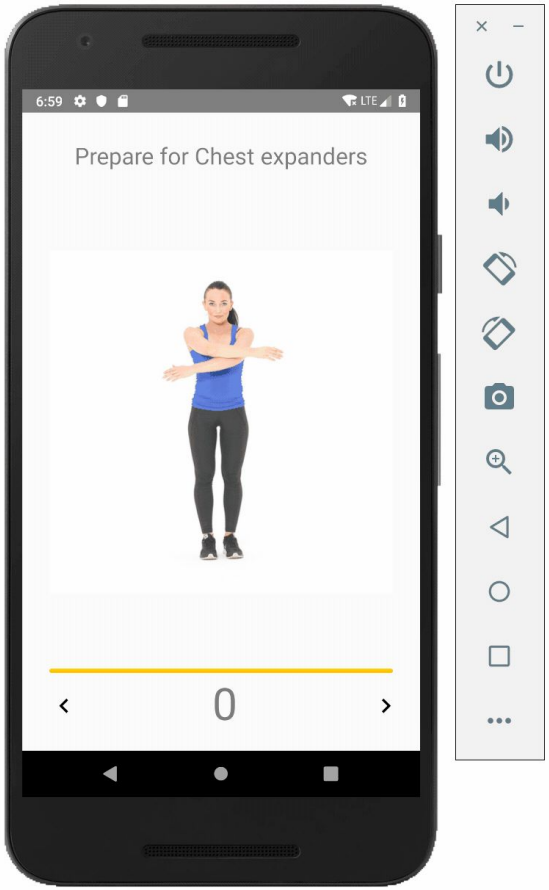
https://github.com/MarcinMoskala/WorkoutMPP

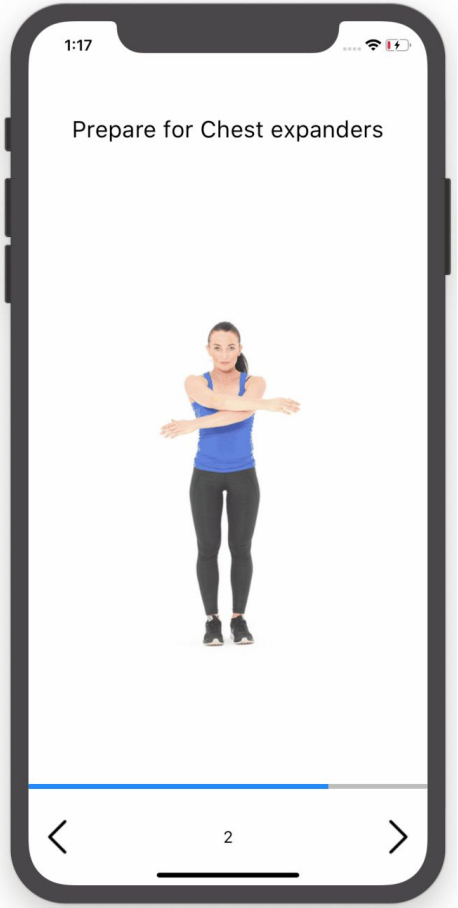
```
extension MutableProp {
    func addListenerTyped<T>(listener: @escaping (T)->()) {
        addListener { elem in
            listener(elem as! T)
            return KotlinUnit()
        }
    }
}
```

https://github.com/MarcinMoskala/WorkoutMPP

```
func setUpViewModel() {
    viewModel.titleProp.addListenerTyped { (text: String) in
        self.titleView.text = text
    }
    viewModel.timerTextProp.addListenerTyped { (text: String) in
        self.timerView.text = text
    }
    viewModel.imgApiUrlProp.addListenerTyped { (url: String) in
        self.imageView.loadImage(url: BASE_URL + "/images/" + url)
    }
    viewModel.progressProp.addListenerTyped { (progress: Int) in
        self.progressView.progress = Float(progress) / 100
    }
}
```

https://github.com/MarcinMoskala/WorkoutMPP


6:59
Prepare for Chest expanders
0


1:17
Prepare for Chest expanders
2

```
@Test fun singleExerciseTimerTest() {
    val firstExercise = Exercise("chest_expander.png", "Chest expanders", 30)
    val timer = FakeTimer()
    val speaker = FakeSpeaker()
    val vm = WorkoutViewModel(timer, speaker, listOf(firstExercise))

    vm.onStart()
    val prepareText = "Prepare for " + firstExercise.nameText
    assertEquals(prepareText, vm.titleProp.get())
    assertEquals(prepareText, speaker.spokenTexts.last())
    assertEquals(0, vm.progressProp.get())
    val prepareTime = WorkoutViewModel.EXERCISE_PREPARE_TIME
    val preparationTime = prepareTime.toString()
    assertEquals(preparationTime, vm.timerTextProp.get())

    timer.tick()
    assertEquals(prepareText, vm.titleProp.get())
    assertEquals(prepareText, speaker.spokenTexts.last())
    assertEquals(100 / prepareTime, vm.progressProp.get())
    assertEquals((prepareTime - 1).toString(), vm.timerTextProp.get())
    //...
}
```

https://github.com/MarcinMoskala/WorkoutMPP

MVVM

View Binding

```
extension UILabel {
    func bindText(with prop: MutableProp) {
        prop.addListenerTyped { (value: String) in
            self.text = value
        }
    }
}

titleView.bindText(with: viewModel.titleProp)
timerView.bindText(with: viewModel.timerTextProp)

viewModel.imgApiUrlProp.addListenerTyped { (url: String) in
    self.imageView.loadImage(url: BASE_URL + "/images/" + url)
}
viewModel.progressProp.addListenerTyped { (progress: Int) in
    self.progressView.progress = Float(progress) / 100
}
```

```
<layout ...>
    <data>
        <variable name="vm" type="sample.WorkoutViewModel"/>
    </data>
    <android.support.constraint.ConstraintLayout>
        <TextView ... android:text="@{vm.titleProp}"/>
        <ImageView ... app:imageUrl="@{vm.imgUrlProp}"/>
        <ProgressBar ... android:progress="@{vm.progressProp}"/>
        <TextView ... android:text="@{vm.timerTextProp}"/>
        <ImageView ... android:onClick="@{()->vm.onNext()}"/>
        <ImageView ... android:onClick="@{()->vm.onPrevious()}"/>
    </android.support.constraint.ConstraintLayout>
</layout>
```

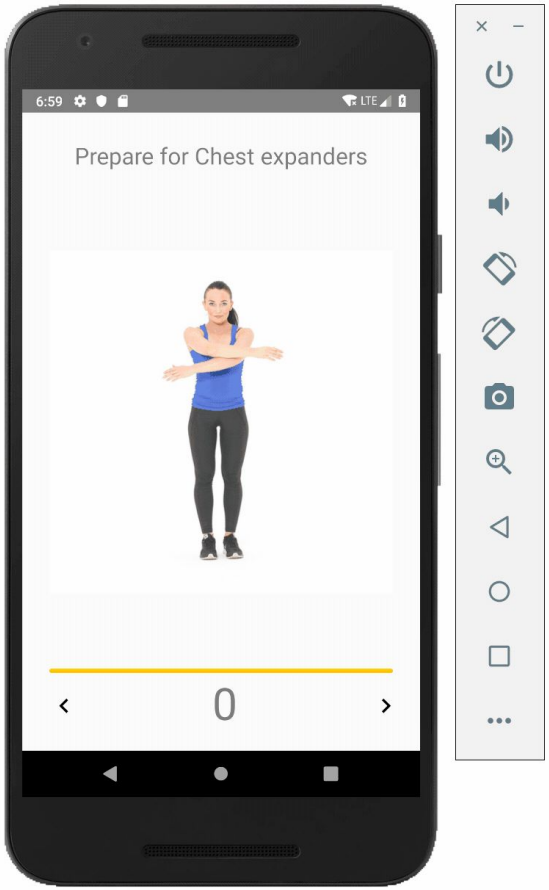
https://github.com/MarcinMoskala/WorkoutMPP

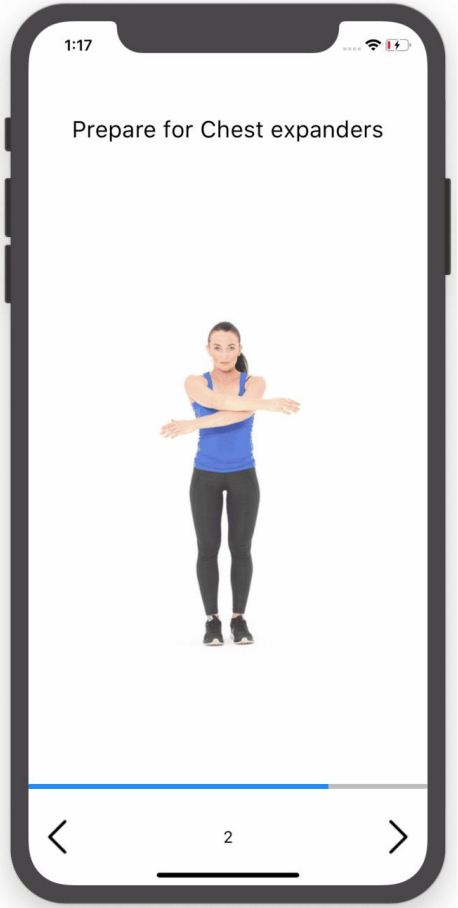
```
class WorkoutViewModel(/*...*/) : ViewModel() {

// Common
expect abstract class ViewModel()

expect class MutableProp<T>(current: T) {
    fun set(elem: T)
    fun get(): T
}

// Android
actual typealias ViewModel = android.arch.lifecycle.ViewModel
actual typealias MutableProp<T> = ObservableField<T>

// iOS
actual abstract class ViewModel actual constructor()
// Same MutableProp as before
```

https://github.com/MarcinMoskala/WorkoutMPP

```
class MainActivity : AppCompatActivity() {

    private val viewModel by lazy {
        WorkoutViewModel(AndroidTimer(), AndroidSpeaker(this))
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        DataBindingUtil.setContentView<ActivityMainBinding>(this,
                R.layout.activity_main)
            .vm = viewModel
        viewModel.onStart()
    }
}
```

https://github.com/MarcinMoskala/WorkoutMPP


6:59
Prepare for Chest expanders
0


1:17
Prepare for Chest expanders
2

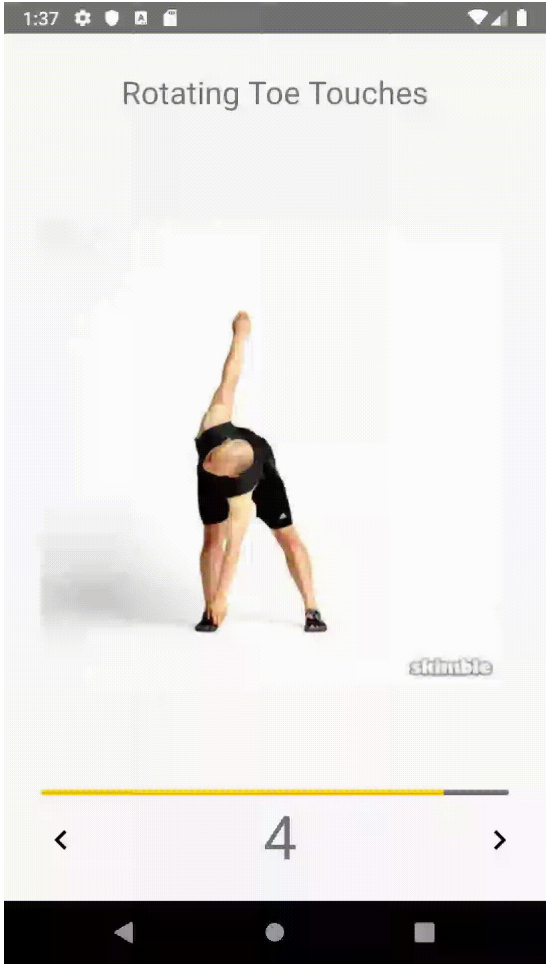
View Binding

DI & state preservation

```
// In Application configuration
val androidModule = module {
    single<Timer> { AndroidTimer() }
    single<Speaker> { AndroidSpeaker(get()) }
    viewModel { WorkoutViewModel(get(), get()) }
}
startKoin(this, listOf(androidModule))

// In Activity
private val vm by viewModel<WorkoutViewModel>()
```

Koin

https://github.com/MarcinMoskala/WorkoutMPP


1:37
Rotating Toe Touches
skimble
4

DI & state preservation

Conclusions

Starting Kotlin MPP is easy

We can extract common logic into presenters

Kotlin/Native interoperability is good, but still having problems with generics

We can have MVVM

View Binding is also simple

Your first multiplatform Kotlin project

Slides will be published on Twitter @marcinmoskala