Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

Android Programming with Kotlin

Eamonn de Leastar
edeleastar@wit.ie

Personal Background

Course Mission

Course Structure

Schedule & Calendar

Assessment, Course Web + Slack Channel

Personal Background

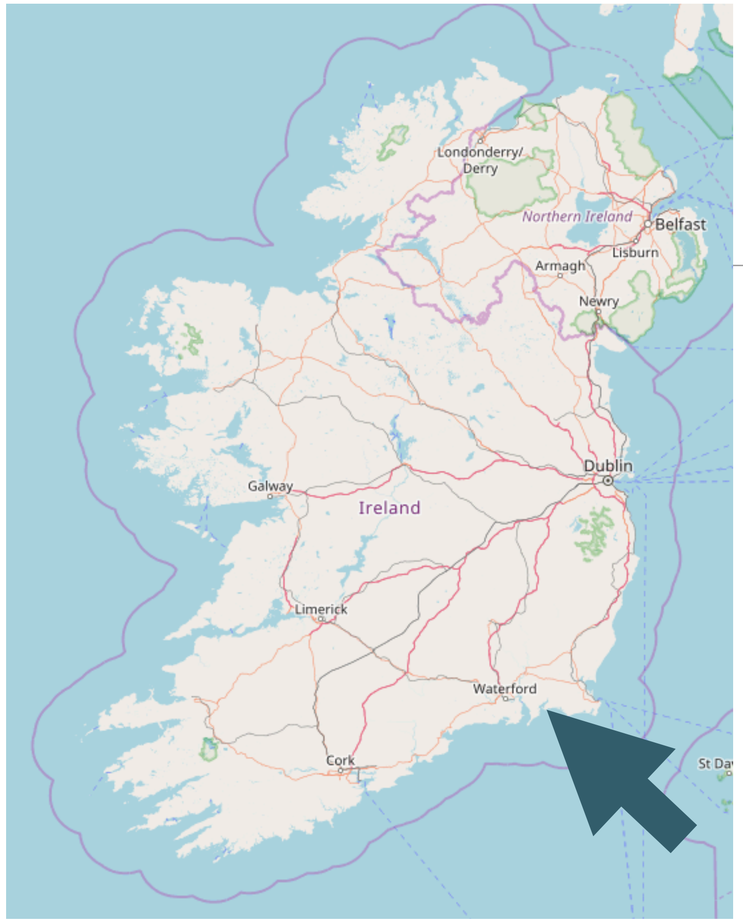
Course Mission

Course Structure

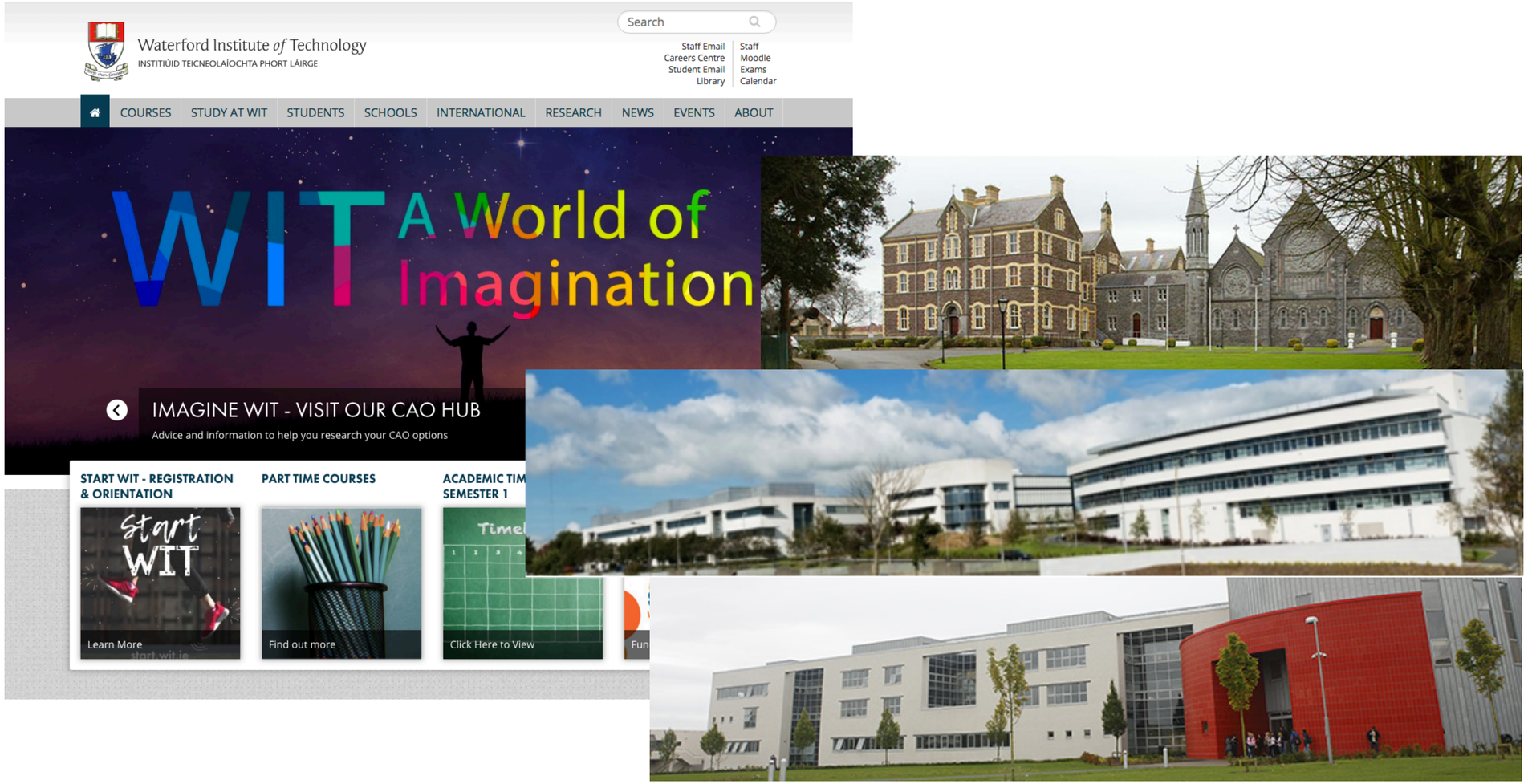
Schedule & Calendar

Assessment, Course Web + Slack Channel

Waterford



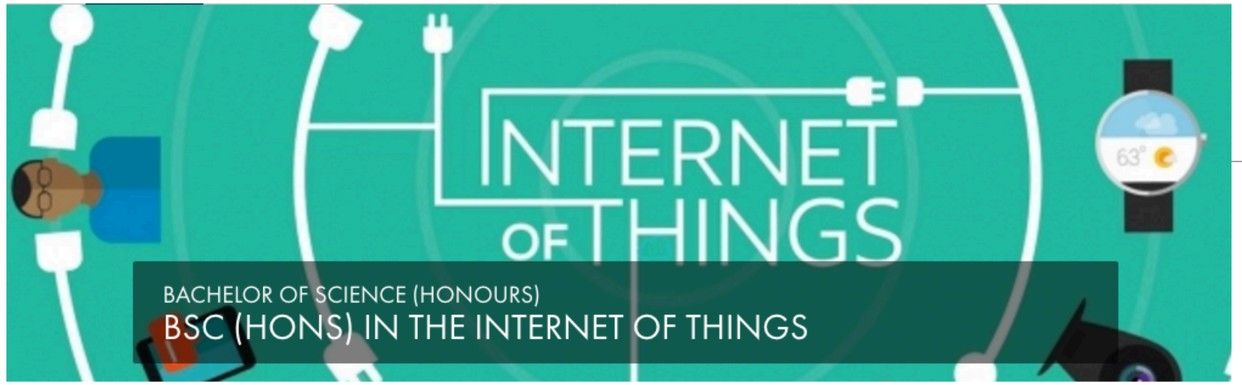
Waterford (from Old Norse *Veðrafjǫrðr*, meaning "ram (wether) fjord", Irish: *Port Láirge*) is a city in Ireland. It is in the South-East Region, Ireland and is part of the province of Munster. The city is situated at the head of Waterford Harbour. It is the oldest[2][3] and the fifth most populous city in the Republic of Ireland. It is the eighth most populous city on the island of Ireland.



Waterford Institute of Technology (WIT) is a university-level institution in the South-East of Ireland with over 10,000 students and 1,000 staff.

WIT offers tuition and research programmes in various areas from Higher Certificate to Degree to PhD.

https://www.wit.ie/courses/bsc-hons-in-applied-computing-internet-of-things



Some background …

Home » Courses » School » Science » Department of Computing, Maths & Physics

BY SCHOOL
- Business
- Lifelong Learning & Education
- Engineering
- Health Sciences
- Humanities
- Science and Computing

Code: WD197 | Delivery: Full Time | 2016 CAO range: 275 - 430
Level: 8 | Campus: Cork Road | Duration: 4 years
Credits: 240 ECTS

DESCRIPTION | OUTLINE | ENTRY | OPPORTUNITIES | PROJECTS | STORIES

Last Word Interview

Today FM Last Word Internet of Things Discussion
Matt Cooper, Eamonn de Leastar(WIT IoT Course Leader) & Barry Morris
0:00 / 11:44
1. Today FM Last Word Internet of Things Discussion by Matt Cooper, Eamonn de Leastar(WIT IoT Course Lead…
2. Limericks Live 95fm: The Internet of Things by Shane McAllister, Eamonn de Leastar & Anthony Quigney

Be part of the Internet of Things (IOT)

On this exciting new course developed in conjunction with the Institute's world class research gr
TSSG, ACG, and CTRG, you will learn how to programme the next wave of connected devices, you
explore the software and hardware that is transforming the world, connecting things to create ra
new services and products.

aurelia | Get the Newsletter

Home | Guides | APIs | Discourse | Blog | Support | Learn

Case Study: Tutors

Posted by Eamonn de Leastar on September 3, 2019

Today, I'm pleased to introduce you to Eamonn de Leastar from the Waterford Institute of Technology. He's built an impressive set of learning technologies for the institute using Aurelia. Here he is to tell you all about it and provide some opportunities to get involved as well.

Most developers have experience working though online tutorials, often complimented with videos and other resources. These types of online resources are central to how we build new skills and competencies in our discipline. The sequence in which the 'labs' or 'lessons' are presented, coupled with rich text formatting and code syntax highlighting, are essential enablers of the learning experience. Additionally, the wider context in which practical labs are embedded is also important, particularly if we are embarking on a complete course as opposed to short focussed lessons. For example, below is an hub site for a complete programme, consisting of multiple individual courses:

https://aurelia.io/blog/2019/09/03/case-study-tutors

http://www.tssg.org/about/people/eamonn-de-leastar/

TSSG

ABOUT» PROJECTS RESEAR

Home » About » People » Eamonn de Leastar

Eamonn de Leastar , Board Member

Eamonn de Leastar
Board Member
Ph: +353 (0)51 302 965
Email: edeleastar@tssg.org

Personal Summary

Eamonn de Leastar is a co-founder and Chief Technical Officer of the Software Systems Group (TSSG).

After graduation in 1982, he taught computer science at Dundalk Ins he was responsible for a range of new curricula within this sector. He then moved into industry, gaining significant experience in senior programmer, analyst, quality assurance and project leadership roles. This included periods at Wordstar International, developing some of the first work processors on the then newly established PC standard and at Contel Business Systems

Personal Background

Course Mission

Course Structure

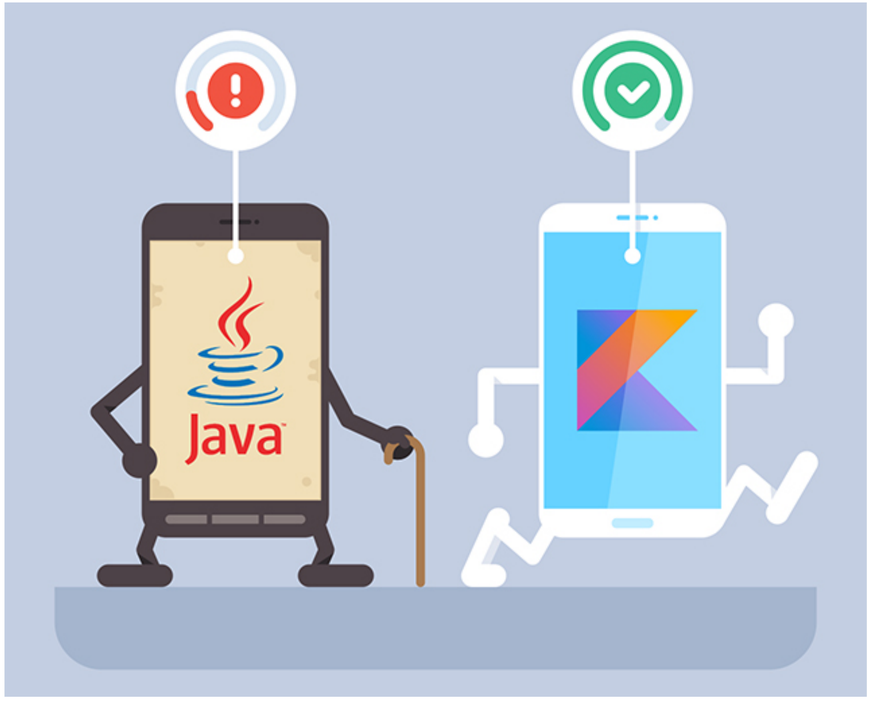
Schedule & Calendar

Assessment, Course Web + Slack Channel

2019

Android Programming with Kotlin

Course Mission: *Accelerate into Android App development leveraging the power & expressiveness of Kotlin*



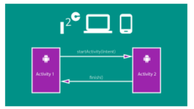
Android Programming in Kotlin

Eamonn de Leastar, WIT for OTH Regensburg

00: Introduction

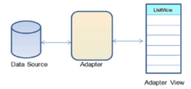
Android is a mobile operating system designed primarily for touchscreen devices

01: Activities



Activities are the fundamental building block of Android applications

02: Adapters



Application models are presented to activities using Adapters.

03: Models

Models capture the essential information realised by the application.

04: Images

Images can be selecting and displayed from the device image gallery.

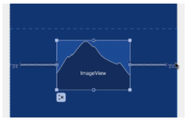
05: Maps

Google Maps can be integrated into and android app

06: Persistence

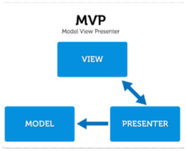
Models can be saved/recovered by the application.

07: Layouts



The UX is organised on screen by Layout components.

08: MVP



The Model View Presenter patterns organises the application structure

09: Location

An application can determine and track the device's current location.

10: Rooms

The Room framework is part Android Architecture Components

11: Firebase

Firebase provides Authentication and Storage cloud services

“Building Android applications is both exciting and challenging. The diversity of the applications one can build is fascinating, however their complexity can be overwhelming. Kotlin, a new language fully supported by google, dramatically enhances the power and simplicity of the programmers task.”

Kotlin is:

Modern

Secure

Typesafe

Expressive

Concise

Assumptions:
Intermediate level Java or closely related language skills

Accelerate into the fundamentals of Kotlin we need

Every topic explores specific Kotlin features *in parall*el to android code

Personal Background

Course Mission

 Course Structure 

Schedule & Calendar

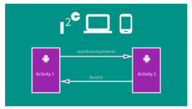
Assessment, Course Web + Slack Channel

12 Topics

00: Introduction

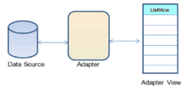
Android is a mobile operating system designed primarily for touchscreen devices

01: Activities



Activities are the fundamental building block of Android applications

02: Adapters



Application models are presented to activities using Adapters.

03: Models

Models capture the essential information realised by the application.

04: Images

Images can be selecting and displayed from the device image gallery.

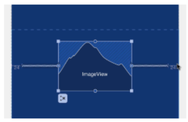
05: Maps

Google Maps can be integrated into and android app

06: Persistence

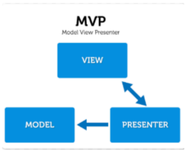
Models can be saved/recovered by the application.

07: Layouts



The UX is organised on screen by Layout components.

08: MVP



The Model View Presenter patterns organises the application structure

09: Location

An application can determine and track the device's current location.

10: Rooms

The Room framework is part Android Architecture Components

11: Firebase

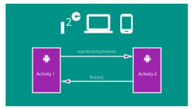
Firebase provides Authentication and Storage cloud services

Online now

00: Introduction

Android is a mobile operating system designed primarily for touchscreen devices

01: Activities



Activities are the fundamental building block of Android applications

02: Adapters

Application models are presented to activities using Adapters.

03: Models

Models capture the essential information realised by the application.

12 Topics

04: Images

Images can be selecting and displayed from the device image gallery.

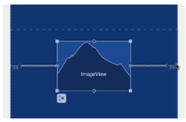
05: Maps

Google Maps can be integrated into and android app

06: Persistence

Models can be saved/recovered by the application.

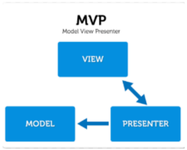
07: Layouts



The UX is organised on screen by Layout components.

Delivered onsite

08: MVP



The Model View Presenter patterns organises the application structure

09: Location

An application can determine and track the device's current location.

10: Rooms

The Room framework is part Android Architecture Components

11: Firebase

Firebase provides Authentication and Storage cloud services

Delivered Online post onsite

00: Introduction

Android is a mobile operating system designed primarily for touchscreen devices

Lab A00 Studio

Download and configure Android Studio. Generate and run a sample application.

Evolution of Kotlin

Programming language heritage, characteristics & type systems

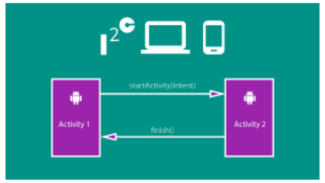
Kotlin in Context

A simple algorithm expressed in Java, Groovy, Swift and Kotlin

Android Overview 1

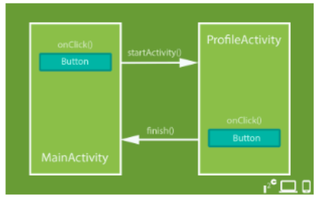
Here we take a brief look at Android, exploring its background and main features.

01: Activities



Activities are the fundamental building block of Android applications

Lab-A01 Activities

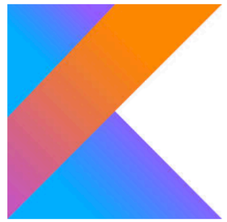


Layout the PlacemarkActivity - supporting create/edit of placemarks

Kotlin Structure

The fundamental characteristics of the language

Basic Syntax I



Rapid tour of the basic syntax of Kotlin

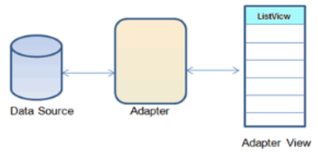
Android Overview 2

Part 2 focuses on setting up the Android environment and makes comparisons with competing platforms.

Android: Foundation

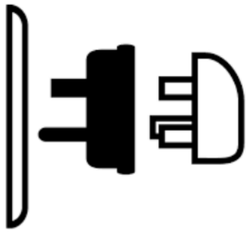
A first look at the structure of an Android application a Kotlin activity.

02: Adapters



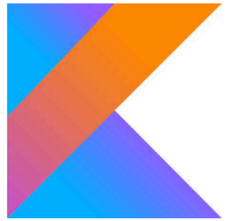
Application models are presented to activities using Adapters.

Lab-A02 Adapters



Introduce new activity to display a list of placemarks. Support adding to this list.

Basic Syntax II

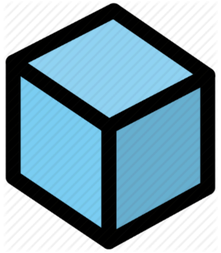


More of the basics of Kotlin

Kotlin Deep Dive

A deep dive into some key Kotlin features: Data Classes & Lambdas

Android Application Object

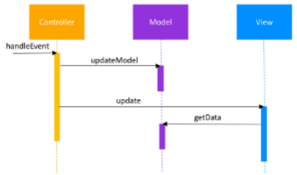


This class enables a global application object to be defined, accessible from all activities

Recycler View + Adapters

Managing lists is facilitated by the RecyclerView + Adapters - 2 key patterns in Android development

03: Models



Models capture the essential information realised by the application.

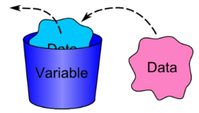
Lab-A03 Models

Application object + Model classes to manage placemarks.

Idioms

A tour of the common idioms in Kotlin

J2K Variables & Expressions



Snippets of the same code in Java & Kotlin

Toolbar

The Android Support Library implements AppCompatActivity + a range of general purpose components, including a ToolBar

Memory Store

Abstract the Placemarks data structure into PlacemarkStore interface + in-memory implementation.

Adapter + Listener

Equip the Adapter with a Listener interface. Use this interface to communicate from the adapter to the host Activity

Parcelable

Encapsulate model data for transmission between Activities

04: Images

Images can be selecting and displayed from the device image gallery.

Lab-A04 Images

Allow an image to be selected from the phones photos, and stored with the placemark

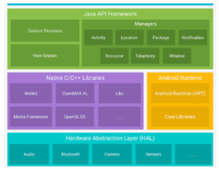
Types

Summary of the basic types in the Kotlin programming language

J2K Collections

Collection examples: Java & Kotlin equivalents

Android Platform, Components & Activities

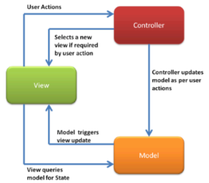


Key features of t platform, the princ components + the a model

Resources

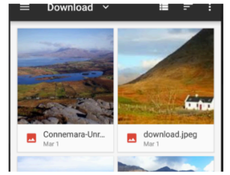
A general term for all layout, strings, bitmaps and other XML artefacts in an Android

Model Updates



Creating and updating a Placemark. Updating the placemark list.

Images



Support selecting image from phone gallery, and then displaying them in an activity.

05: Maps

Google Maps can be integrated into and android app

Lab-A05 MapActivity

Include a MapActivity, enabling the user to select the location of the placemark

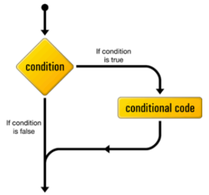
Kotlin Packages

Source files start with a package declaration. This can be used to access (import) features from the source file.

Kotlin Control Flow

if, when, for and while statements

J2K Control Flow

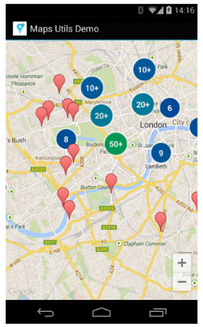


Control flow examples in Java and Kotlin

App Structure

Key classes and relationships in the application.

Map Activity



Google Map Activity can be inserted into an app via a Wizard from Studio. API Keys must be acquired from google directly.

Cameras & Markers

These abstractions enable the map view to be manipulated programatically, and facilitate direct manipulation by the user.

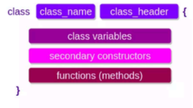
06: Persistence

Models can be saved/recovered by the application.

Lab-A06 JSON

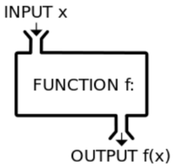
Save and restore placemarks from a JSON formatted file

Classes & Inheritance



In kotlin, classes are more concise, explicit and fine-grained then Java

J2K: Functions



Functions in Java & Kotlin

Android Anatomy 2

Here we continue our look at the Android eco-system and the components that make up this system.

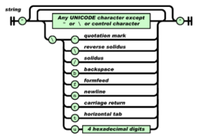
File Formats

```
<?xml version="1.0"
encoding="UTF-8"?>

<coffee objname="c1">
    <name> mocha </name>
    <shop> costa </shop>
    <price> 2.0 </price>
    <rating> 3.5</rating>
    <favourite> 0 </favourite>
</coffee>
```

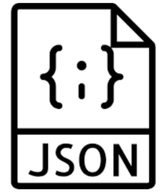
When a simple file persistence strategy, CSV, Name/Value, YAML, XML & JSON are all potential candidate formats for file-

JSON



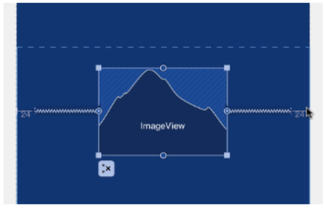
A detailed look at the syntax of JSON

JSON Store



A new PlacemarkStore implementation - PlacemarkJSONStore - to persist placemarks to a JSON file.

07: Layouts

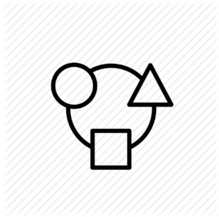


The UX is organised on screen by Layout components.

Lab-A07 MapView

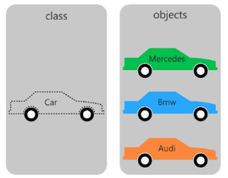
Display all placemarks on a map in a new activity

Properties & Fields



Kotlin properties and fields offer a richer set of features and variants ov

J2K : Classes



Classes in Java & Kotlin

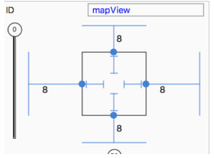
Android Anatomy 3

Here we continue our discussion on the components th the Andrc

Android Anatomy 4

Here we take a look at the activity and fragment life-cycles within Android.

Constraint Layout

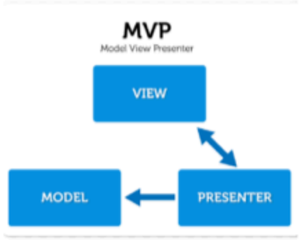


ConstraintLayout allows you to create large and complex layouts with a flat view

Overall Map Activity

Introduce an activity to display all placemarks on a single map.

08: MVP



The Model View Presenter patterns organises the application structure

09: Location

An application can determine and track the device's current location.

10: Rooms

The Room framework is part Android Architecture Components

11: Firebase

Firebase provides Authentication and Storage cloud services

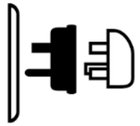
Lab A00 Studio

Download and configure Android Studio. Generate and run a sample application.

Lab-A01 Activities

Layout the PlacemarkActivity - supporting create/edit of placemarks

Lab-A02 Adapters



Introduce new activity to display a list of placemarks. Support adding to this list.

Lab-A03 Models

Application object + Model classes to manage placemarks.

17 Labs

Lab-A04 Images

Allow an image to be selected from the phones photos, and stored with the placemark

Lab-A05 MapActivity

Lab-A06 JSON

Lab-A07 MapView

Display all placemarks on a map in a new activity

Lab-A08a MVP I

Refactor Activities to use the Model View Presenter pattern

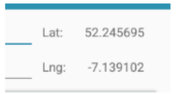
Lab-A08b MVP II

Complete the presenter pattern imple introducing B BasePresent

Lab-A09a Location

When creating a new placemark, use the current location as the starting

Lab-A09b Tracking



Extend the location facility to track location in real time.

Lab-A10a AndroidX

Migrate to AndroidX + manage library versions more optimally

Lab-A10b Rooms

Introduce a new PlacemarkStore implementation to persist to an SQLite database

Lab-A11a Firebase Auth

Authenticate users against the Firebase Authentication service

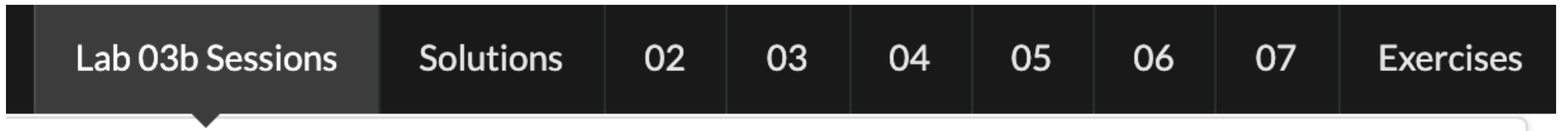
Lab-A11b Firebase Database

Store placemarks in Firebase Realtime Database

Lab-A11c Firebase Storage

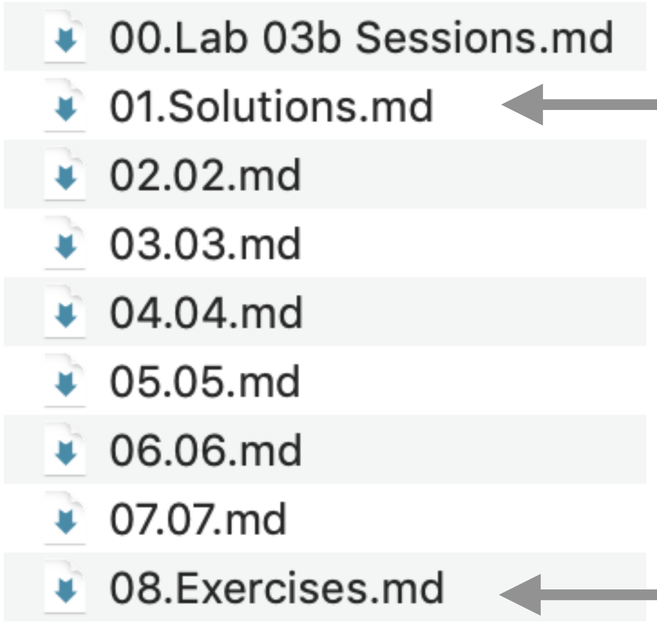
Store images in Firebase Storage

Labs Step Structure



"Constructivist" Lab steps.

Incrementally enhance an artefact/feature/ design



Solutions to Previous Lab

Lab Bookends

Reinforcement Exercises for this Lab

Conclude with exercises to bring artefact/feature/design forward to next Lab

Mix of:

Headings
Code Fragments
Images
Text

Labs Step Content

LAB-A02 ADAPTERS | EXERCISE SOLUTIONS | 02 | 03 | 04 | 05 | 06 | EXERCISES

MainApp

Create a new package called `org.wit.placemark.main`, and introduce this class:

MainApp

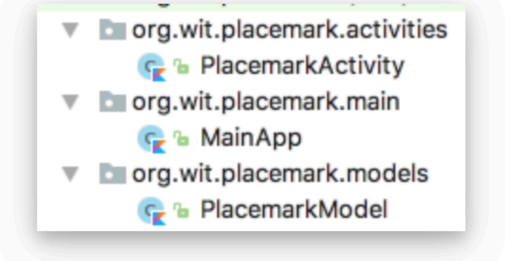
```
package org.wit.placemark.main

import android.app.Application
import org.jetbrains.anko.AnkoLogger
import org.jetbrains.anko.info

class MainApp : Application(), AnkoLogger {

  override fun onCreate() {
    super.onCreate()
    info("Placemark started")
  }
}
```

The package structure should look like this:



In addition, change the AndroidManifest to specifically reference this class:

Labs Exercises

Complete archive of lab artefact before Exercises

Maximum of 2-3 Exercises
Solutions provided as first step of next lab

Solution

Placemark application so far:

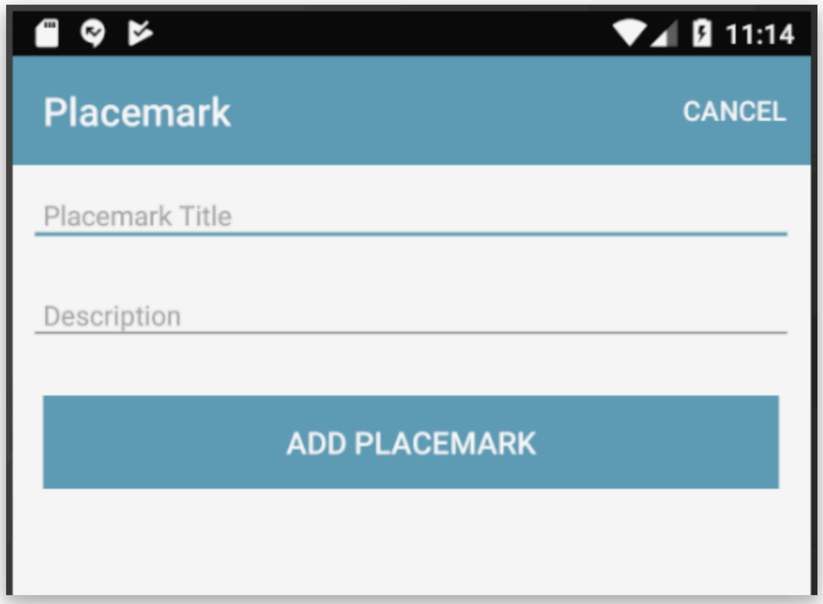
- placemark-02.zip

Exercise 1: Sample Solution

Make sure you can download and run the sample solution (archive above)

Exercise 2: Cancel Button

Incorporate new 'Cancel' action into `PlacemarkActivity`. This should return to PlacemarkListActivity without adding a new Placemark.

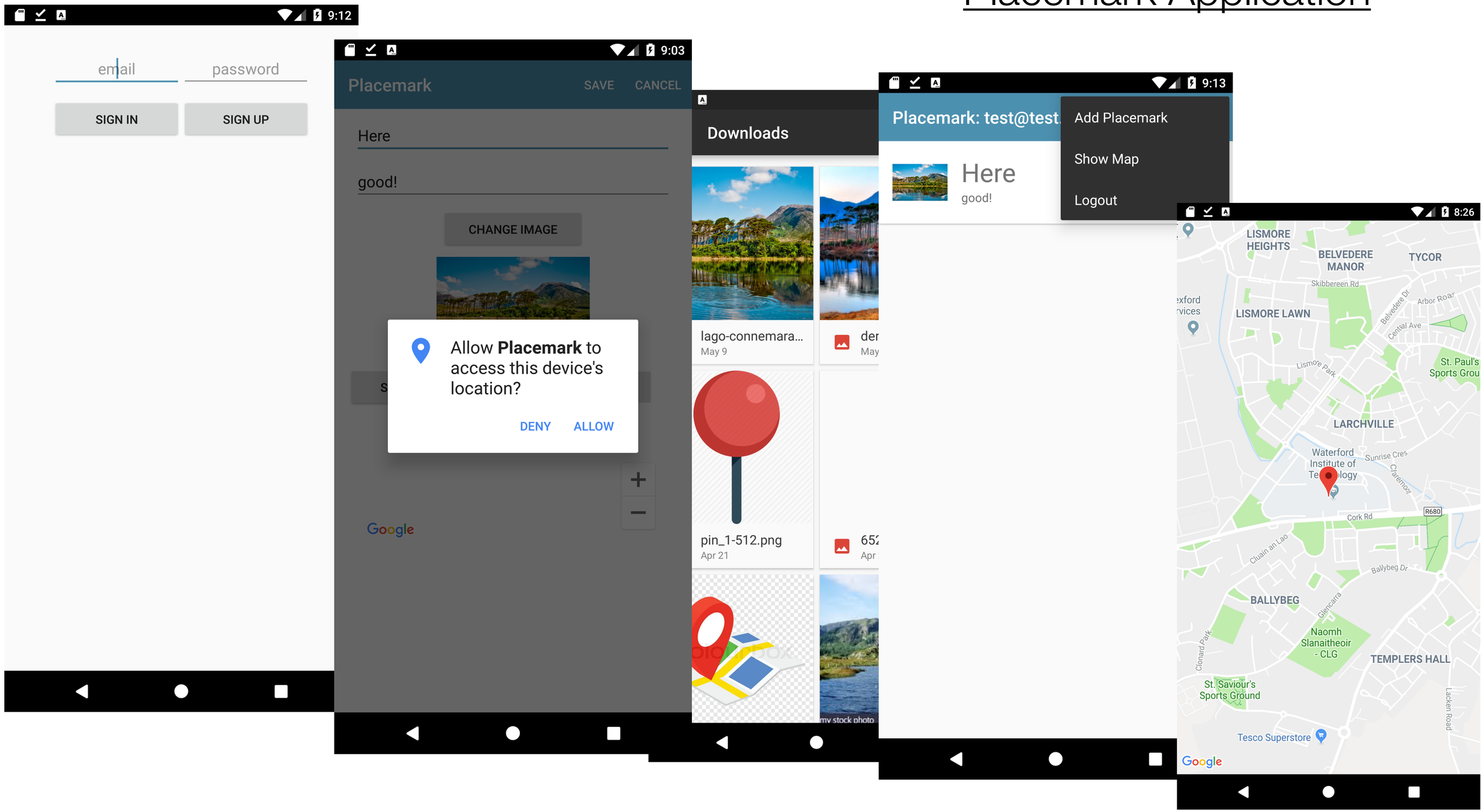


Follow the steps 05 & 06 in this lab as a guide to doing this. Remember, you will be introducing the menu/action into PlacemarkActivity.

Exercise 3: Refactor PlacemarkAdapter to its own source file.

The PlacemarkAdapter class is currently in the same source file as the PlacemarkListActivity class. For clarity and ease of maintenance, move this into its own source file.

Placemark Application



Personal Background

Course Mission

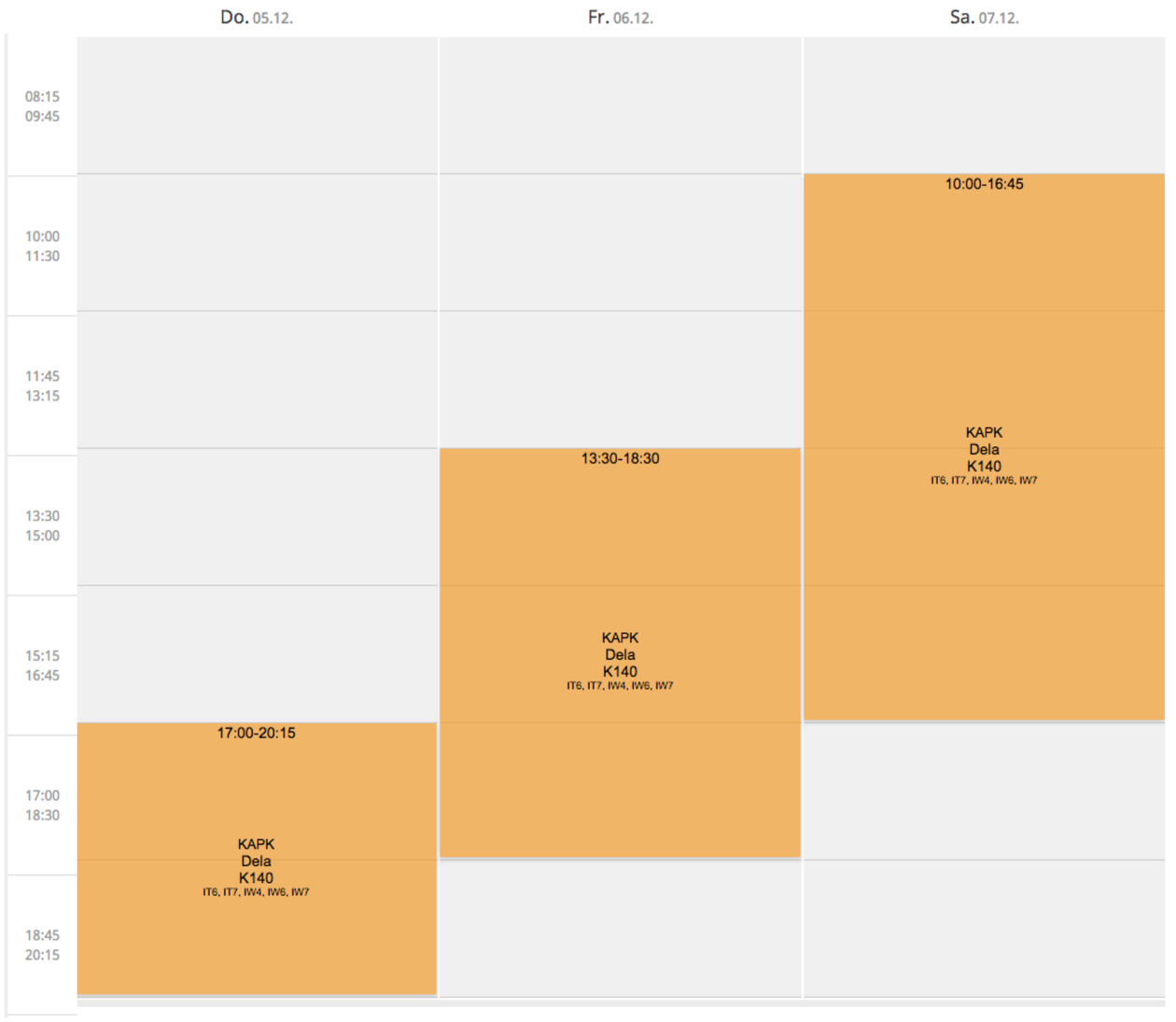
Course Structure

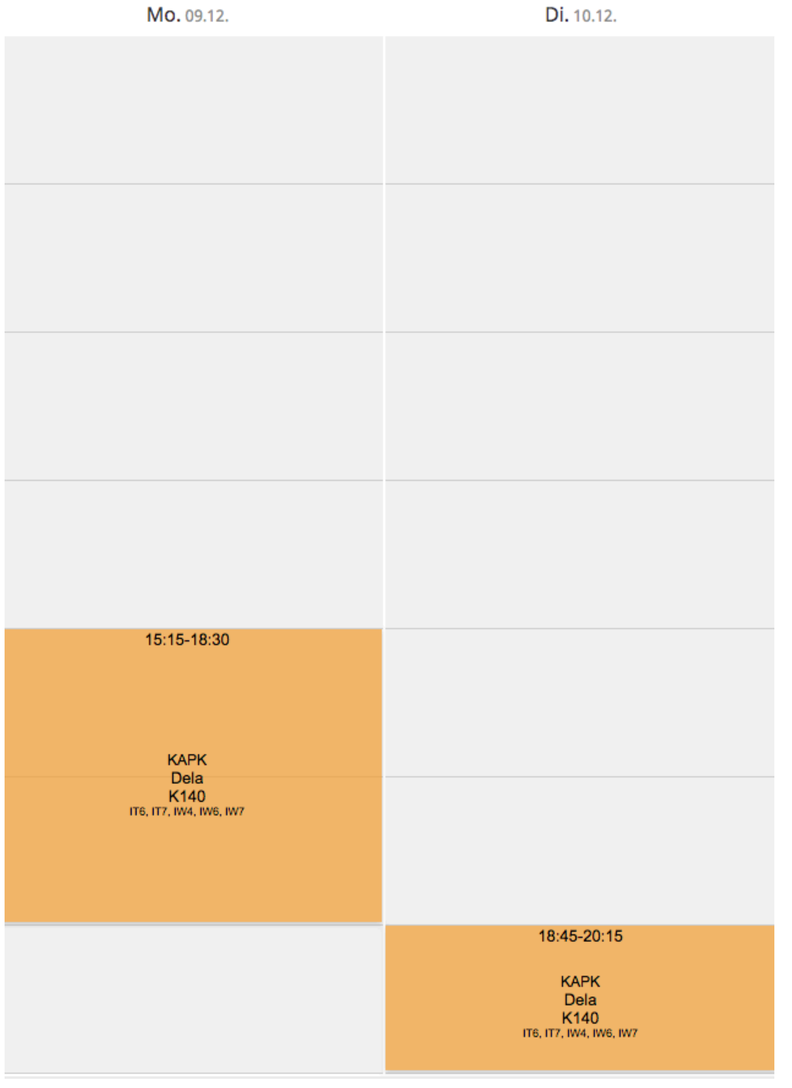
 Schedule & Calendar 

Assessment, Course Web + Slack Channel

Schedule (revised)

October	29	30	1	2	3	4	5	
	6	7	8	9	10	11	12	
	13	14	15	16	17	18	19	
	19	21	22	23	24	25	26	virtual
November	27	28	29	30	31	1	2	virtual
	3	4	5	6	7	8	9	virtual
	10	11	12	13	14	15	16	virtual
	17	18	19	20	21	22	23	virtual
December	24	25	26	27 on-site	28	29	30	
	1	2	3	4	5	6	7	on-site
	8	9	10	11	12	13	14	on-site
	15	16	17	18	19	20	21	
	22	23	24	25	26	27	28	
January	29	30	31	1	2	3	4	
	5	6	7	8	9	10	11	
	12	13	14	15	16	17	18	
	19	20	21	22	23	24	25	assignment submission


Do. 05.12.
Fr. 06.12.
Sa. 07.12.
08:15
09:45
10:00
11:30
11:45
13:15
13:30
15:00
15:15
16:45
17:00
18:30
18:45
20:15
17:00-20:15
KAPK
Dela
K140
IT6, IT7, IW4, IW6, IW7
13:30-18:30
KAPK
Dela
K140
IT6, IT7, IW4, IW6, IW7
10:00-16:45
KAPK
Dela
K140
IT6, IT7, IW4, IW6, IW7


Mo. 09.12.
Di. 10.12.
15:15-18:30
KAPK
Dela
K140
IT6, IT7, IW4, IW6, IW7
18:45-20:15
KAPK
Dela
K140
IT6, IT7, IW4, IW6, IW7

Personal Background

Course Mission

Course Structure

Schedule & Calendar

Assessment, Course Web + Slack Channel

Single Assignment

	S	M	T	W	T	F	S	Modules
September	1	2	3	4	5	6	7	
	8	9	10	11	12	13	14	
	15	16	17	18	19	20	21	
	22	23	24	25	26	27	28	
October	29	30	1	2	3	4	5	
	6	7	8	9	10	11	12	
	13	14	15	16	17	18	19	
	19	21	22	23	24	25	26	virtual
November	27	28	29	30	31	1	2	virtual
	3	4	5	6	7	8	9	virtual
	10	11	12	13	14	15	16	virtual
	17	18	19	20	21	22	23	virtual
December	24	25	26	27	28	29	30	
	1	2	3	4	5	6	7	on-site
	8	9	10	11	12	13	14	on-site
	15	16	17	18	19	20	21	
	22	23	24	25	26	27	28	
January	29	30	31	1	2	3	4	
	5	6	7	8	9	10	11	
	12	13	14	15	16	17	18	
	19	20	21	22	23	24	25	assignment submission

Due January 25, 2020

Assignment

Android Programming in Kotlin

Tutors 0.4.1 (1.0.8)

Assignment Specification

Archaeological Fieldwork

Specification, grading guidelines and schedule for the Assignment

Project Submission Template

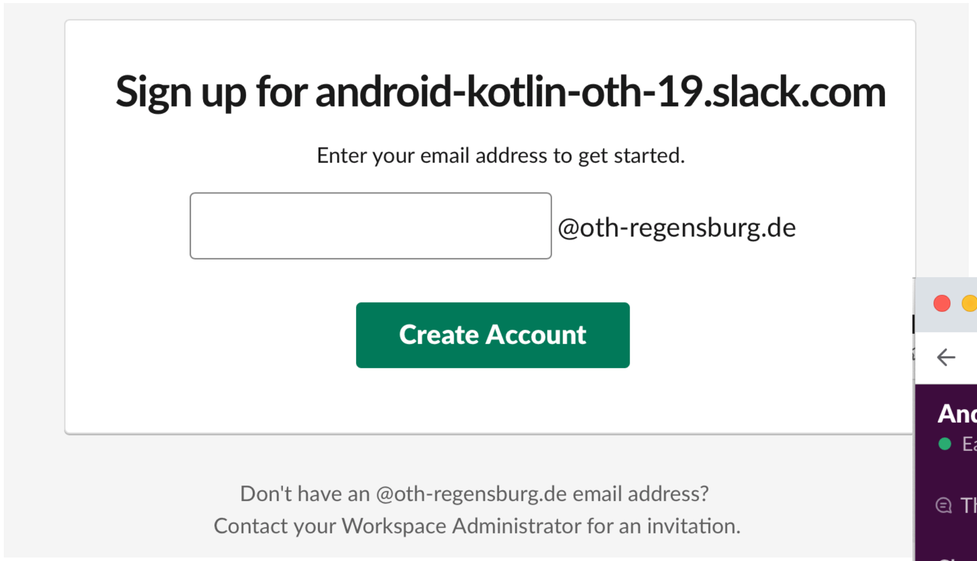
An archive of a word document to be submitted with the assignment

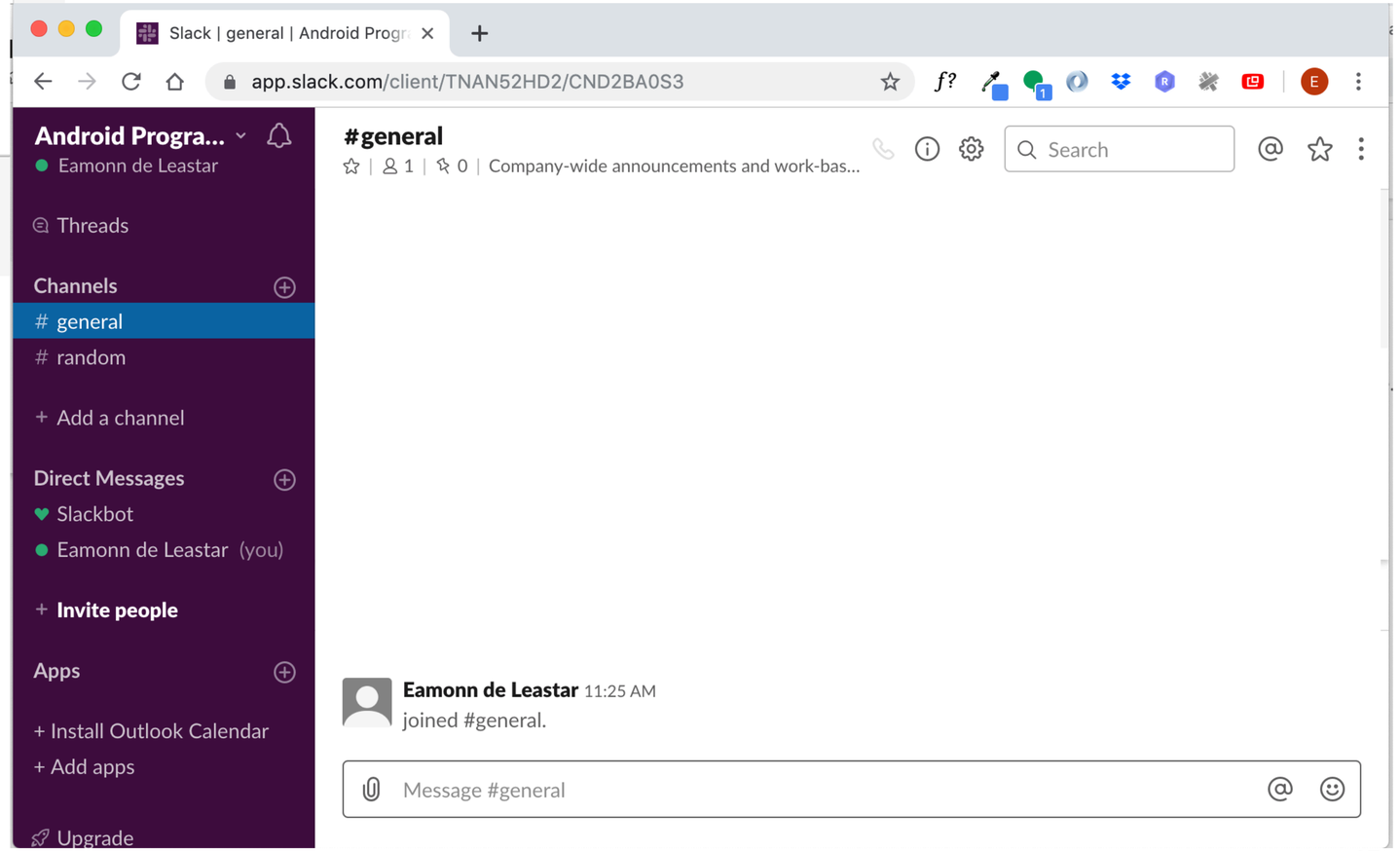
Assignment Submission

Moodle assignment dropbox for Assignment

https://android-kotlin-oth-19.slack.com/signup

Course communications, Lab support + announcements via Slack





https://oth-mobile-app-dev-2019.netlify.com

Android Programming in Kotlin

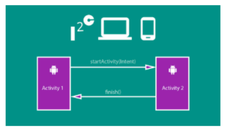
Tutors 0.4.1 (1.0.8)

Eamonn de Leastar, WIT for OTH Regensburg

00: Introduction

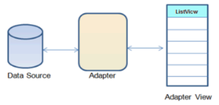
Android is a mobile operating system designed primarily for touchscreen devices

01: Activities



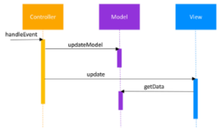
Activities are the fundamental building block of Android applications

02: Adapters



Application models are presented to activities using Adapters.

03: Models



Models capture the essential information realised by the application.

04: Images

Images can be selecting and displayed from the device image gallery.

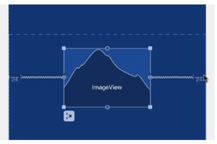
05: Maps

Google Maps can be integrated into and android app

06: Persistence

Models can be saved/recovered by the application.

07: Layouts



The UX is organised on screen by Layout components.