CS 528 Mobile and Ubiquitous Computing
Lecture 5a: More ML Kit & Databases

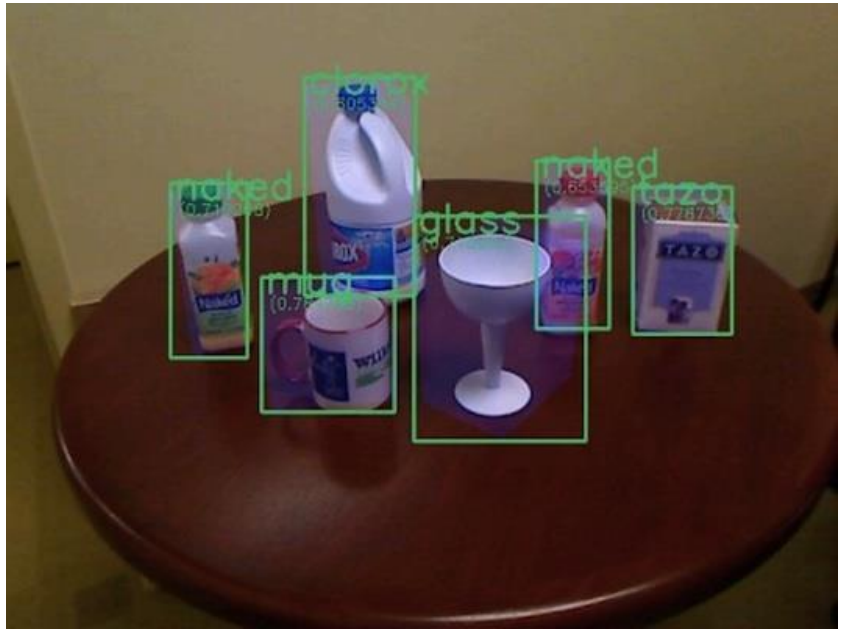
Emmanuel Agu

ML Kit: What else?

ML Kit Object Detection

Ref: https://developers.google.com/ml-kit/vision/face-detection

- **Face detection:** Special case of object-class detection
- **Object-class detection task:** find locations and bounding box of all objects in an image that belong to a given class.
  - E.g: all bottles, cups, pedestrians or cars



ML Kit Object Detection

Ref: https://developers.google.com/ml-kit/vision/object-detection

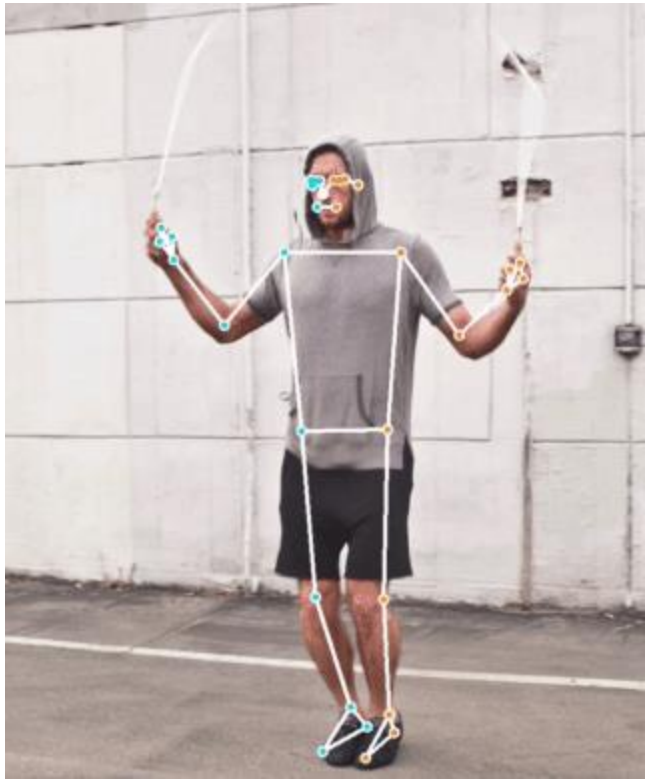
- **Object Detection and tracking:** Detect objects and their locations
- **Optimized on-device model:** optimized for real-time even on low end mobile devices
- **Prominent object detection:** determine most prominent object in image
- **Coarse category classification:** Classify objects into broad classes (home goods, fashion goods, food, plants and places)
- **Classification using custom model:** user can provide image classification model

Object 0	
Bounds	(1, 97), (332, 97), (332, 332), (1, 332)
Category	FASHION_GOOD
Classification confidence	0.95703125
Object 1	
Bounds	(186, 80), (337, 80), (337, 226), (186, 226)
Category	FASHION_GOOD
Classification confidence	0.84375

ML Kit Pose Detection

Ref: https://developers.google.com/ml-kit/vision/pose-detection

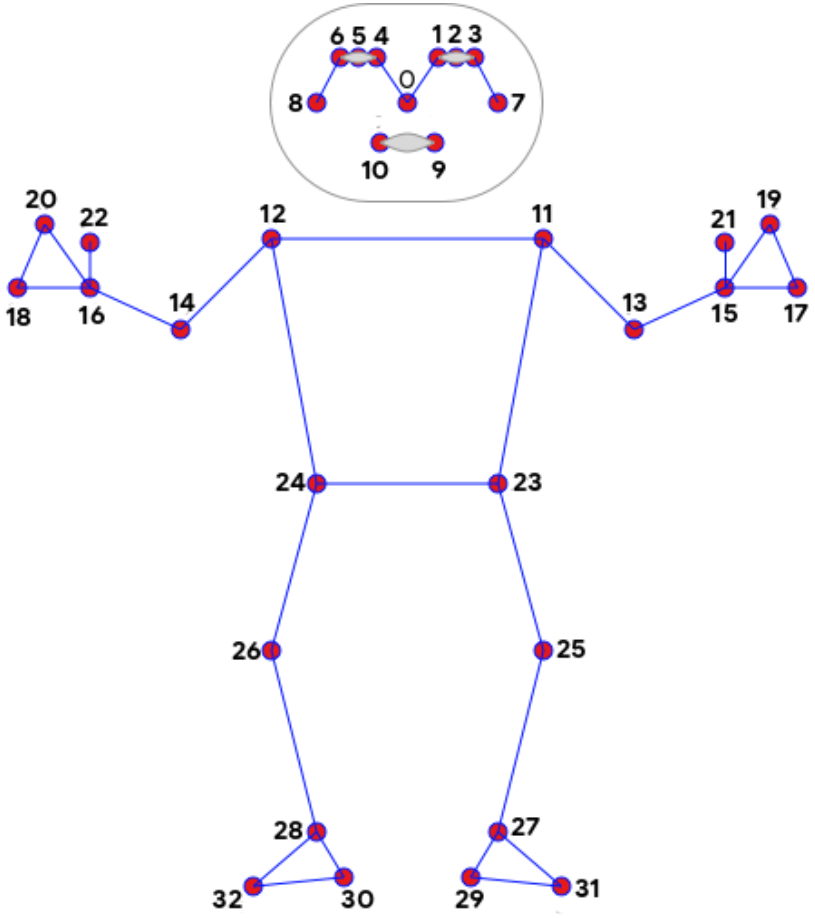
- Detects pose of subject's body in real-time from video or image
- Pose: body's position at a given moment
- Set of skeletal landmark points (e.g. shoulders, hips)
- ML kit pose detection produces 33 skeletal pose points, which includes facial landmarks (ears, eyes, mouth and nose)



ML Kit Pose Detection

Ref: https://developers.google.com/ml-kit/vision/pose-detection

- Landmarks generated



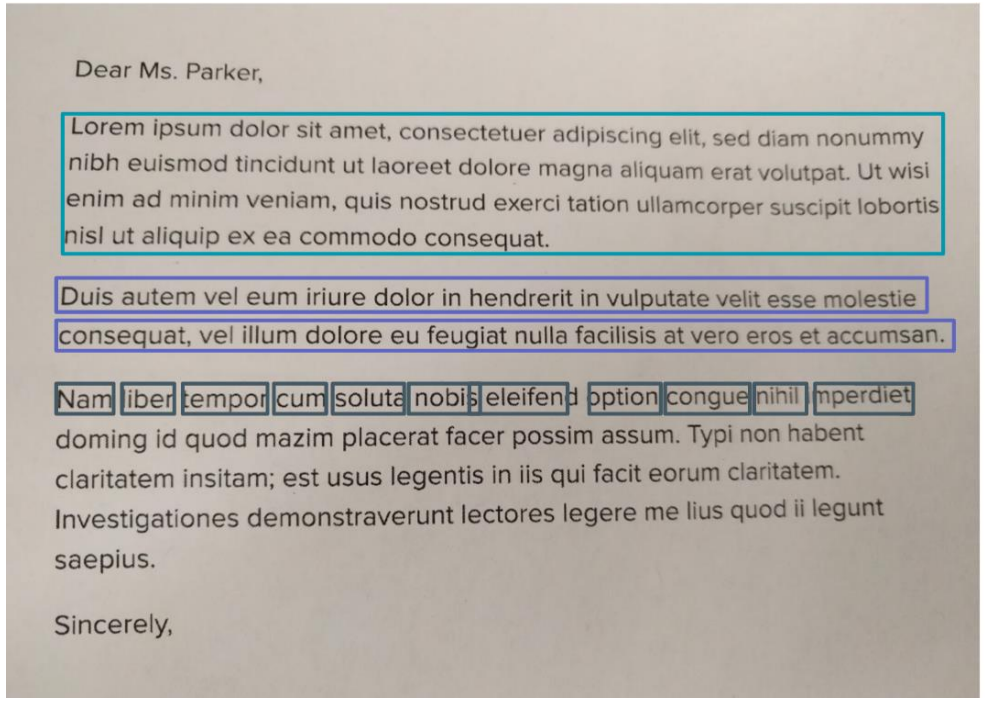
Landmarks

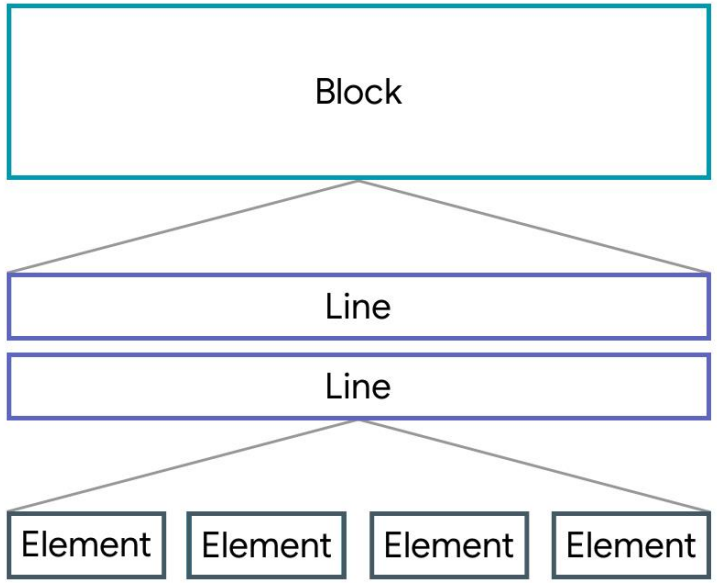
0. Nose
1. Left_eye_inner
2. Left_eye
3. Left_eye_outer
4. Right_eye_inner
5. Right_eye
6. Right_eye_outer
7. Left_ear
8. Right_ear
9. Left_mouth
10. Right_mouth
11. Left_shoulder
12. Right_shoulder
13. Left_elbow
14. Right_elbow
15. Left_wrist
16. Right_wrist
17. Left_pinky
18. Right_pinky
19. Left_index
20. Right_index
21. Left_thumb
22. Right_thumb
23. Left_hip
24. Right_hip
25. Left_knee
26. Right_knee
27. Left_ankle
28. Right_ankle
29. Left_heel
30. Right_heel
31. Left_foot_index
32. Right_foot_index

ML Kit Text Recognition

Ref: https://developers.google.com/ml-kit/vision/text-recognition/v2

- **Recognizes text in various languages:**
  - Chinese, Devanagari, Japanese, Korean and Latin scripts
- Analyzes text structure, detects symbols, elements, lines and paragraphs
- Identifies what language text is written in
- Real-time recognition on a wide range of devices

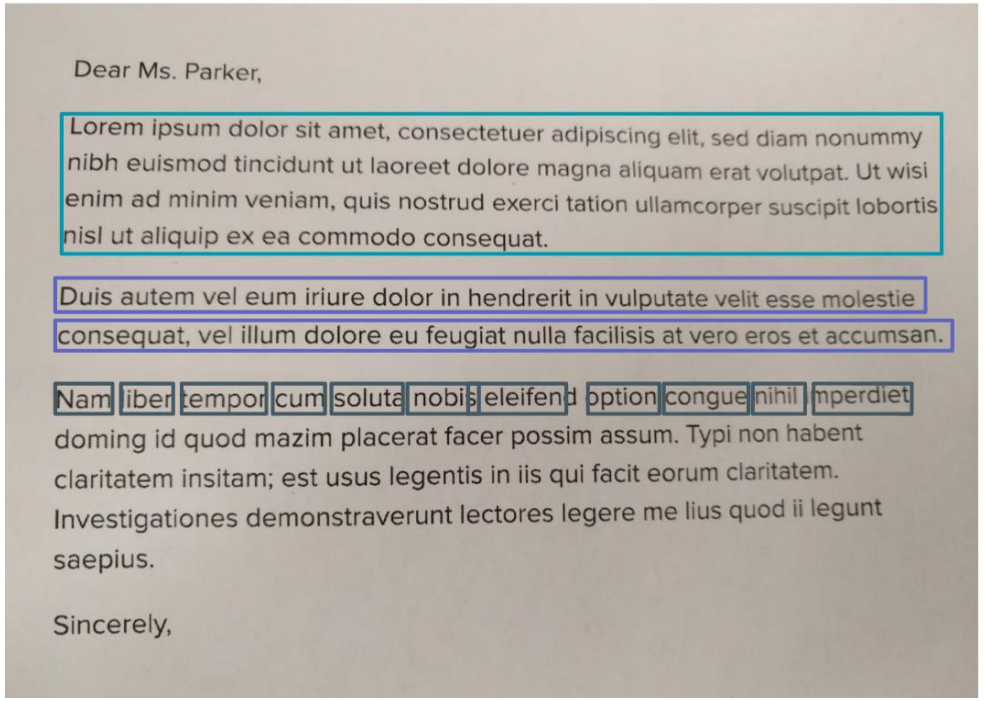


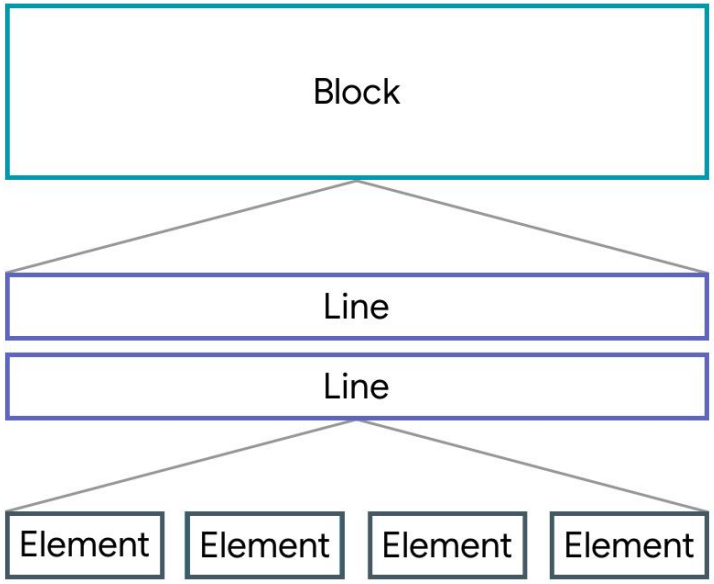


ML Kit Text Recognition

Ref: https://developers.google.com/ml-kit/vision/text-recognition/v2

- **Text structure:** segments (divides) text into blocks, lines, elements and symbols
  - **Block:** contiguous set of lines (e.g. paragraph)
  - **Line:** contiguous set of words on same axis
  - **Element:** Contiguous set of alphanumeric characters (“word”) on same axis in most latin languages
  - **Symbol:** single alphanumeric character in most latin languages





ML Kit Text Recognition

Ref: https://developers.google.com/ml-kit/vision/text-recognition/v2

- **Text Recognition Example**

Recognized Text	
Text	Wege der parlamentarischen Demokratie
Blocks	(1 block)

Block 0	
Text	Wege der parlamentarischen Demokratie
Frame	(296, 665 - 796, 882)
Corner Points	(296, 719), (778, 665), (796, 828), (314, 882)
Recognized Language Code	de
Lines	(3 lines)

Line 0	
Text	Wege der
Frame	(434, 678 - 670, 749)
Corner Points	(434, 705), (665, 678), (670, 722), (439, 749)
Recognized Language Code	de
Confidence Score	0.8766741
Rotation Degree	-6.6116457
Elements	(2 elements)

Element 0	
Text	Wege
Frame	(434, 689 - 575, 749)
Corner Points	(434, 705), (570, 689), (575, 733), (439, 749)
Recognized Language Code	de
Confidence Score	0.8964844
Rotation Degree	-6.6116457
Elements	(4 elements)

Symbol 0	
Text	W
Frame	(434, 698 - 500, 749)
Corner Points	(434, 706), (495, 698), (500, 741), (439, 749)
Confidence Score	0.87109375
Rotation Degree	-6.611646

ML Kit BarCode Scanning

Ref: https://developers.google.com/ml-kit/vision/barcode-scanning

- **Reads most standard barcode formats:**
  - Linear formats: Codabar, Code 39, Code 93, Code 128, EAN-8, EAN-13, ITF, UPC-A, UPC-E
  - 2D formats: Aztec, Data Matrix, PDF417, QR Code
- **Specify format**
  - Scan for all supported formats (slow)
  - Restrict scan to a specific format (faster)
- **Extracts structured data:**
  - URLs, email addresses, phone numbers, ISBNs, etc
- **Works with any orientation:**
  - Right-side-up, upside-down, or sideways

Result	
Corners	(49,125), (172,125), (172,160), (49,160)
Raw value	2404105001722

Result		
Corners	(87,87) (612,87) (612,612) (87,612)	
Raw value	WIFI:S:SB1Guest;P:12345;T:WEP;;	
WiFi information	SSID	SB1Guest
	Password	12345
	Type	WEP

ML Kit Image Labeling

Ref: https://developers.google.com/ml-kit/vision/image-labeling

- Identifies people, things, places, activities in a picture
- Returns confidence of each type returned

Category	Example labels

People	`Crowd`
	`Selfie`
	`Smile`

Activities	`Dancing`
	`Eating`
	`Surfing`

Things	`Car`
	`Piano`
	`Receipt`

Animals	`Bird`
	`Cat`
	`Dog`

Plants	`Flower`
	`Fruit`
	`Vegetable`

Places	`Beach`
	`Lake`
	`Mountain`

ML Kit Image Labeling

Ref: https://developers.google.com/ml-kit/vision/image-labeling

- Sample output

Label 0	
Text	Stadium
Confidence	0.9205354
Label 1	
Text	Sports
Confidence	0.7531109
Label 2	
Text	Event
Confidence	0.66905296
Label 3	
Text	Leisure
Confidence	0.59904146

ML Kit Image Labeling

Ref: https://developers.google.com/ml-kit/vision/image-labeling

- Sample output

Label 4	
Text	Soccer
Confidence	0.56384534
Label 5	
Text	Net
Confidence	0.54679185
Label 6	
Text	Plant
Confidence	0.524364

Digital Ink Recognition

Ref: https://developers.google.com/ml-kit/vision/digital-ink-recognition

- Recognize handwritten text, classify gestures and sketches on digital surface in over 300 languages
- Allows user to write instead of typing
- Recognizes hand-drawn shapes, emojis

Digital Ink Recognition

Ref: https://developers.google.com/ml-kit/vision/digital-ink-recognition

User drew this

Ink object generated

- 4 strokes in ink object. First 2 strokes are:

Ink		
Stroke 1	x	392, 391, 389, 287, ...
	y	52, 60, 76, 97, ...
	t	0, 37, 56, 75, ...
Stroke 2	x	497, 494, 493, 490, ...
	y	167, 165, 165, 165, ...
	t	694, 742, 751, 770, ...

Digital Ink Recognition

Ref: https://developers.google.com/ml-kit/vision/digital-ink-recognition

User drew this

Ink object generated

- Sending **Ink** object to recognizer returns possible transcriptions, in order of decreasing confidence

RecognitionResult	
RecognitionCandidate #1	handw
RecognitionCandidate #2	handrw
RecognitionCandidate #3	hardw
RecognitionCandidate #4	handu
RecognitionCandidate #5	handwe

Digital Ink Recognition

Ref: https://developers.google.com/ml-kit/vision/digital-ink-recognition

- Gesture classifiers classify ink stroke into one of nine gesture classes

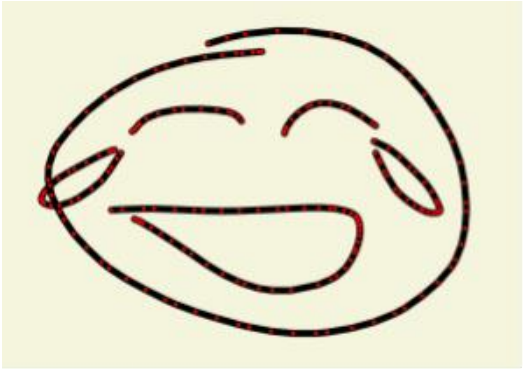
Gesture	Example
`arch:above` `arch:below`	An `arch`, or a curve touching two words, can be used to conect two wo rds toge ther. Could be above or below the text line.
`caret:above` `caret:below`	A `caret` is commonly used to place the text editing cursor. Similar to `arch`, could be above or below the line.
`circle`	Drawing a `circle`-like shape around the text is usually used to do the text selection.
corner:downleft	A `corner` looks like an inverted L shape is used to break the line. It looks like a non-inverted L in the right-to-left languages.
`scribble`	A `scribble` gesture is used to delete a text. It could be a vertical or a horizontal gesture.
`strike`	A `strike` could be either a deletion or a selection gesture. As with any other gesture, it is up to developer to decide how to interpret it for their use case.
`verticalbar`	A `verticalbar` gesture could be used to break a word in two.
`writing`	Any strokes that were not classified as gestures will be classified as `writing`, and the respective text and/or shape recognizer could be used to recognize the writing.

Digital Ink Recognition

Ref: https://developers.google.com/ml-kit/vision/digital-ink-recognition

- Emoji sketch example

User drew this



Ink object

- **Ink** object contains six strokes

Ink		
Stroke 1	x	269, 266, 262, 255, ...
	y	40, 40, 40, 41, ...
	t	0, 36, 56, 75, ...
Stroke 2	x	179, 182, 183, 185, ...
	y	157, 158, 159, 160, ...
	t	2475, 2522, 2531, 2541, ...

Results of emoji recognizer

RecognitionResult	
RecognitionCandidate #1	😂 (U+1f62d)
RecognitionCandidate #2	😅 (U+1f605)
RecognitionCandidate #3	😹 (U+1f639)
RecognitionCandidate #4	😄 (U+1f604)
RecognitionCandidate #5	😆 (U+1f606)

Text Language Identification

Ref: https://developers.google.com/ml-kit/language/identification

- Determines language of a string of text with confidence score
- Identifies Arabic, Bulgarian, Greek, Hindi, Japanese, Russian, and Chinese text in both native and romanized script
- **Examples:**

Example results

Simple language identification	
"My hovercraft is full of eels."	en (English)
"Dao shan xue hai"	zh-Latn (Latinized Chinese)
"ph'nglui mglw'nafh wgah'nagl fhtagn"	und (undetermined)

Confidence distribution	
"an amicable coup d'etat"	en (0.52) fr (0.44) ca (0.03)

Translation

Ref: https://developers.google.com/ml-kit/language/translation

- Translate between over 50 languages including
  - Africaans, Arabic, Belarusian, Bulgarian, Bengali, Catalan, Czech, Welsh, Danish, German, Greek, English, Spanish, etc
  - See: https://developers.google.com/ml-kit/language/translation/translation-language-support
- **Simple translations:** online (on device)
- **More complex translations:** use cloud API

Entity Extraction

Ref: https://developers.google.com/ml-kit/language/entity-extraction

- Recognizes entities within static texts
  - In multiple languages including Arabic, English, Spanish, Dutch, Thai, Turkish, Polish, etc
- Various actions can be suggested based on recognized entities
- Currently supported entities
- **Note:** IBAN is international numbering scheme for banks

Entity	Example
Address	350 third street, Cambridge MA
Date-Time	2019/09/29, let's meet tomorrow at 6pm
Email address	entity-extraction@google.com
Flight Number (IATA flight codes only)	LX37
IBAN	CH52 0483 0000 0000 0000 9
ISBN (version 13 only)	978-1101904190
Money/Currency (Arabic numerals only)	$12, 25 USD
Payment / Credit Cards	4111 1111 1111 1111
Phone Number	(555) 225-3556 12345
Tracking Number (standardized international formats)	1Z204E380338943508
URL	www.google.com https://en.wikipedia.org/wiki/Platypus

Entity Extraction

Ref: https://developers.google.com/ml-kit/language/entity-extraction

- Entity recognition examples

Examples

Input text	Detected entities
Meet me at 1600 Amphitheatre Parkway, Mountain View, CA, 94043 Let's organize a meeting to discuss.	Entity 1 type: Address Entity 1 text: "1600 Ampitheatre Parkway, Mountain View, CA 94043"
You can contact the test team tomorrow at info@google.com to determine the best timeline.	Entity 1 type: Date-Time Entity 1 text: = "June 24th, 2020" Entity 2 type: Email address Entity 2 text: info@google.com
Your order has shipped from Google. To follow the progress of your delivery please use this tracking number: 9612804152073070474837	Entity type: Tracking number Entity text: "9612804152073070474837"
Call the restaurant at 555-555-1234 to pay for dinner. My card number is 4111-1111-1111-1111.	Entity 1 type: Phone number Entity 1 text: "555-555-1234" Entity 2 type: Payment card Entity 2 text: "4111 1111 1111 1111"

Smart Reply

Ref: https://developers.google.com/ml-kit/language/smart-reply

- Automatically generate relevant replies to messages
- Helps users respond quickly
- Facilitates replies on devices with limited input capabilities
- Casual language, only English language currently supported
- Analyzes up to 10 most recent messages, provides up to 3 suggested responses

Example results

Input

Timestamp	User ID	Local User?	Message
Thu Feb 21 13:13:39 PST 2019		true	are you on your way?
Thu Feb 21 13:15:03 PST 2019	FRIEND0	false	Running late, sorry!

Suggested replies

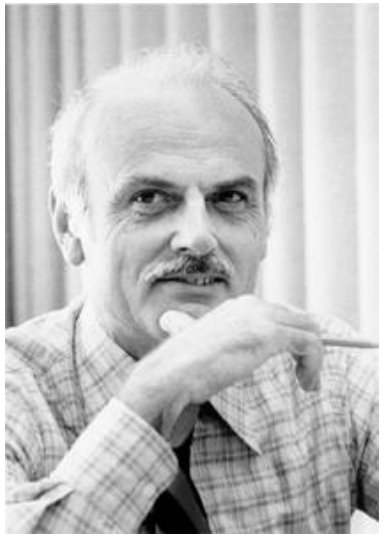
Suggestion #1	Suggestion #2	Suggestion #3
No worries	😔	No problem!

Background on Relational Databases

Background on Databases

- Relational DataBase Management System (RDBMS)
  - Introduced by E. F. Codd (Turing Award Winner in 1981)

- Relational Database
  - data stored in tables
  - relationships among data stored in tables
  - data can be accessed and viewed in various ways



Example Wines Database

- **Relational Data:** Data in different tables can be related

Table of wineries

Winery Table

Winery ID	Winery name	Address	Region ID
1	Moss Brothers	Smith Rd.	3
2	Hardy Brothers	Jones St.	1
3	Penfolds	Arthurton Rd.	1
4	Lindemans	Smith Ave.	2
5	Orlando	Jones St.	1

Table of geographic regions

Region Table

Region ID	Region name	State
1	Barossa Valley	South Australia
2	Yarra Valley	Victoria
3	Margaret River	Western Australia

Ref: Web Database Applications with PHP and MySQL, 2nd Edition , by Hugh E. Williams, David Lane, O'Reilly Media

Keys

- Each table has a key
- **Key:** column used to uniquely identify each row

KEYS

Winery Table

Winery ID	Winery name	Address	Region ID
1	Moss Brothers	Smith Rd.	3
2	Hardy Brothers	Jones St.	1
3	Penfolds	Arthurton Rd.	1
4	Lindemans	Smith Ave.	2
5	Orlando	Jones St.	1

Region Table

Region ID	Region name	State
1	Barossa Valley	South Australia
2	Yarra Valley	Victoria
3	Margaret River	Western Australia

SQL and Databases

- **SQL:** language used to manipulate Relational Database (RDBMS)

- SQL Commands:
  - **CREATE TABLE** - creates new database table
  - **ALTER TABLE** - alters a database table
  - **DROP TABLE** - deletes a database table
  - **SELECT** - retrieve data from a database table
  - **UPDATE** - change data in a database table
  - **DELETE** - remove data from a database table
  - **INSERT INTO** - insert new data in a database table

Region Table

Region ID	Region name	State
1	Barossa Valley	South Australia
2	Yarra Valley	Victoria
3	Margaret River	Western Australia

Database in Android

- **SQLite:** http://www.sqlite.org/
  - Open source relational database
  - Subset of SQL (most but not all)

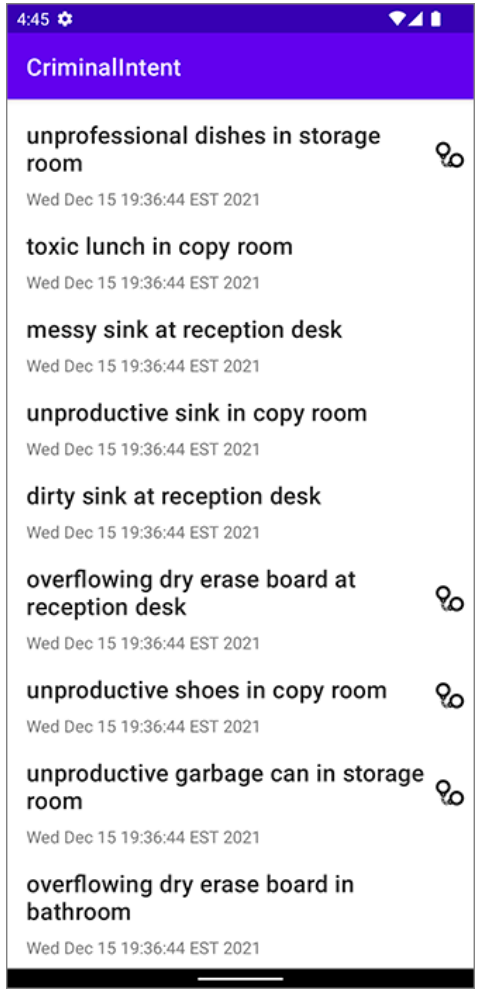
- Android includes a SQLite database
- SQL a bit low level?
- **Newer:** Higher level Android database API called Room

Android Nerd Ranch Ch 12
Co-Routines and Databases (New)

Co-Routines and Databases

ANR (5th edition): Chapter 12

- **ANR Chapter 12:**
  - Implements database for **CriminalIntent**,
  - Seed database with dummy data
- But description in HFAD is clearer.
- So I will now walk through HFAD database description
  - **Note:** students read ANR Ch 12 themselves



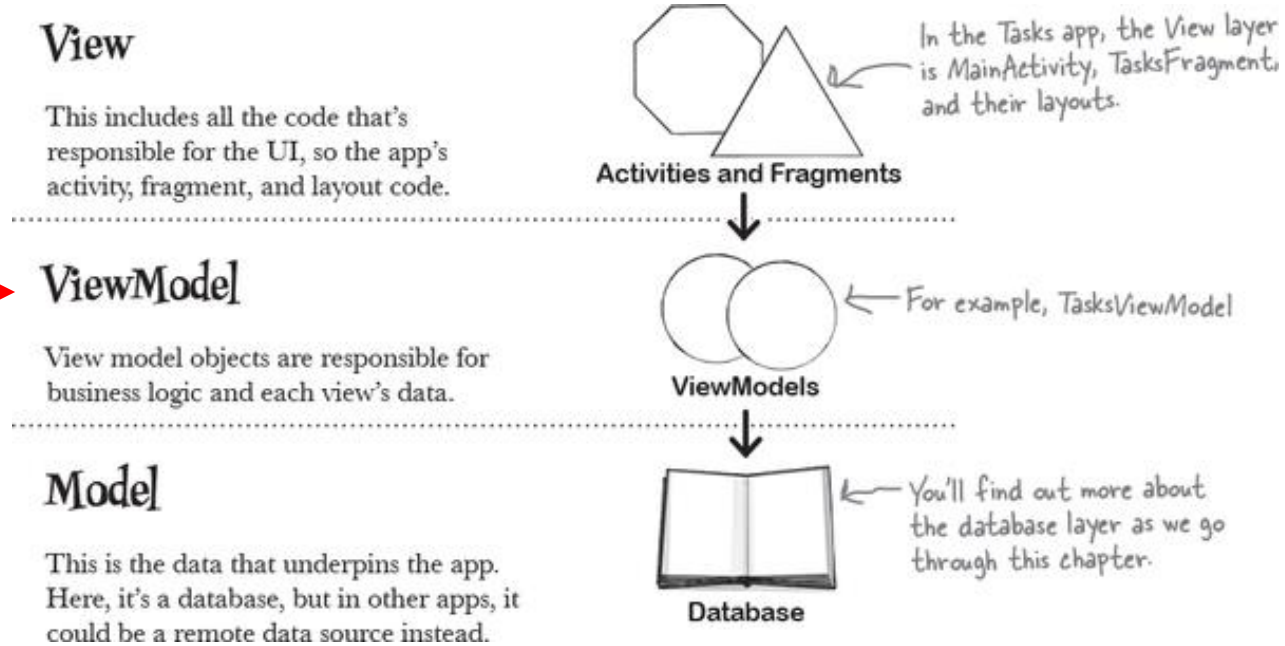
Android Room Databases

Android Room Databases

HFAD (3rd edition), Ch 14, pg 573

- Databases can be used to persist data (i.e store data after app exits)
- Android uses SQLite databases under the hood, lightweight, fast, optimized
- But SQLite can be tricky, error-prone to program directly
- Android Jetpack includes data persistence library called Room, sits on top of SQLite
- Room code is simpler, less error-prone
- Room apps structured as MVVM

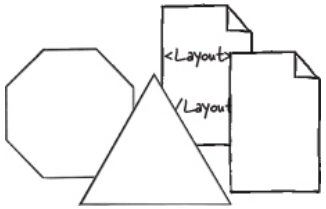
Previously used **ViewModel** to persist across device rotation



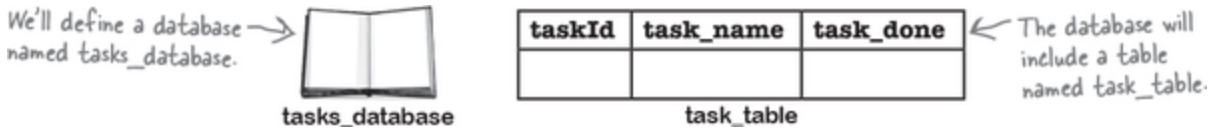
Android Room Databases: High Level Steps

HFAD (3rd edition), Ch 14, pg 573

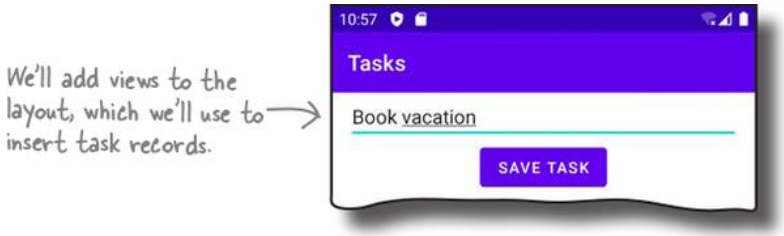
1. **Set up app:**
   - Update build.gradle
   - Create activity, fragment and layout XML file



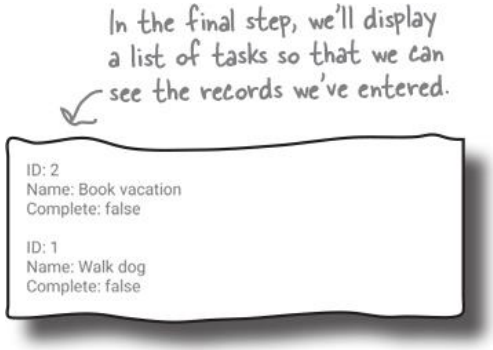
2. **Write database code:**
   - Create database with a table
   - Add methods to interact with table's data



3. **Insert records:**
   - Create view model
   - Configure app's fragment to insert records



4. **Display list of records:**
   - Update view model and fragment code to display all task records in database



Android Room Databases: How Room Databases are Created

HFAD (3rd edition), Ch 14

- Room uses annotated classes and interfaces to create and configure SQLite database. Needs

1. **Database class:**
   - Defines database including name and version number

Database

2. **Data classes for tables:**
   - All database data stored in tables
   - Define a table to specify table's name and columns

taskId	task_name	task_done
1	Walk dog	false
2	Book vacation	false
3	Arrange picnic	false

Table

3. **Interfaces for data access:**
   - App interacts with each table using interfaces, which specify data access methods. Example methods:
     - **insert( )** method to insert records
     - **getAll( )** method to get all records

Data Access Interface

Android Room Databases: How Room Databases are Created

HFAD (3rd edition), Ch 14

- Create relational tables: separate table for each type of data
- Example task table:

These are records, which the user will enter using the app.

taskId	task_name	task_done
1	Walk dog	false
2	Book vacation	false
3	Arrange picnic	false

The table has three columns: taskId, task_name, and task_done.

- Define data class for each table
  - Declare a property (variable) for each of table's columns

(Data) Task
taskId: Long
taskName: String
taskDone: Boolean

The Task data class includes a property for each of the table's columns.

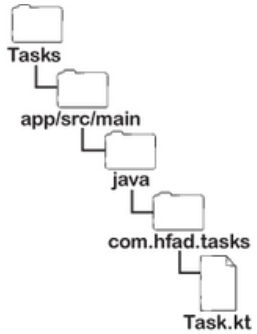
- Create the Task data class

```
package com.hfad.tasks
```

Make sure that you make Task a data class.

```
data class Task(
    var taskId: Long = 0L,
    var taskName: String = "",
    var taskDone: Boolean = false
)
```

Add these three properties.



Android Room Databases: How Room Databases are Created

HFAD (3rd edition), Ch 14

- Give table a name (e.g. task_table) using @Entity

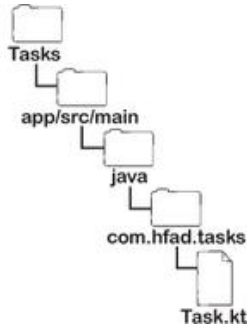
```
...
@Entity(tableName = "task_table")
data class Task(
...
```

This tells Room that the class describes the task_table entity.

- Specify the primary key using @PrimaryKey

```
...
    @PrimaryKey(autoGenerate = true)
    var taskId: Long = 0L,
...
```

taskId is the primary key, and Room will autogenerate its values.



- Specify names of "other" columns using @columnInfo

```
...
    @ColumnInfo(name = "task_name")
    var taskName: String = "",

    @ColumnInfo(name = "task_done")
    var taskDone: Boolean = false
}
```

These annotations override the column names that will be used for these two properties.

Android Room Databases: How Room Databases are Created

HFAD (3rd edition), Ch 14

- Complete code:

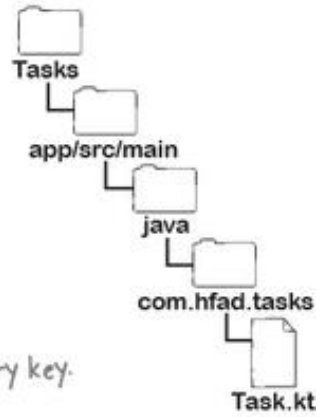
```
package com.hfad.tasks

import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "task_table")
data class Task(

    @PrimaryKey(autoGenerate = true)
    var taskId: Long = 0L,

    @ColumnInfo(name = "task_name")
    var taskName: String = "",

    @ColumnInfo(name = "task_done")
    var taskDone: Boolean = false
)
```

Import these classes.

Name the table.

Specify the primary key.

Name these two columns.



- Table looks like this:

taskId	task_name	task_done

task_table

Android Room Databases: How Room Databases are Created

HFAD (3rd edition), Ch 14

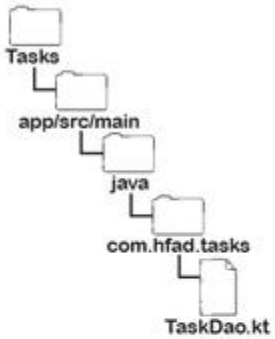
- Allow access to database table data using **Data Access Object (DAO)** that includes relevant methods
- E.g. insert, read, update, delete and query:

```
package com.hfad.tasks

import androidx.lifecycle.LiveData
import androidx.room.Dao
import androidx.room.Delete
import androidx.room.Insert
import androidx.room.Query
import androidx.room.Update

@Dao
interface TaskDao {
    @Insert
    fun insert(task: Task)

    @Update
    fun update(task: Task)

    @Delete
    fun delete(task: Task)

    @Query("SELECT * FROM task_table WHERE taskId = :taskId")
    fun get(taskId: Long): LiveData<Task>

    @Query("SELECT * FROM task_table ORDER BY taskId DESC")
    fun getAll(): LiveData<List<Task>>
}
```

Import these classes.

Mark the interface with @Dao

Include these data access methods



Android Room Databases: How Room Databases are Created

HFAD (3rd edition), Ch 14

- Query method can be used for everything else. E.g.
  - The following **get( )** method returns record with matching taskId

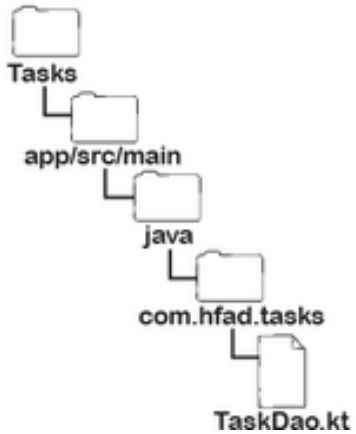
```
@Query("SELECT * FROM task_table WHERE taskId = :taskId")
fun get(taskId: Long): LiveData<Task>
```

This returns the task record with a certain taskId.

  - The following **getAll( )** method returns list of all records in table

```
@Query("SELECT * FROM task_table ORDER BY taskId DESC")
fun getAll(): LiveData<List<Task>>
```

This returns all the task records in descending order of taskId.



Android Room Databases: How Room Databases are Created

HFAD (3rd edition), Ch 14

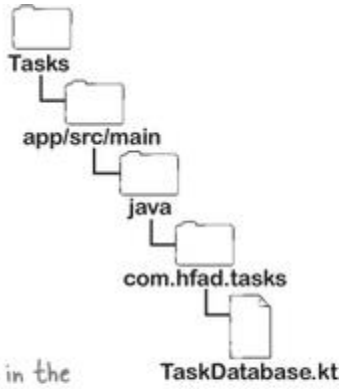
- Remember: A database contains tables. Need to create a database
- Define database by creating abstract class, which extends **RoomDatabase**

```
package com.hfad.tasks

import androidx.room.RoomDatabase

abstract class TaskDatabase : RoomDatabase() {
}
```

Import this class.

Make sure the class extends RoomDatabase.

- Then add tables previously created to the database



Recall: we previously declared Task data class

```
@Entity(tableName = "task_table")
data class Task(

    @PrimaryKey(autoGenerate = true)
    var taskId: Long = 0L,

    @ColumnInfo(name = "task_name")
    var taskName: String = "",

    @ColumnInfo(name = "task_done")
    var taskDone: Boolean = false
)
```

Name the table.

Specify the primary key.

Name these two columns.

```
package com.hfad.tasks

import androidx.room.Database
import androidx.room.RoomDatabase

@Database(entities = [Task::class], version = 1, exportSchema = false)
abstract class TaskDatabase : RoomDatabase() {
}
```

Import the Database class.

This adds the table defined in the Task data class to the database.

Specifies classes (marked with @entity) corresponding to tables to add to database

Version is **int** that Specifies database version

Tells Room whether to Export database schema

Android Room Databases: How Room Databases are Created

HFAD (3rd edition), Ch 14

- Recall, we previously created DAO interface

```
@Dao ← Mark the interface with @Dao
interface TaskDao {
    @Insert
    fun insert(task: Task)

    @Update
    fun update(task: Task)

    @Delete
    fun delete(task: Task)

    @Query("SELECT * FROM task_table WHERE taskId = :taskId")
    fun get(taskId: Long): LiveData<Task>

    @Query("SELECT * FROM task_table ORDER BY taskId DESC")
    fun getAll(): LiveData<List<Task>>
}
```

Include these data access methods

- Tell Room that the DAO interface previously defined will be used for data access

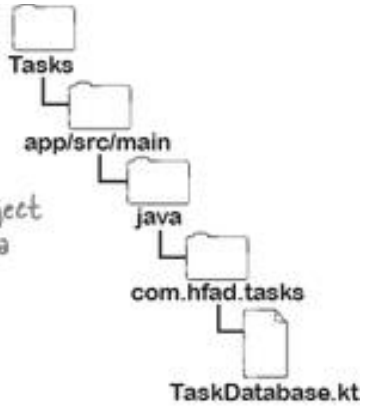
```
...
@Database(entities = [Task::class], version = 1, exportSchema = false)
abstract class TaskDatabase : RoomDatabase() {
    abstract val taskDao: TaskDao
}
```

This tells Room that you want to use the data access methods specified in TaskDao.

Android Room Databases: How Room Databases are Created

HFAD (3rd edition), Ch 14

- Create and return instance of the database




```
...
abstract class TaskDatabase : RoomDatabase() {
    ...
    companion object {
        @Volatile
        private var INSTANCE: TaskDatabase? = null

        fun getInstance(context: Context): TaskDatabase {
            synchronized(this) {
                var instance = INSTANCE
                if (instance == null) {
                    instance = Room.databaseBuilder(
                        context.applicationContext,
                        TaskDatabase::class.java,
                        "tasks_database"
                    ).build()
                    INSTANCE = instance
                }
                return instance
            }
        }
    }
}
```

We're putting getInstance() in a companion object so that we can call it without first creating a TaskDatabase. You'll see why later on.

This method is used to get an instance of the database. It will look nearly the same for every database you create.

If the database doesn't already exist, the getInstance() method goes and builds it.

Return an instance of TaskDatabase.

Android Room Databases

HFAD (3rd edition), Ch 14, pg 573

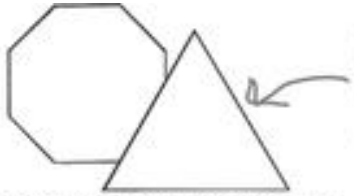
- Recall MVVM model

Next, write code for ViewModel

Now finished writing code for the Model

View

This includes all the code that's responsible for the UI, so the app's activity, fragment, and layout code.

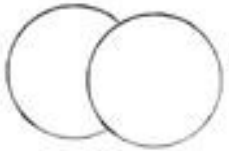


Activities and Fragments

In the Tasks app, the View layer is MainActivity, TasksFragment, and their layouts.

ViewModel

View model objects are responsible for business logic and each view's data.



ViewModels

For example, TasksViewModel

Model

This is the data that underpins the app. Here, it's a database, but in other apps, it could be a remote data source instead.

Database

You'll find out more about the database layer as we go through this chapter.

Android Room Databases

HFAD (3rd edition), Ch 14, pg 573

- Create a View Model

```
package com.hfad.tasks

import androidx.lifecycle.ViewModel

class TasksViewModel(val dao: TaskDao) : ViewModel() {
    var newTaskName = ""

    fun addTask() {
        val task = Task()
        task.taskName = newTaskName
        dao.insert(task)
    }
}
```

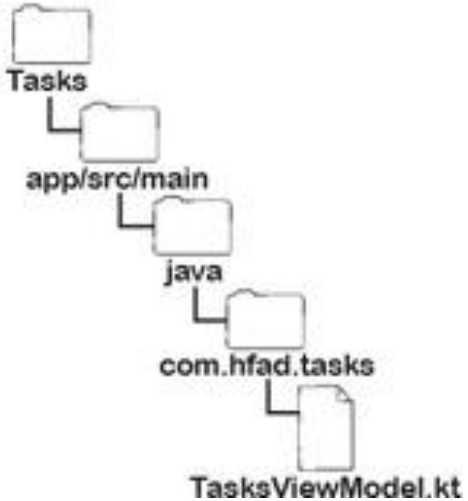
Import this class.

Make TasksViewModel extend ViewModel.

Pass a TaskDao object to TasksViewModel.

This is for the task name.

This method creates a Task object, and uses the TaskDao's insert() method to add it to the database.



Android Room Databases

HFAD (3rd edition), Ch 14

- But database operations (e.g. inserting records into database) can be very low
- Android rules:
  - UI must be in main thread
  - Database and networking code must be in background thread, cannot be in main thread
    - Why? Can take a long time to complete or block
- Co-routines, kotlin solution, like a lightweight thread used define work to be run asynchronously
  - Database operations (e.g. insert a record into a database) can be launched as background job, rest of code can continue to run
  - Functions can suspend: paused till long-running operation completes
  - When code in a co-routine is suspended, thread is free to execute other things (e.g. draw UI, respond to touch events, etc.)
  - Co-routines use threads under the hood but programmer does not need to know details

Android Room Databases

HFAD (3rd edition), Ch 14

- Co-routines are suspendable functions that can run in background. How?
  1. Mark each DAO method (e.g. insert( ), update( )) with suspend

This turns the insert() method into a coroutine.

```
@Insert
suspend fun insert(task: Task)
```

  2. Launch DAO co-routines in background. E.g.

This launches the coroutine in the same scope as the view model.

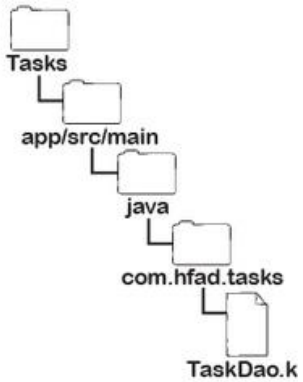
```
viewModelScope.launch {
    ...
    dao.insert(task)
}
```

Scope: defines duration of time co-routine is active/valid that is same as ViewModel

Android Room Databases

HFAD (3rd edition), Ch 14

1. **Complete code:** Mark insert( ), update( ) and delete( ) DAO methods with suspend




```
package com.hfad.tasks

import androidx.lifecycle.LiveData
import androidx.room.Dao
import androidx.room.Delete
import androidx.room.Insert
import androidx.room.Query
import androidx.room.Update

@Dao
interface TaskDao {
    @Insert
    suspend fun insert(task: Task)

    @Update
    suspend fun update(task: Task)

    @Delete
    suspend fun delete(task: Task)

    @Query("SELECT * FROM task_table WHERE taskId = :key")
    fun get(key: Long): LiveData<Task>

    @Query("SELECT * FROM task_table ORDER BY taskId DESC")
    fun getAll(): LiveData<List<Task>>
}
```

Mark the insert(), update(), and delete() methods with suspend.

The get() and getAll() methods use live data, which runs in the background. This means we can keep these methods exactly as they are.

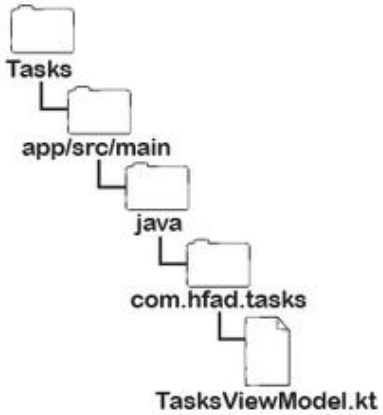
Android Room Databases

HFAD (3rd edition)

2. **Complete code:** Launch DAO co-routines in background.

```
package com.hfad.tasks

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import kotlinx.coroutines.launch

class TasksViewModel(val dao: TaskDao) : ViewModel() {
    var newTaskName = ""

    fun addTask() {
        viewModelScope.launch {
            val task = Task()
            task.taskName = newTaskName
            dao.insert(task)
        }
    }
}
```

Launch the coroutine in the same scope as the view model.



- See rest of Chapter 14 in Head First Android
- Also, read Chapter 12 in Android Nerd Ranch on adding Room database to **CriminalIntent**

Firebase Cloud API

Firebase

- Mobile cloud backend service for
  - Analytics
  - Messaging
  - Authentication
  - Database
  - Crash reporting, etc

- Previously 3rd party company
- Acquired by Google in 2014
  - Now part of Google. See https://firebase.google.com/
  - Fully integrated, could speed up development. E.g. final project

Firebase

- Relatively easy programming, few lines of code
- E.g. to create database

```
FirebaseDatabase database = FirebaseDatabase.getInstance()
// write
database.child("users").child("userId").setValue(user);

// read / listen
database.child("users").addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // ...
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {}
});
```

References

- Android Nerd Ranch, 5th edition
- Head First Android (3rd edition)
- Google Android Online tutorials
- ML Kit Online