COS 470 - MOBILE DEVELOPMENT

# INTRODUCTION

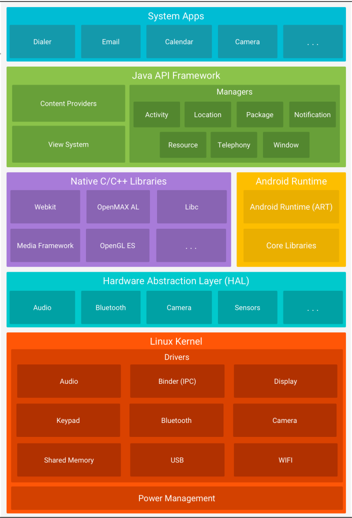
1

---

COS 470 MOBILE DEVELOPMENT

## WHAT IS ANDROID

- Linux-based
- Java/Kotlin
- Android Runtime (ART)
- System Apps
  - SMS, Calendar, etc.



Platform Architecture

2

---

COS 470 MOBILE DEVELOPMENT

## CORE OS

- Linux (64 bit)
  - Each app is a different user
  - Processes run in virtual machines
- JVM and View System (UI)
- Resource, Notification, and Activity Managers
- Content Providers (contacts, etc.)

3

4

COS 470 MOBILE DEVELOPMENT

## APP COMPONENTS – MANIFEST.XML

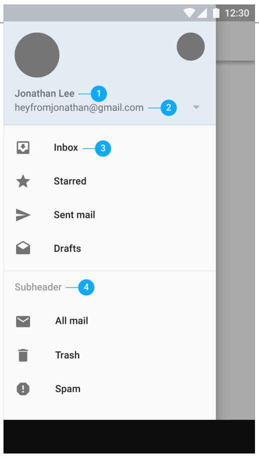
- Activities (launched by Intents)
  - The main entry point and interactive aspects
- Services
  - Background tasks
- Broadcast Receivers
  - Alarms, "push" data
- Content Providers
  - Shared data from the app, e.g. contacts

5

COS 470 MOBILE DEVELOPMENT

## MATERIAL DESIGN

- Visual language to describe experience
- Principles
  - Is a metaphor
  - Bold, graphic, intentional
  - Motion provides meaning



Material Design

6

COS 470 MOBILE DEVELOPMENT

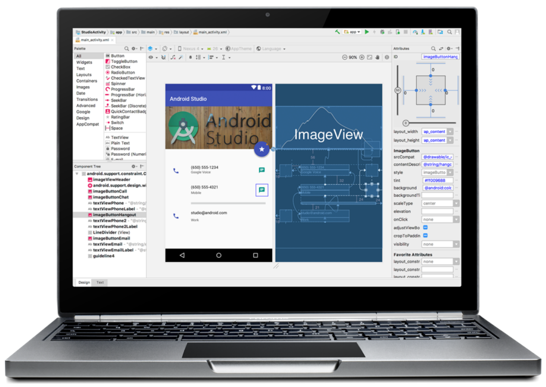
## PLATFORM AND DEVELOPMENT

- Tools
- Languages
- Frameworks
- Design Strategy

COS 470 MOBILE DEVELOPMENT

## DEVELOPMENT TOOLS

Android Studio

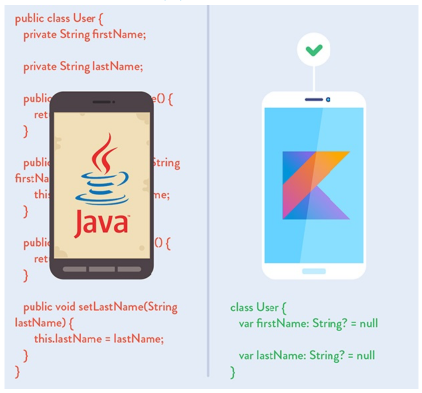


COS 470 MOBILE DEVELOPMENT

## DEVELOPMENT LANGUAGE(S)

COS 470 MOBILE DEVELOPMENT

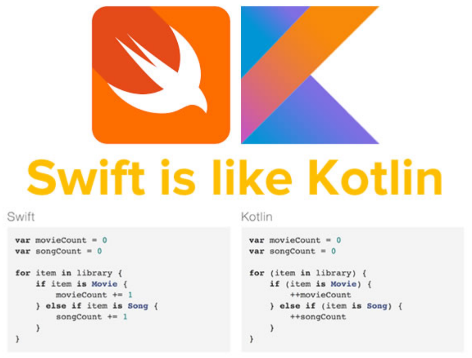
## DEVELOPMENT LANGUAGE(S)



```
public class User {
  private String firstName;

  private String lastName;

  publi          e() {
    ret
  }

  publi          String
first:Na
    thi          me;
  }

  publi          () {
    ret
  }

  public void setLastName(String
lastName) {
    this.lastName = lastName;
  }
}
```

```
class User {
  var firstName: String? = null

  var lastName: String? = null
}
```

## DEVELOPMENT LANGUAGE(S)



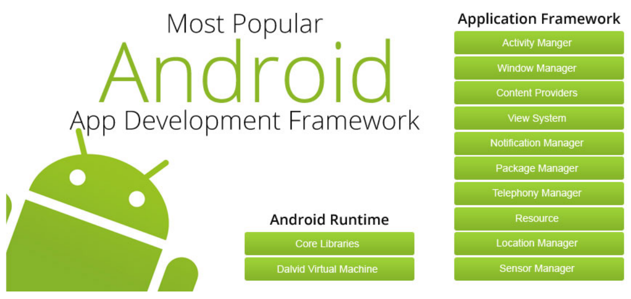
Swift

```
var movieCount = 0
var songCount = 0

for item in library {
    if item is Movie {
        movieCount += 1
    } else if item is Song {
        songCount += 1
    }
}
```

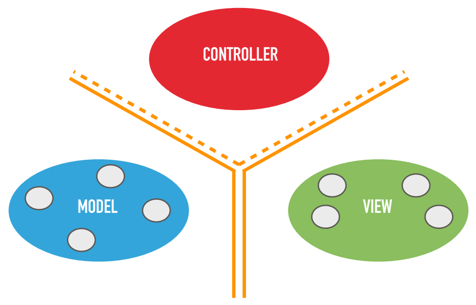
Kotlin

```
var movieCount = 0
var songCount = 0

for (item in library) {
    if (item is Movie) {
        ++movieCount
    } else if (item is Song) {
        ++songCount
    }
}
```

## DEVELOPMENT FRAMEWORKS



## DESIGN STRATEGY - MODEL-VIEW-CONTROLLER (MVC)



## ANDROID STUDIO DEMO 1 – THE BASICS

13

---

## ANDROID STUDIO DEMO 1 – THE BASICS

- Task List Sample
  - Android Studio 3.0.1 (latest)
  - Java (not Kotlin… yet)
  - Android Simulator (no phone needed)
  - Controller and View

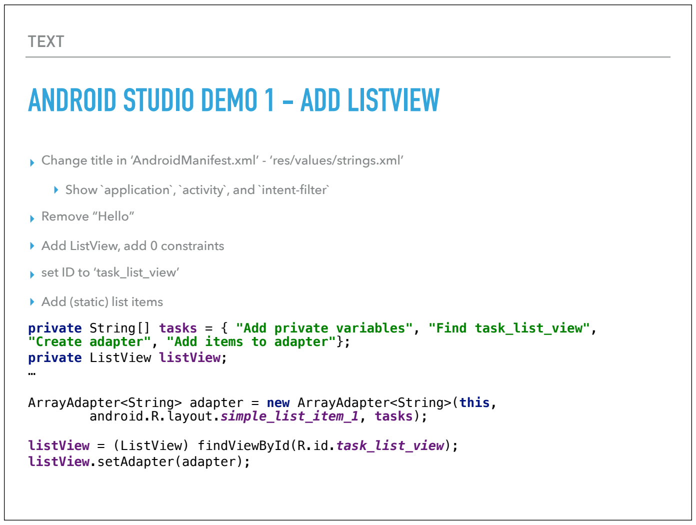
14

---

## ANDROID STUDIO DEMO 1 – THE BASICS

- Start a new project, java-based,
- phone and tablet - show choice and target %
- API 19 KitKat (90%)
- Basic Activity (with floating action button)
- Rename TaskListActivity
- Explore Android Studio interface
- Compile and run…
  - Create new virtual device, configure API level and image
  - Choose a recommended one with Google API's

15

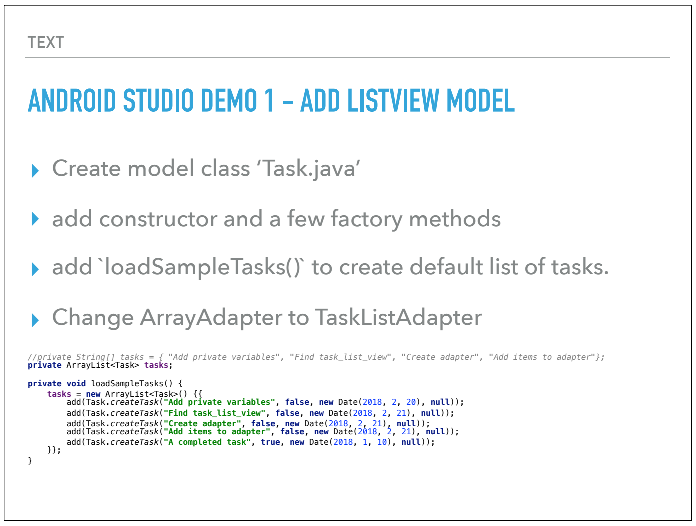

TEXT

## ANDROID STUDIO DEMO 1 – ADD LISTVIEW

- Change title in ‘AndroidManifest.xml’ - ‘res/values/strings.xml’
  - Show `application`, `activity`, and `intent-filter`
- Remove “Hello”
- Add ListView, add 0 constraints
- set ID to ‘task_list_view’
- Add (static) list items

```
private String[] tasks = { "Add private variables", "Find task_list_view",
"Create adapter", "Add items to adapter"};
private ListView listView;
…

ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, tasks);

listView = (ListView) findViewById(R.id.task_list_view);
listView.setAdapter(adapter);
```

16

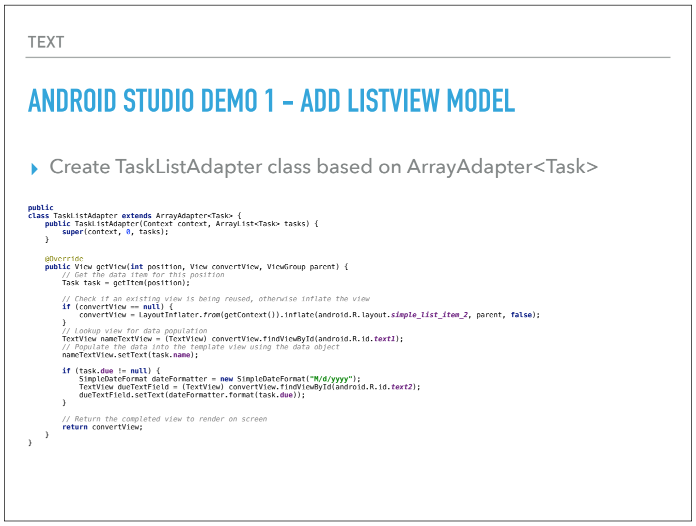

TEXT

## ANDROID STUDIO DEMO 1 – ADD LISTVIEW MODEL

- Create model class ‘Task.java’
- add constructor and a few factory methods
- add `loadSampleTasks()` to create default list of tasks.
- Change ArrayAdapter to TaskListAdapter

```
//private String[] tasks = { "Add private variables", "Find task_list_view", "Create adapter", "Add items to adapter"};
private ArrayList<Task> tasks;

private void loadSampleTasks() {
    tasks = new ArrayList<Task>() {{
        add(Task.createTask("Add private variables", false, new Date(2018, 2, 20), null));
        add(Task.createTask("Find task_list_view", false, new Date(2018, 2, 21), null));
        add(Task.createTask("Create adapter", false, new Date(2018, 2, 21), null));
        add(Task.createTask("Add items to adapter", false, new Date(2018, 2, 21), null));
        add(Task.createTask("A completed task", true, new Date(2018, 1, 10), null));
    }};
}
```

17

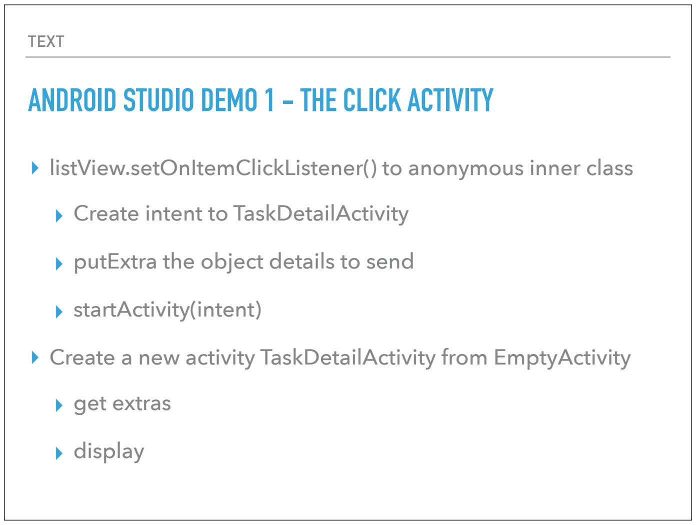

TEXT

## ANDROID STUDIO DEMO 1 – ADD LISTVIEW MODEL

- Create TaskListAdapter class based on ArrayAdapter<Task>

```
public
class TaskListAdapter extends ArrayAdapter<Task> {
    public TaskListAdapter(Context context, ArrayList<Task> tasks) {
        super(context, 0, tasks);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        // Get the data item for this position
        Task task = getItem(position);

        // Check if an existing view is being reused, otherwise inflate the view
        if (convertView == null) {
            convertView = LayoutInflater.from(getContext()).inflate(android.R.layout.simple_list_item_2, parent, false);
        }
        // Lookup view for data population
        TextView nameTextView = (TextView) convertView.findViewById(android.R.id.text1);
        // Populate the data into the template view using the data object
        nameTextView.setText(task.name);

        if (task.due != null) {
            SimpleDateFormat dateFormatter = new SimpleDateFormat("M/d/yyyy");
            TextView dueTextField = (TextView) convertView.findViewById(android.R.id.text2);
            dueTextField.setText(dateFormatter.format(task.due));
        }

        // Return the completed view to render on screen
        return convertView;
    }
}
```

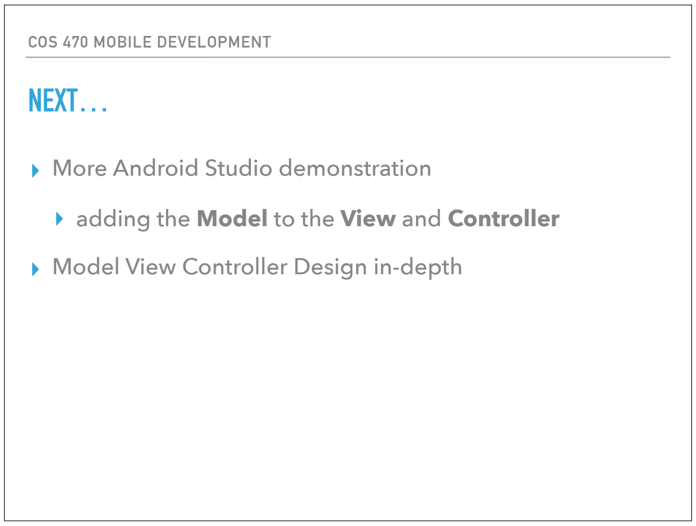
18



19

## ANDROID STUDIO DEMO 1 – THE CLICK ACTIVITY

- listView.setOnItemClickListener() to anonymous inner class
  - Create intent to TaskDetailActivity
  - putExtra the object details to send
  - startActivity(intent)
- Create a new activity TaskDetailActivity from EmptyActivity
  - get extras
  - display



20

## NEXT…

- More Android Studio demonstration
  - adding the **Model** to the **View** and **Controller**
- Model View Controller Design in-depth

21