BSE2209
MOBILE PROGRAMMING PROJECT

In Last Lecture

Introduction to mobile programming

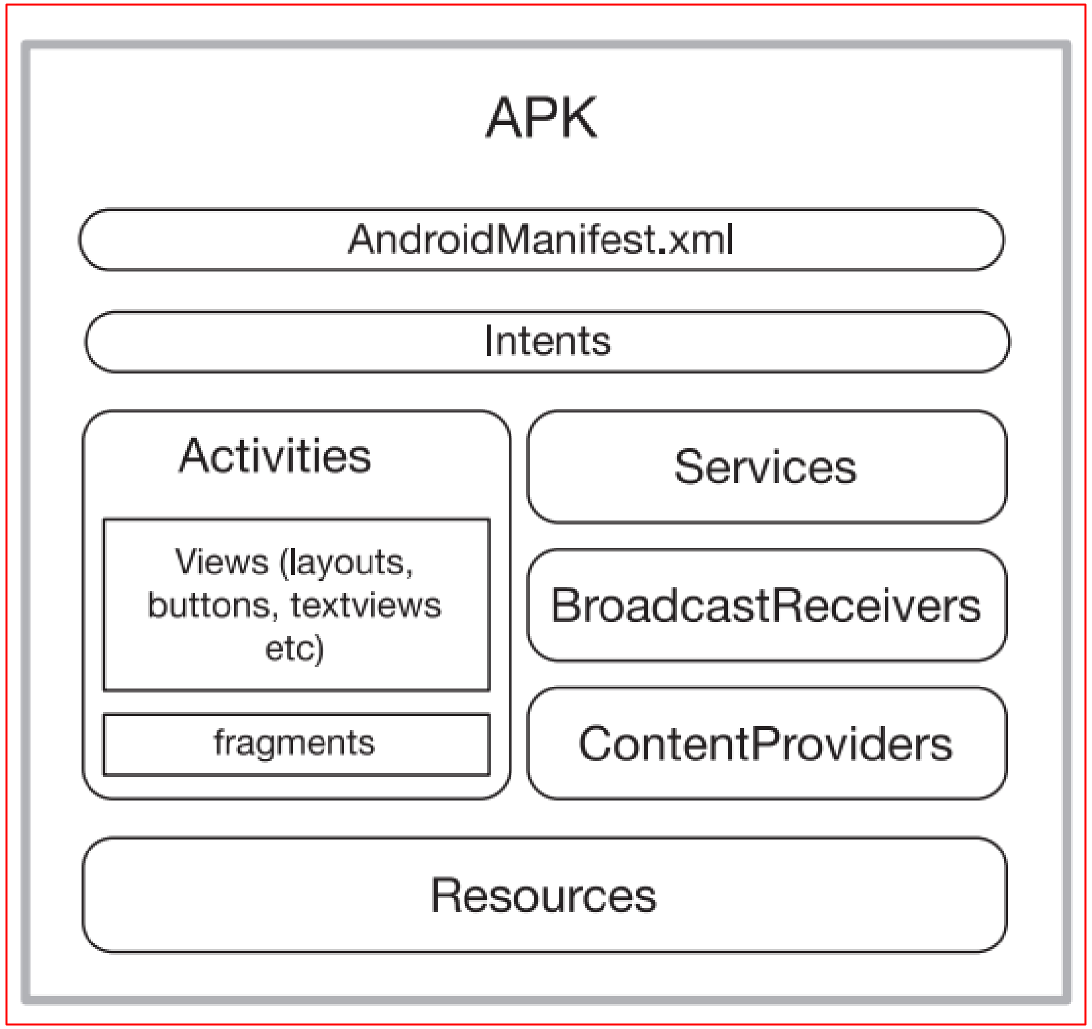
- Evolution of mobile application development
- Mobile Programming ecosystems: Apple, Google, Microsoft
- Hybrid mobile applications

In This Lecture

- Android components
- Activities and layouts
- View and ViewGroup objects
- Activity lifecycle
- Containers

Contents of an Android App

- Android app contains loosely assembled components and other resources.
- These are held together inside an Android Package file or APK.
- Activities, Services, BroadcastReceivers, and ContentProviders are called Android components



Contents of an Android App

- Activity: This is where we put together things that the user can see or do.
- Inside the Activity, there are Views and Fragments.
- Views are classes that are used to draw content into the screen; some examples of View objects are buttons and textviews.
- A Fragment is similar to an Activity; it is also a composition unit but a smaller one.
- Fragments can also hold View objects

Contents of an Android App

- **Fragments** can be turned on or off depending on available screen real estate and/or orientation
- Services: Services are codes that run in the background.
- **BroadcastReceivers** allow an application to listen for specific messages from either the Android system or from other applications
- **ContentProviders** allow applications to share data to other applications.

Contents of an Android App

- **Resources** are visual or audio assets your app may need.
- **The AndroidManifest** declares a few things about the application:
  - name of app
  - Which Activity will show up first when the user launches the app
  - What kind of components are in the app.
  - If it has activities, the manifest declares them—names of classes and all.
  - If the app has services, their class names will also be declared in manifest.
  - What kinds of things can the app do? What are its permissions? Is it allowed to access the internet or the camera? Can it record GPS locations?
  - Does it use external libraries? What version(s) of Android will this app run on?

Component Activation

- An app is just a collection of components held together by a manifest file.
- Each of the components can be activated by sending a message to it.
- Android Intents is a component activation mechanism. The implement a message-passing mechanism you use if you want to activate any android component such as Activity, Service, ContentProvider, or BroadcastReceiver.
- To activate any component, you create an Intent and pass it to the component you want to activate.
- In an application that has more than one Activity, Intents are used to switch control or focus from one Activity to another.

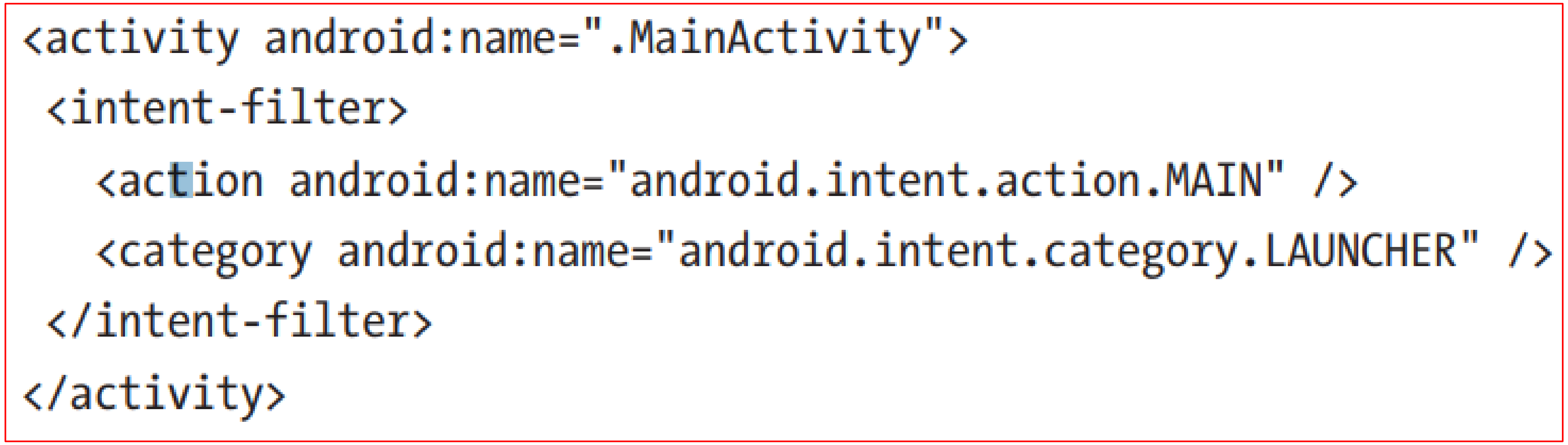
Creating Apps with Android Studio

- We create apps by creating new projects from Android Studio.
- Android uses the Gradle build tool; when the project creation wizard finishes, Gradle pulls several files from internet repositories.
- Projects creations requires to fill in a few fields e.g app name, company domain, and project location, API version

Activities and layouts

- An app that shows a screen to the user requires at least three things:
  - An Activity class that acts as the main program file;
  - A layout file that contains all UI definitions; and
  - A Manifest file, which ties all the project's contents together.
- When an application is launched, the Android runtime creates an Intent object and inspects the manifest file looking for a specific value of the intent-filter node; trying to see if the application has a defined entry point.

Sample Excerpt from AndroidManifest.xmlle



```
<activity android:name=".MainActivity">
 <intent-filter>
   <action android:name="android.intent.action.MAIN" />
   <category android:name="android.intent.category.LAUNCHER" />
 </intent-filter>
</activity>
```

- The first line of the definition has an attribute called android:name. This attribute points to the class name of an Activity.
- The second line declares the intent-filter; the android. intent.action.MAIN, on the intent-filter node, it means the Activity is the entry point for the application.

s
setContentView

The Activity Class

- The main Activity class is responsible for the initial interaction with the user; its a Kotlin or Java class, and in it, we can:
  - Choose which UI file to use. When we call the setContentView(xml:file) function from inside the Activity, it will bind the Activity to xml:file. This is called "Layout binding." When the Activity binds to the layout, the screen will be filled with user interface elements that users can touch or swipe.
  - Get references to view objects. View objects are also called widgets or controls. When we have a programmatic reference to the view objects, we can manipulate them, change their properties, or associate them with an event. This is called View binding.

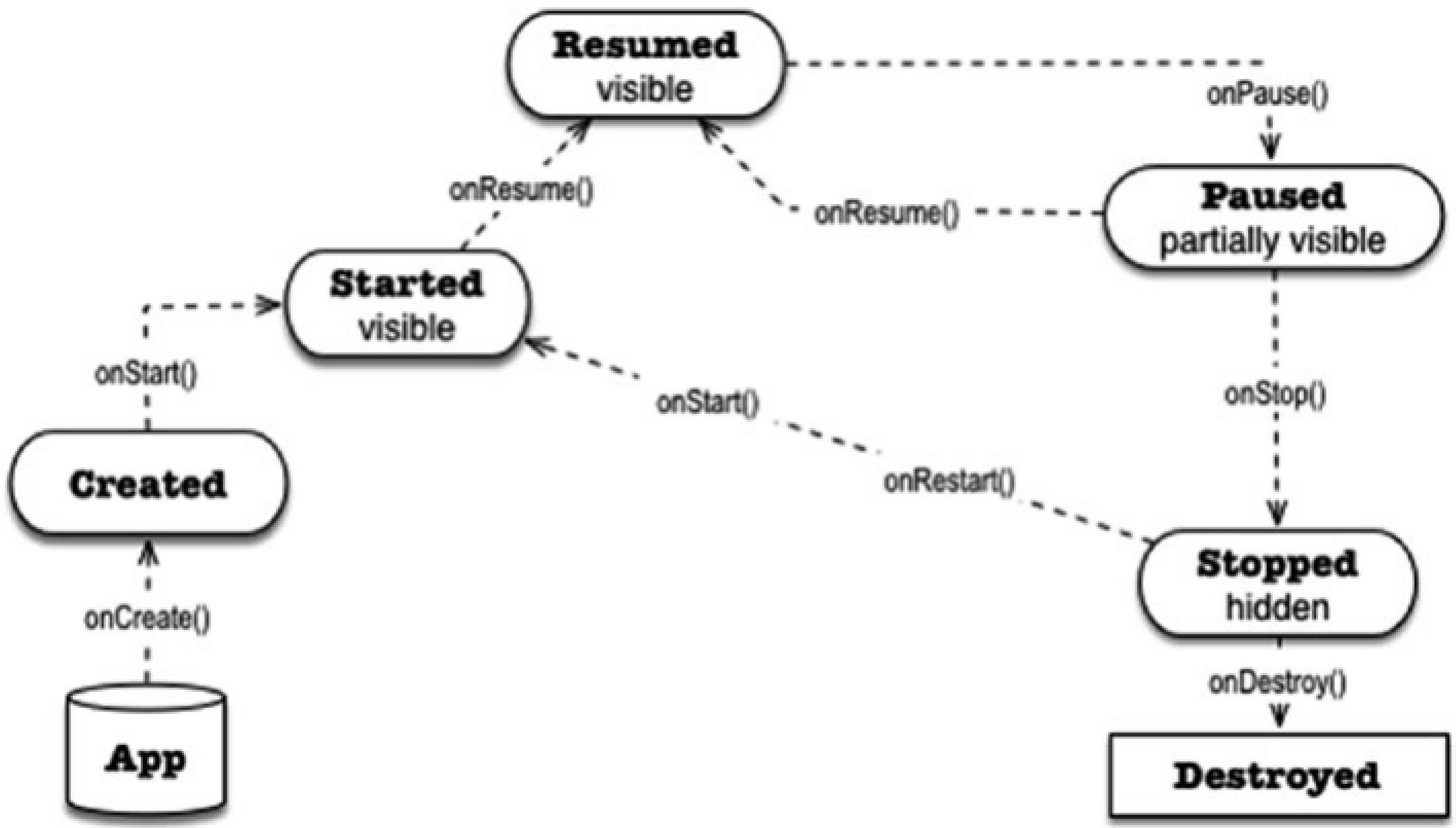
s
setContentView

The Activity Class

- The Activity class inherits from android.app.Activity
- In our example, it inherits from AppCompatActivity; this is a child of FragmentActivity, which in turn is a child of android.app.Activity.
- We use the AppCompatActivity class so we can put modern UI elements like ToolBars in our project, and still run them on older versions of Android where ToolBars are otherwise unsupported—hence, the “Compat” in the name AppCompatActivity.

s
setContentView

Activity Life Cycle



s
setContentView

Activity Life Cycle

- When the runtime launches the app, it calls the onCreate() function of the main Activity, which brings the state of the Activity to "created." You can use this function to perform initialization routines like preparing event handling codes, etc.
- The Activity will proceed to the next state, which is "**started**"; the Activity is visible to the user at this point, but it's not yet ready for interaction.

s
setContentView

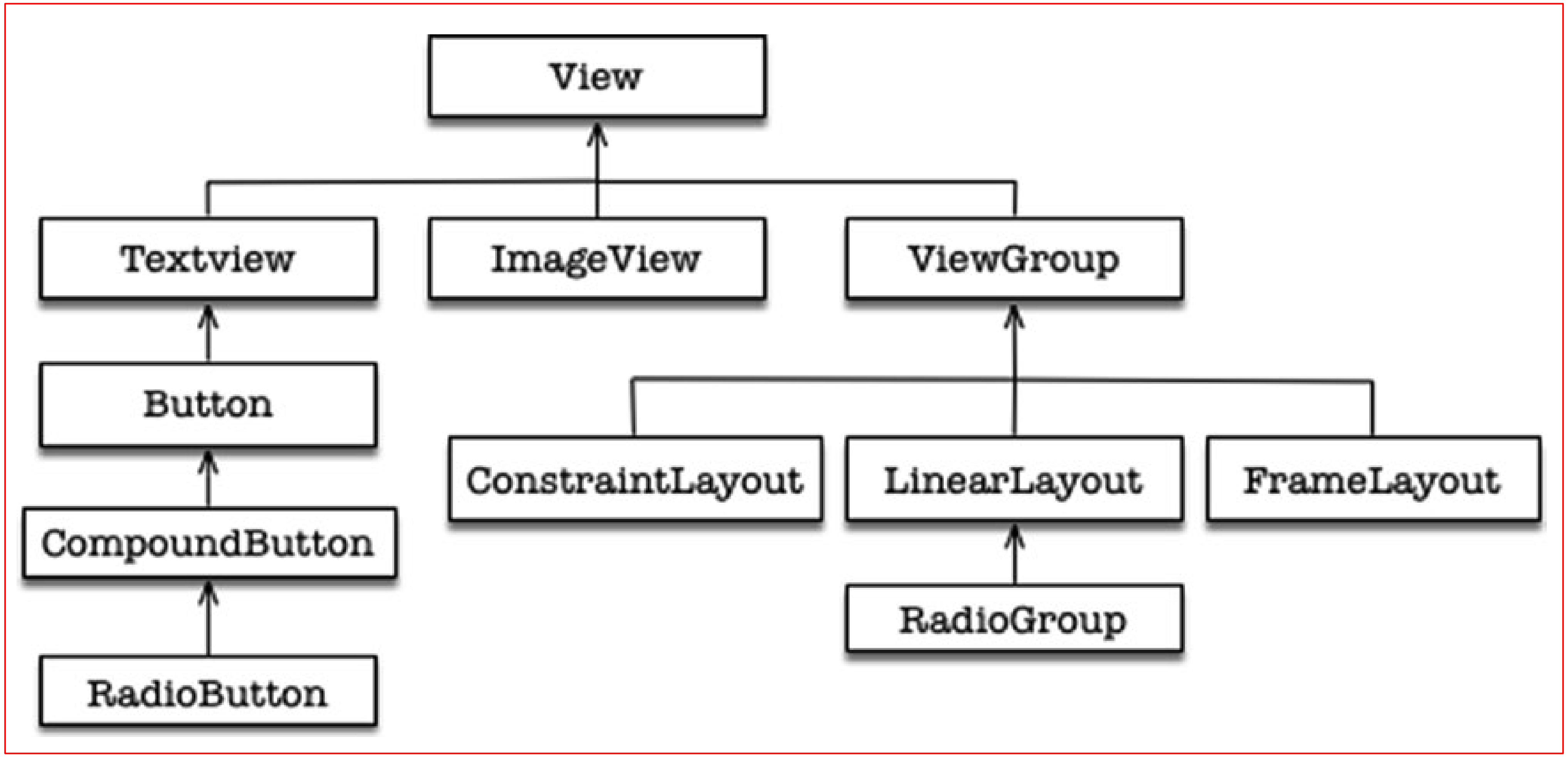
The Layout File

- A layout file contains view objects that are arranged in an XML hierarchy
- The user interface elements like buttons or text fields are written inside an XML file.

View and ViewGroup Objects

- A view object is a composition unit. You build a UI by arranging one or more view objects alongside each other, or sometimes embedded in each other.
- There are two kinds of views;a “view” and a “view group.”
- An example of a View object is a button or a text field.
- A **ViewGroup**, can contain child views; they’re sometimes called containers

View and ViewGroup Objects



Containers

- The ViewGroup class form as the basis for layout managers.
- A layout manager is a container that's responsible for controlling how child views are positioned on the screen, relative to the container and to each other.
- Android comes with a couple of pre-built layout managers.

Containers

Layout Manager	Description
LinearLayout	positions the widgets in single row or column, depending on the selected orientation. Each widget can be assigned a weight value that determines the amount of space the widget occupies compared to the other widgets.
TableLayout	arranges the widgets in a grid format of rows and columns
FrameLayout	stacks child views on top of each other. The last entry on the XML layout file is the one on top of the stack.

Containers

Layout Manager	Description
RelativeLayout	Views are positioned relative to other views and the container by specifying alignments and margins on each view
ConstraintLayout	The ConstraintLayout is the newest layout. It also positions widgets relative to each other and the container (like RelativeLayout). But it accomplishes the layout management by using more than just alignments and margins. It introduces the idea of a "constraint" object which anchors a widget to target. This target could be another widget or a container; or another anchor point.

Reading Tasks

Read and prepare a presentation about the following topics on Kotlin Programming Language:

- Kotlin Basics **(Groups 1,2,3,4,13,14,31,32)**
  - (Literals, Variables, Experessions, Statements, Keywords, Operators, Blocks)
  - Basic Types (Numbers, Literal Constants, Characters, Booleans, Arrays, Strings and String Templates)
  - Flow Control (ifs, when statement, while statement, for loops)
  - Exception Handling
  - Handling nulls

Reading Tasks

- Functions **(Groups 5,6,7,8,16,18,30)**
  - Declaring functions
  - Default arguments
  - Named Arguments
  - Variable Number of Arguments
  - Extension Functions
  - Infix Functions
  - Operator Overloading

Reading Tasks

(Groups 9,10,11,12,19,20,29)

- Interfaces
- Classes
  - Constructors
  - Inheritance
- Data Classes
- Visibility Modifiers
- Access Modifiers
- Object Declarations

Reading Tasks

(Groups 21,22,23,24,25,26,27,28)

- Arrays
- Collections
  - Lists
  - Sets
  - Maps
  - Collections Traversal
- Filter and Map