7. Homework

Answer these questions

Question 1

What are some of the differences between fragments and activities? Select all the statements that are true.

* When creating a fragment, you inflate the layout in the onCreateView() method. When creating an activity, you inflate the layout in onCreate().
* An activity has its own layout, but a fragment cannot have its own layout.
* An activity has its own lifecycle, but a fragment doesn't.
* When inflating the layout for either a fragment or an activity, you can reference the layout as R.layout.*layoutname*.

**Answer 1) The first one is true that is:**

* **When creating a fragment, you inflate the layout in the onCreateView() method. When creating an activity, you inflate the layout in onCreate().**

Question 2

Which of the following statements about fragments are true? Select all that apply.

* You can use a fragment in more than one activity.
* One activity can have multiple fragments.
* After you define a fragment in a Kotlin class, the fragment is automatically added to the activity\_main.xml layout file.
* Use the <fragment> tag to define the place in a layout file where a fragment is to be inserted.

**Answer 2) The first, second and fourth one is True that is:**

* **You can use a fragment in more than one activity.**
* **One activity can have multiple fragments.**
* **Use the <fragment> tag to define the place in a layout file where a fragment is to be inserted.**

**1. Welcome**

In this lab, you learn about fragments, and you create a fragment inside a starter app called AndroidTrivia. In the next codelab, you learn more about navigation and do further work on the AndroidTrivia app.

What you should already know

* The fundamentals of Kotlin
* How to create basic Android apps in Kotlin
* How to work with layouts

What you'll learn

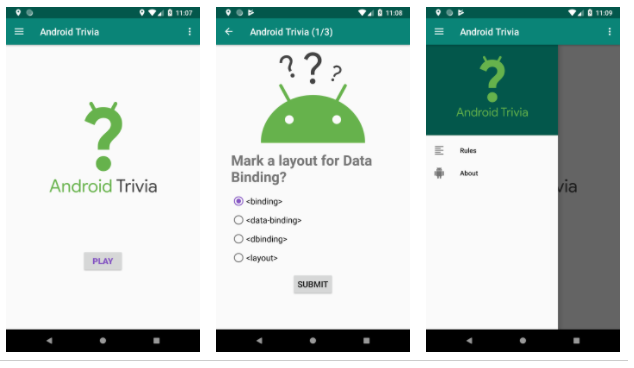
* How to add a Fragment statically to your app

What you'll do

* Create a fragment inside an activity.

## 2. App overview

We will work on an app called AndroidTrivia, an app is a game in which the user answers three trivia questions about Android coding. If the user answers all three questions correctly, they win the game and can share their results.



The AndroidTrivia app illustrates navigation patterns and controls. The app has several components:

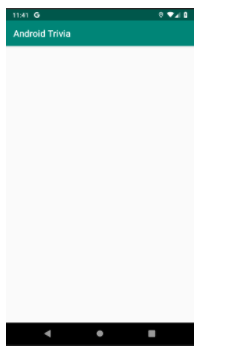
* In the title screen, shown on the left in the screenshot above, the user starts the game.
* In the game screen with questions, shown in the middle above, the user plays the game and submits their answers.
* The navigation drawer, shown on the right above, slides out from the side of the app and contains a menu with a header. The drawer icon https://www.dcs.bbk.ac.uk/~gr/muc/labs-2020/mciot_labs/codelabs/kotlin-android-training-create-and-add-fragment/img/7277f85db3a1ad13.png opens the navigation drawer. The navigation-drawer menu contains a link to the About page and a link to the rules of the game.

The top of the app displays a colored view called the *app bar*, which is also known as the *action bar*.

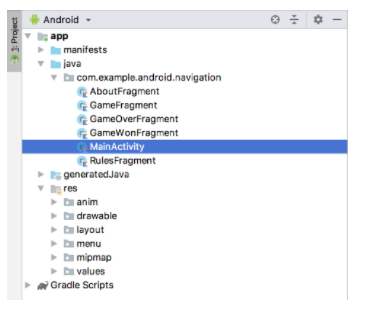
## 3. Task: Explore the starter app project

In this codelab, you work from a starter app that provides template code and fragment classes that you need as you complete the Trivia app.

1. Download the [AndroidTrivia-Starter](https://github.com/google-developer-training/android-kotlin-fundamentals-starter-apps/archive/e85e5ac704de119585920d2db98d186e5726674d.zip" \t "_blank) Android Studio project. Please only use this link as it will enable you to access a version of the source code that works with Studio 3.6.
2. Open the project in Android Studio and run the app. When the app opens, it doesn't do anything other than display the app name and a blank screen.

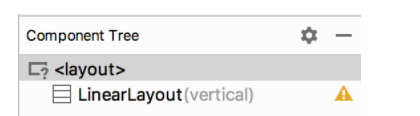


**3** In the Android Studio Project pane, open the Project: Android view to explore the project files. Open the **app > java** folder to see the MainActivity class and fragment classes.



**4** Open the **res > layout** folder and double-click on **activity\_main.xml**. The activity\_main.xml file appears in the Layout Editor.

5 Click the **Design** tab. The **Component Tree** for the activity\_main.xml file shows the root layout as vertical LinearLayout.



In a vertical [linear layout](https://developer.android.com/guide/topics/ui/layout/linear), all the child views in the layout are aligned vertically.

## 4. Task: Add a fragment

A [fragment](https://developer.android.com/guide/components/fragments) represents a behavior or a portion of user interface (UI) in an activity. You can combine multiple fragments in a single activity to build a multi-pane UI, and you can reuse a fragment in multiple activities.

Think of a fragment as a modular section of an activity, something like a "sub activity" that you can also use in other activities:

* A fragment has its own lifecycle and receives its own input events.
* You can add or remove a fragment while the activity is running.
* A fragment is defined in a Kotlin class.
* A fragment's UI is defined in an XML layout file.

The AndroidTrivia app has a main activity and several fragments. Most of the fragments and their layout files have been defined for you. In this task, you create a fragment and add the fragment to the app's main activity.

## Step 1: Add a fragment class

In this step, you create a blank TitleFragment class. Start by creating a Kotlin class for a new fragment:

1. In Android Studio, click anywhere inside the Project pane to bring the focus back to the project files. For example, click the **com.example.android.navigation** folder.
2. Select **File > New > Fragment > Fragment (Blank)**.
3. For the fragment name, use **TitleFragment**. Clear all the checkboxes, including**create Layout XML**, **include fragment factory methods**, and **include interface callbacks**.
4. Click **Finish**.
5. Open the TitleFragment.kt fragment file, if it is not already open. It contains the [onCreateView()](https://developer.android.com/reference/android/app/Fragment.html" \l "onCreateView(android.view.LayoutInflater,%20android.view.ViewGroup,%20android.os.Bundle)" \t "_blank) method, which is one of the methods that's called during [a fragment's lifecycle](https://developer.android.com/reference/android/app/Fragment#Lifecycle).
6. In onCreateView(), remove the return TextView(activity).apply section, including the line that starts with setText. The onCreateView() function is left with only the following code:
7. override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
8. savedInstanceState: Bundle?): View? {
9. }

**Create a binding object**

The fragment won't compile now. To make the fragment compile, you need to create a binding object and inflate the fragment's view (which is equivalent to using setContentView() for an activity).

1. In the onCreateView() method in TitleFragment.kt, create a binding variable (val binding).
2. To inflate the fragment's view, call the [DataBindingUtil.inflate()](https://developer.android.com/reference/android/databinding/DataBindingUtil" \t "_blank) method on the fragment's Binding object, which is FragmentTitleBinding.  
     
   Pass four parameters into the method:

* inflater, which is the LayoutInflater used to inflate the binding layout.
* The XML layout resource of the layout to inflate. Use one of the layouts that is already defined for you, R.layout.fragment\_title.
* container for the parent ViewGroup. (This parameter is optional.)
* false for the attachToParent value.

1. Assign the binding that DataBindingUtil.inflate returns to the binding variable.
2. Return binding.root from the method, which contains the inflated view. Your onCreateView() method now looks like the following code:

override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,

savedInstanceState: Bundle?): View? {

val binding = DataBindingUtil.inflate<FragmentTitleBinding>(inflater,

R.layout.fragment\_title,container,false)

return binding.root

}

## Step 2: Add the new fragment to the main layout file

In this step, you add the TitleFragment to the app's activity\_main.xml layout file.

1. Open **res > layout > activity\_main.xml** and click the **Text** tab to view the layout XML code.
2. Inside the existing LinearLayout element, add a fragment element.
3. Set the fragment's ID to titleFragment.
4. Set the fragment's name to the full path of the fragment class, which in this case is com.example.android.navigation.TitleFragment.
5. Set the layout width and height to match\_parent.

<layout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<fragment

android:id="@+id/titleFragment"

android:name="com.example.android.navigation.TitleFragment"

android:layout\_width="match\_parent"

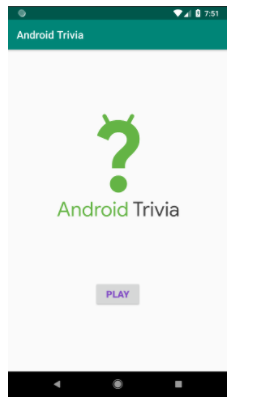
android:layout\_height="match\_parent"

/>

</LinearLayout>

</layout>

1. Run the app. The fragment has been added to your main screen.



## 5. Summary

In this lab, you added a fragment to the AndroidTrivia app, which you will keep working on next.

* A fragment is a modular section of an activity.
* A fragment has its own lifecycle and receives its own input events.
* Use the <fragment> tag to define the layout for the fragment in the XML layout file.
* Inflate the layout for a fragment in onCreateView().
* You can add or remove a fragment while the activity is running.

## 6. Learn more

Android developer documentation:

* [Linear layout](https://developer.android.com/guide/topics/ui/layout/linear)
* [LinearLayout](https://developer.android.com/reference/android/widget/LinearLayout)
* [Fragment lifecycle](https://developer.android.com/reference/android/app/Fragment#Lifecycle)
* [DataBindingUtil](https://developer.android.com/reference/android/databinding/DataBindingUtil)