Week 3 Test

Create an application which has two fragments.

Fragment 1: Create a UI to add a “car” object to a list of car objects. The UI should at at least 3 fields (Model, Type, Year).

Fragment 2: Should have a list view/recyclerView to show the list of car objects.

-Both of the fragments should be visible at all times

-Adding a fragment to the list would update the list and show the item in the fragment 2

**What are shared preferences? When do we use Shared Preferences?**

**SharedPreferences:** store primitive private data on key-value pairs. Used for store small amount of data, that may be required frequently.

**How are the values stored in shared preferences?**

save the primitive data: booleans, floats, ints, longs, and strings. These data will persist in the user session.

use one of two methods:

* getSharedPreferences (String name, int mode) - Use if you need several preferences files identified by name that will be passed in the first parameter.
* getPreferences (int mode) - Use if you just need a preferences file for your activity.

**How do you get the associated file of shared preferences?**

stored in an xml *file* in the app data folder

**Can you have multiple shared preferences file?**

Yes, we can maintain as many shared preference files for an app as you can. Just define separate classes for each of them.

**How do you write a file to internal/external storage?**

**Internal storage:**

* Call openFileOutput () with the file name and the operating mode (in case MODE\_PRIVATE). This returns a FileOutputStream;
* Write on file with the write ();
* Close the stream with close ().

**External storage:**

* Android-compatible device supports an "external memory" shared that you can use to save files.
* This may be removable storage media (such as an SD card) or an internal memory
* you should always call Environment.getExternalStorageState () to check that the media is available

**What permission(s) are needed for the above?**

**Save a file on internal storage**

Your app's internal storage directory is specified by your app's package name in a special location of the Android file system that can be accessed with the following APIs.

After you [request storage permissions](https://developer.android.com/training/data-storage/files#ExternalStoragePermissions) and [verify that storage is available](https://developer.android.com/training/data-storage/files#CheckExternalAvail), you can save two different types of files:

* [Public files](https://developer.android.com/training/data-storage/files#PublicFiles): Files that should be freely available to other apps and to the user. When the user uninstalls your app, these files should remain available to the user. For example, photos captured by your app or other downloaded files should be saved as public files.
* [Private files](https://developer.android.com/training/data-storage/files#PrivateFiles): Files that rightfully belong to your app and will be deleted when the user uninstalls your app. Although these files are technically accessible by the user and other apps because they are on the external storage, they don't provide value to the user outside of your app.

**How do you create a SQL database in Android?**

override fun onCreate(db: SQLiteDatabase) { db.execSQL(SQL\_CREATE\_ENTRIES) }

What are methods you need to implement when extend the “native database” class?

**How to do you update the schema of the database?**

onUpdate(SQLiteDatabase db,int old Version,int newVerison)

**How do you execute a query in database helper class?**

you should create a Java class that extends [SQLiteOpenHelper](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html). Here, you have to override the onCreate() and onUpdate() methods. As one can suppose, the first is called when the DB is created, while the latter is called when the DB is modified.

**Name the methods in the database helper class used for CRUD operations?**

The **SQLiteDBHelper** class is a sub class of **android.database.sqlite.SQLiteOpenHelper**

**How do you have a image in the SQL database?**

Most probably you could use BLOB in you database, convert the image to byte array and store in database.

**Write a query to create a table?**

* the name starting with sqlite\_ because these names are reserved for SQLite’s internal use.
* The database to which table belongs
* The name of each column, its [data type](http://www.sqlitetutorial.net/sqlite-data-types/), and an optional constraint. SQLite supports [PRIMARY KEY](http://www.sqlitetutorial.net/sqlite-primary-key/), [UNIQUE](http://www.sqlitetutorial.net/sqlite-unique-constraint/), [NOT NULL](http://www.sqlitetutorial.net/sqlite-not-null-constraint/), and [CHECK](http://www.sqlitetutorial.net/sqlite-check-constraint/) constraints.
* The [primary key](http://www.sqlitetutorial.net/sqlite-primary-key/) of the table: is a column or a group of columns that uniquely identifies a row in the table

**Write a query to select everything from a table?**

public Cursor queueAll() {

String[] columns = new String[] { KEY\_ID, KEY\_CONTENT1 };

Cursor cursor = sqLiteDatabase.query(MYDATABASE\_TABLE, columns, null,

null, null, null, null);

return cursor;

}

**What is a recyclerView?**

is a more advanced and more flexible version of the ListView. This new component is a big step because the ListView is one of the most used UI widgets. The CardView widget, on the other hand, is a new component that does not “upgrade” an existing component.

**What is a list view?**

**view** group that displays a **list** of scrollable items. The **list** items are automatically inserted to the **list** using an Adapter that pulls content from a source such as an array or database query and converts each item result into a **view** that's placed into the **list**

**Which is better and why?**

**a recyclerView**

How would you implement the RecyclerView.Adapter class? Write a sample class explaining what each method is responsible for.

How can you change the orientation of the recyclerView?

orientationLand = (newConfig.orientation == Configuration.ORIENTATION\_LANDSCAPE ? true : false);

**Name some layout managers you can use with the recyclerView?**

[RecyclerView](https://developer.android.com/reference/android/support/v7/widget/RecyclerView.html) provides these built-in layout managers:

* LinearLayoutManager shows items in a vertical or horizontal scrolling list.
* GridLayoutManager shows items in a grid.
* StaggeredGridLayoutManager shows items in a staggered grid.

**What is runnable? How would you run a task in the runnable?**

**Runnable** is an interface which used for creating a new thread class similar to the thread class created by extending java.lang.Thread class. Only difference is, **Runnable** interface allows the class to extend other class (if required) to override/inherit functionality of some class.

**How can you use the Thread class to run a task?**

[Thread](https://developer.android.com/reference/java/lang/Thread.html) is a basic classes that, on their own, have only limited power. Instead, they're the basis of powerful Android classes such as [HandlerThread](https://developer.android.com/reference/android/os/HandlerThread.html), [AsyncTask](https://developer.android.com/reference/android/os/AsyncTask.html), and [IntentService](https://developer.android.com/reference/android/app/IntentService.html). [Thread](https://developer.android.com/reference/java/lang/Thread.html) and [Runnable](https://developer.android.com/reference/java/lang/Runnable.html) are also the basis of the class [ThreadPoolExecutor](https://developer.android.com/reference/java/util/concurrent/ThreadPoolExecutor.html). This class automatically manages threads and task queues, and can even run multiple threads in parallel.

**How would you communicate back to the UI thread using a Handler? Explain what is a Handler class?**

A **Handler** allows **communicating back with UI thread** from other background **thread**. This is useful in android as android doesn't allow other threads to **communicate** directly **with UI thread**. A **Handler** allows you to send and process Message and Runnable objects associated **with** a **thread's** MessageQueue.

How do you associate a Handler object with the Main thread?

**Write a hander.post method?**

mHandler = new Handler();

new Thread(new Runnable() {

@Override

public void run () {

// Perform long-running task here

// (like audio buffering).

// you may want to update some progress

// bar every second, so use handler:

mHandler.post(new Runnable() {

@Override

public void run () {

// make operation on UI - on example

// on progress bar.

}

});

}

}).start();

How would you communicate back to the UI thread after waiting some time in the thread?

**What is an Asynctask? When do we use it?**  
Android defines **AsyncTask** as “a class that extends the Object class to allow short operations to run asynchronously in the background.” With “doInBackground” and “**onPostExecute**,” **Async can** run tasks asynchronously on new threads. Asynchronous tasks **use**: Params, parameters that **are** sent to the task upon execution.

Explain each type of parameter used in an Asynctask<param1, param2, param3>.

Write a sample Asynctask class, explaining each one of them.

**Which method runs on the worker thread? What is the name of that thread?**

1. Do not block the UI thread
2. Do not access the Android UI toolkit from outside the UI thread

* [Activity.runOnUiThread(Runnable)](https://developer.android.com/reference/android/app/Activity.html#runOnUiThread(java.lang.Runnable))
* [View.post(Runnable)](https://developer.android.com/reference/android/view/View.html#post(java.lang.Runnable))
* [View.postDelayed(Runnable, long)](https://developer.android.com/reference/android/view/View.html#postDelayed(java.lang.Runnable, long))

**What is EventBus?**

**EventBus** :is an open-source library for Android and Java using the **publisher/subscriber** pattern for loose coupling. EventBus enables central communication to decoupled classes with just a few lines of code – simplifying the code, removing dependencies, and speeding up app development.

**How do you setup eventbus in your project?**

**Explain step by step how would you communicate using eventbus. Write appropriate annotations and eventbus methods.**

**Step 1: Define events**

public class MessageEvent {

    public final String message;

    public MessageEvent(String message) {

        this.message = message;

    }

}

### Step 2: Prepare subscribers

// This method will be called when a MessageEvent is posted (in the UI thread for Toast)

@Subscribe(threadMode = ThreadMode.MAIN)

public void onMessageEvent(MessageEvent event) {

    Toast.makeText(getActivity(), event.message, Toast.LENGTH\_SHORT).show();

}

// This method will be called when a SomeOtherEvent is posted

@Subscribe

public void handleSomethingElse(SomeOtherEvent event) {

    doSomethingWith(event);

### }

@Override

public void onStart() {

    super.onStart();

    EventBus.getDefault().register(this);

}

@Override

public void onStop() {

    EventBus.getDefault().unregister(this);

    super.onStop();

}

### Step 3: Post events

EventBus.getDefault().post(new MessageEvent("Hello everyone!"));

**What are fragments?**

part of Android Development. They help us reuse components in Activities, ViewPagers, and Navigation drawers, among other advantages. On the other hand, their lifecycle, use cases, and interaction with Activities can be a complicated mess.

**What is the fragment lifecycle?**

**onAttach(**),**onCreate(),onCreateView(),onActivityCreated(), onStart(),onResume(), onPause(),onStop(),onDestroyView(),onDestroy()**

**How is a fragment lifecycle associated with the activity lifecycle?**

such that each **lifecycle** callback for the **activity** results in a similar callback for each **fragment**. For example, when the **activity** receives onPause() , each **fragment** in the **activity** receives onPause() .

**What are the way you can add a fragment to the activity?**

To **add a fragment** into a **Activity** or FramentActivity it requires a Container. **That** container should be a " Framelayout ", which **can** be included in xml or else **you can** use the default container for **that** like " android.R.id.content " to remove or replace a **fragment** in **Activity**

**Write the code to add a fragment to a layout in the activity?**

import android.os.Bundle;  
import android.support.v4.app.Fragment;  
import android.view.LayoutInflater;  
import android.view.ViewGroup;  
  
public class ArticleFragment extends Fragment {  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
        Bundle savedInstanceState) {  
        // Inflate the layout for this fragment  
        return inflater.inflate(R.layout.article\_view, container, false);  
    }  
}

Is a fragment added to the backstack by default? How would you add it if not?

No,

**How do you remove a fragment from the activity?**

getFragmentManager().beginTransaction().remove(getFragmentManager().findFragmentById(R.id.your\_container)).commit();

or

private static void removeAllFragments(FragmentManager fragmentManager) {

while (fragmentManager.getBackStackEntryCount() > 0) {

fragmentManager.popBackStackImmediate();

}

}

**In which method would you bind the views in a fragment? And how? Write the code to bind a textview**.

public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

View inflate = inflater.inflate(R.layout.fragment\_a, null);

View header = LayoutInflater.from(getActivity()).inflate(R.layout.header, null);

ButterKnife.bind(this, header);

mPagerMenu = (ViewPager) header.findViewById(R.id.pager\_menu);

mPagerAd = (ViewPager) header.findViewById(R.id.pager\_ad);

return inflate;

}

**How to send data to a fragment though an activity before fragment is added?**

use the Bundle

**How do you communicate with the activity from a fragment? Explain step by step.**

 using interfaces

**Can you add a fragment without the xml?**

 No, fragment is a combination of an XML layout file and a java class much like an Activity.

**What are the factory methods in fragment class used for creating a new instance?**

In this case, the newInstance() method is a “static factory method,” allowing us to initialize and setup a new Fragment without having to call its constructor and additional setter methods. Providing static factory methods for your fragments is good practice because it encapsulates and abstracts the steps required to setup the object from the client.