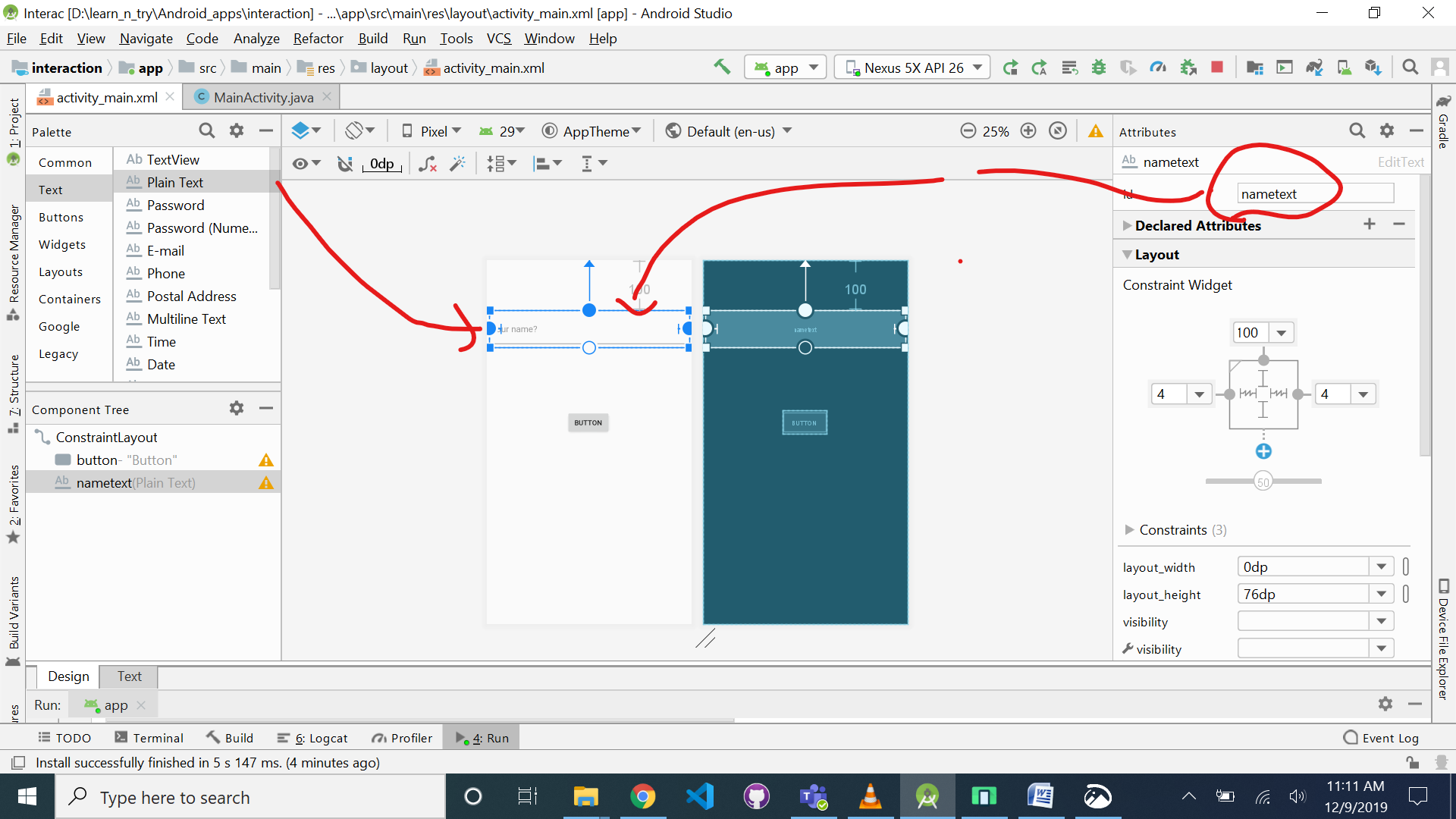
**Resources:**

1. **Udemy Course - The Complete Android Oreo Developer Course - Build 23 Apps!**
2. **Android Documentation, Forums like stack overflow for troubleshooting**
3. **Brushing up by seeing topic wise youtube videos, channels like** [**Telusko**](https://www.youtube.com/playlist?list=PLsyeobzWxl7p-lZvWabkVJdM_UVURhUh4)**.**

**My Notes:**

****

**In main activity create “click” function which is the onClick function for our button.**

**public void** click(View view)  
{  
 EditText nameEditText = (EditText) findViewById(R.id.***nametext***);  
 Log.*i*(**"Holaaa"**,**"Welcome"**);  
 Log.*i*(**"text"**, nameEditText.getText().toString());  
}

While creating 3 connect game –

To give alternate chance to players and not letting overwrite each other’s move, follow this code.

Use imageView.getDrawable()!=null

**package** com.example.connect3;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.ImageView;  
**import** android.widget.Toast;  
  
**public class** MainActivity **extends** AppCompatActivity {  
 **boolean re**=**true**;  
 **public void** dropIn(View view){  
 ImageView counter = (ImageView)view;  
 **if**(counter.getDrawable()==**null**) {  
 counter.setTranslationY(-1500);  
 **if** (**re**) {  
 counter.setImageResource(R.drawable.***red***);  
 **re** = **false**;  
 } **else** {  
 counter.setImageResource(R.drawable.***yellow***);  
 **re** = **true**;  
 }  
 counter.animate().translationYBy(1500).setDuration(1000);  
 }  
 }  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 }  
}

If resource name is required follow this link -<https://stackoverflow.com/questions/23357706/how-to-check-which-current-image-resource-is-attached-to-imageview-in-android-xm>

**To import a video, first make a folder “raw” inside res and paste video in it. Then create a VideoView and write this code in OnCreate method :-**

VideoView vid = (VideoView ) findViewById(R.id.videoView);

vid.setVideoPath(“android.resource://”+getPackageName()+”/”+R.raw.demovideo);

*//ADD MEDIA CONTROLLER TO ADD PAUSE PLAY SEEK OPTION:*

MediaController med = new MediaController(this);

med.setAnchorView(vid);

vid.setMediaController(med);

vid.start();

**TO run an Audio, we don’t have to add any view, just adding media controller will do.**

MediaPlayer med = MediaPlayer.create(context:this,R.raw.demoaudio);

med.start();

//to add controls, u have to add buttons on your own and add this function to play button function…

//for seekbar, chk out audio seeking video mod -4 vid -14

//If filename is stored in some variable, or saved as tag of the button being tapped, we can use this code snippet :

**public void** play(View view){  
 Button btn= (Button) view;  
 Log.*i*(**"Button pressed"**,btn.getTag().toString());  
 MediaPlayer m = MediaPlayer.*create*(**this**, getResources().getIdentifier(btn.getTag().toString(), **"raw"**, getPackageName()));  
 m.start();  
}

in this code, we had different buttons which were supposed to play different audio on being tapped.

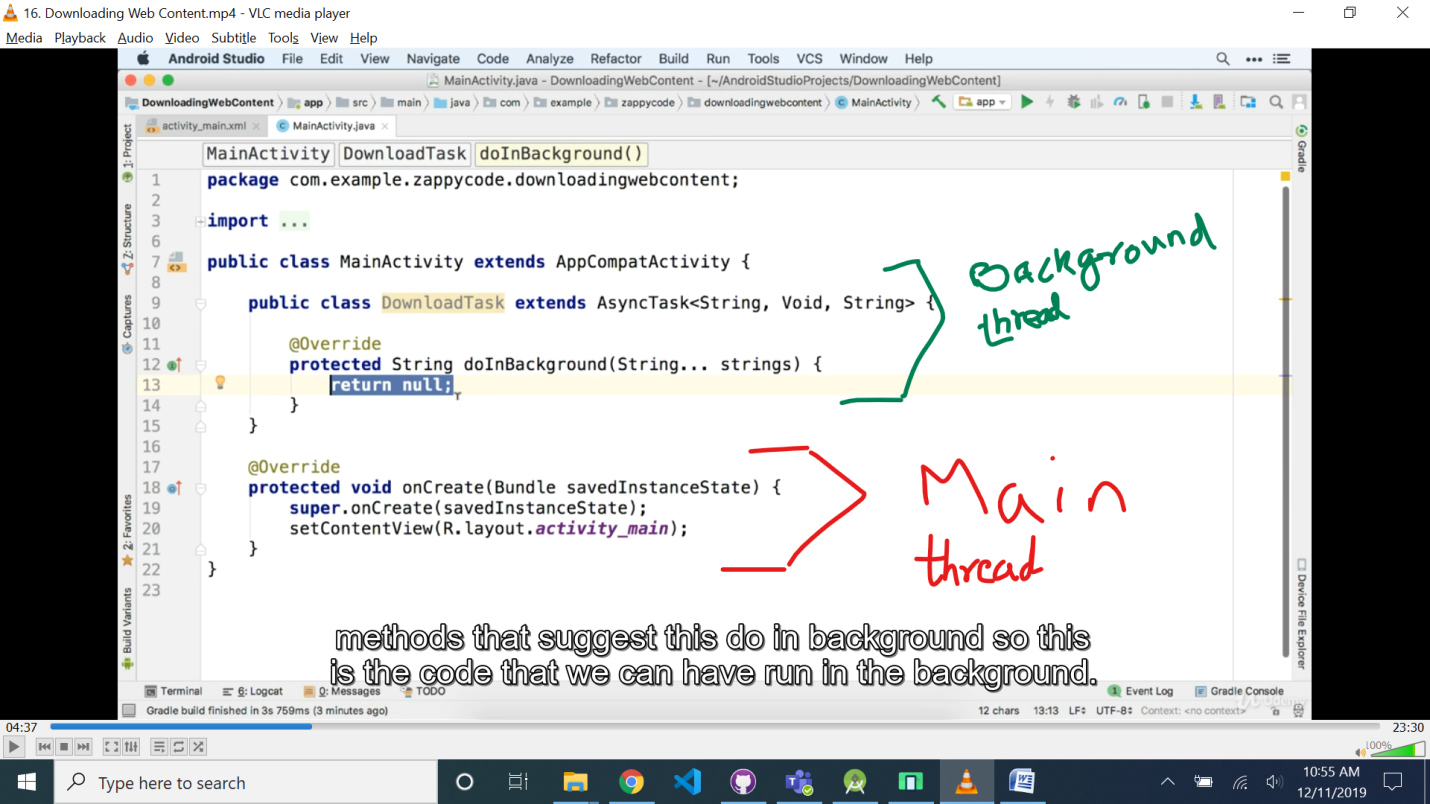
All the buttons had the tag set as their corresponding audio filenames. And to avoid writing separate functions for each, we used a generic func play which was called whenever a button was tapped.

The view shared by the button helped in playing the respective audio file.

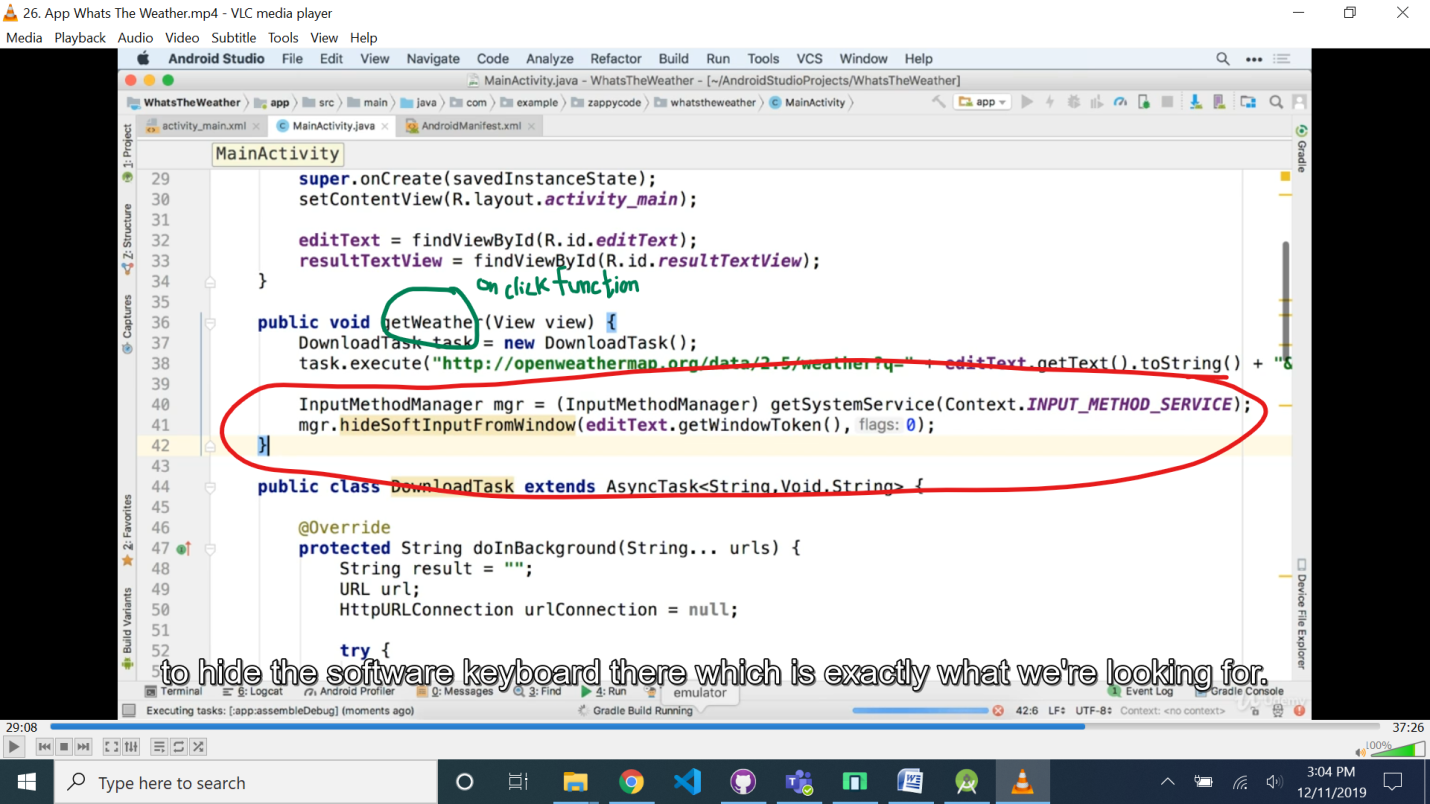
**The processes run on threads, there is this main thread where we usually put in all the functions.**

But it is often advised to make use of other threads to run things simultaneously in the background as well, background thread comes real handy when downloading data.

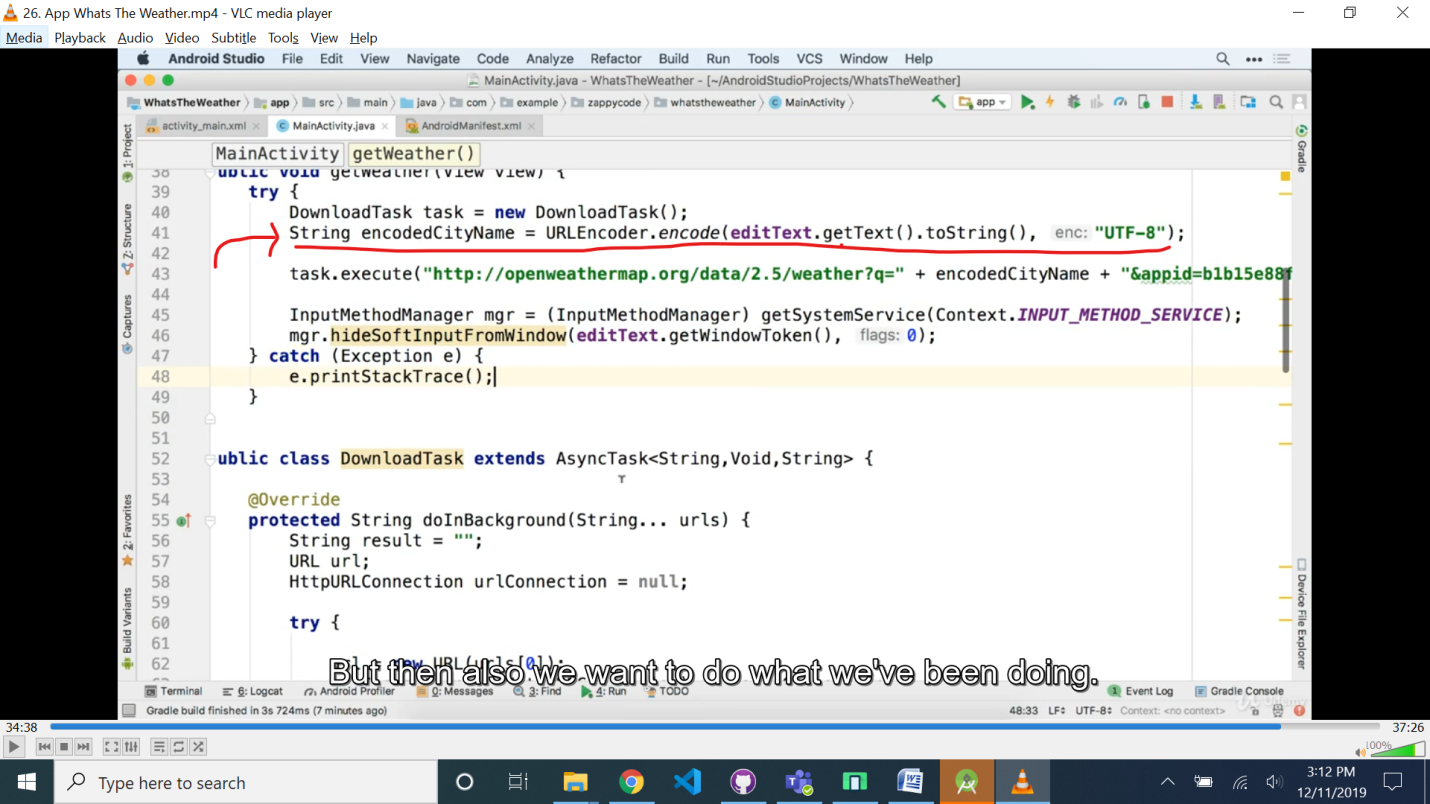
Use AsyncTask



To hide keyboard after clicking on a button:



Usually, spaces inside urls are converted into “%20” in android, but if doesn’t, include this code:



**Storing Data**

Storing data locally inside app uses shared preferences :

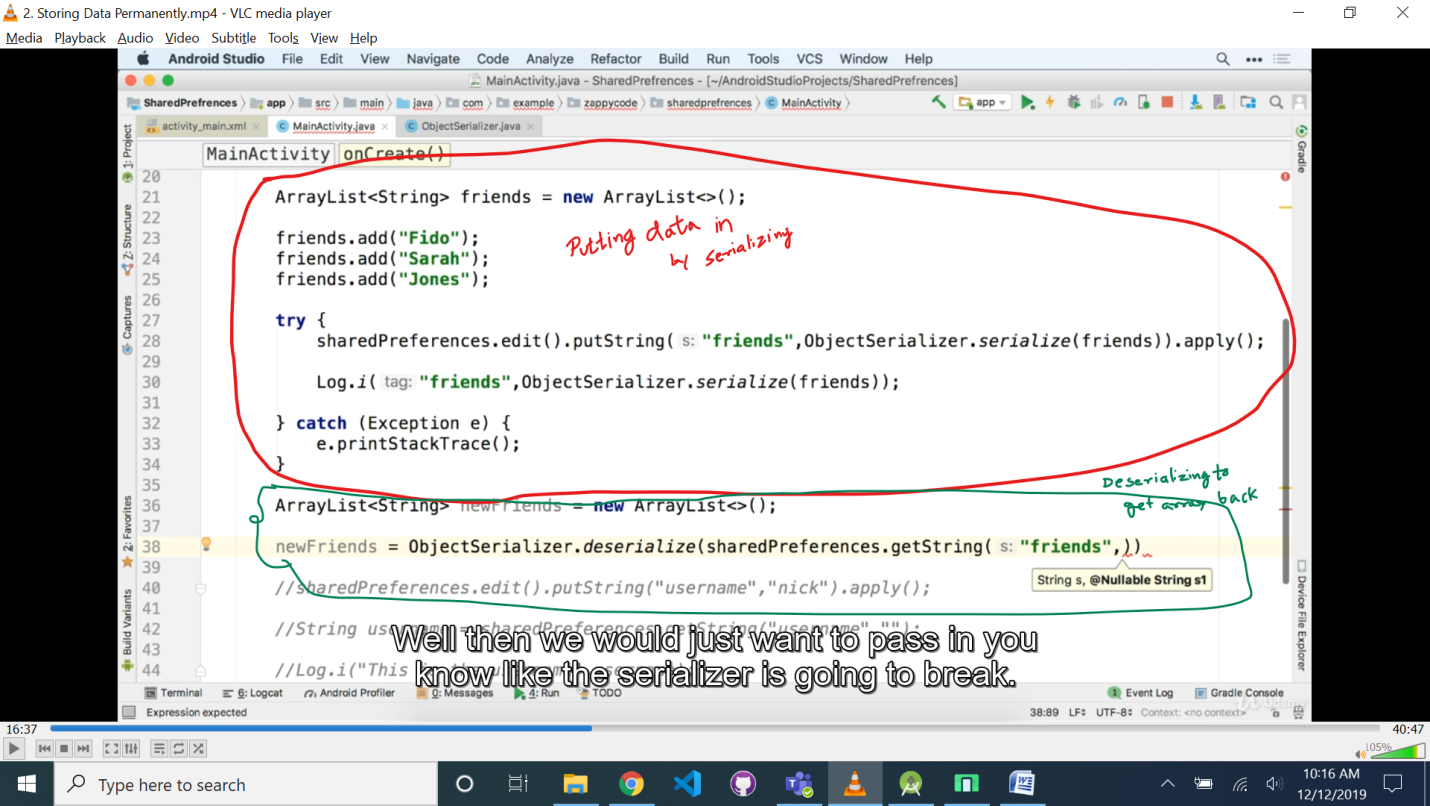
**package** com.example.sharedpreferences;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.content.Context;  
**import** android.content.SharedPreferences;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**public class** MainActivity **extends** AppCompatActivity {  
 SharedPreferences **sp**;  
  
 **public void** onclick(View view){  
 EditText e = (EditText)findViewById(R.id.***name***);  
 **sp**.edit().putString(**"username"**,e.getText().toString()).apply();  
  
 Toast.*makeText*(**this**, **sp**.getString(**"username"**,**"user not found"**), Toast.***LENGTH\_SHORT***).show();  
 }  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 **sp** = **this**.getSharedPreferences(**"com.example.sharedpreferences"**, Context.***MODE\_PRIVATE***);  
  
 Toast.*makeText*(**this**, **sp**.getString(**"username"**,**"user not found"**), Toast.***LENGTH\_SHORT***).show();  
 }  
}

***We can use shared preferences to store strings, float, int and other primitive data types.***

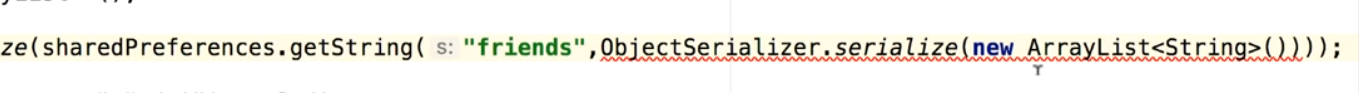
***Although we can directly store array in string set, the contents wouldn’t be saved in order…***

In order to store array items in order, we can serialize the array into a single string (by making a new class ObjectSerializer (copy the code from – [www.androiddevcourse.com/objectserializer.html](http://www.androiddevcourse.com/objectserializer.html) , replace everything except the package name))

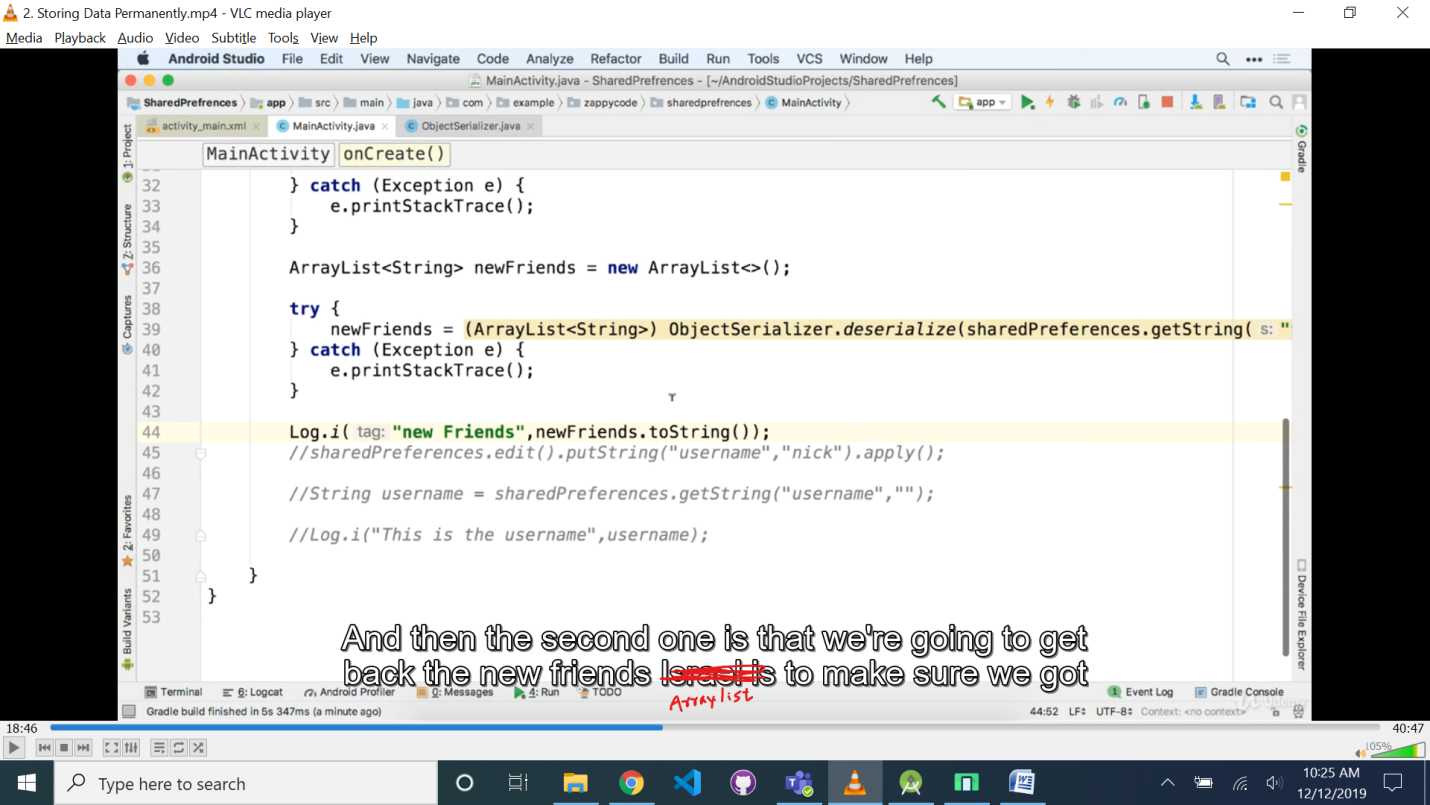
Entering data in Shared preference :



Put the next arg as this, so that if data is not retrieved it will return empty array(we cant put empty string coz the container/variable is an array not string)



Bigger picture:



Or my idea:

Merge contents of array into a single string by having some symbol in between each item and then splitting it back when retrieving it.

**Multiple Screens / Activities**

Refer Example app.

Main\_Activity :

**package** com.example.exampleapp;  
**import** androidx.appcompat.app.AppCompatActivity;  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.widget.EditText;  
  
**public class** MainActivity **extends** AppCompatActivity {  
 **public void** callfun(View view){  
 Log.*i*(**"info"**,**"Button pressed!!"**);  
 EditText usrname = (EditText)findViewById(R.id.***editText***);  
 Intent intent = **new** Intent(MainActivity.**this**,loggedin.**class**);  
 Log.*i*(**"info"**,usrname.getText().toString());  
 intent.putExtra(**"nme"**,usrname.getText().toString());  
 startActivity(intent);  
  
 }  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 }  
}

Create new Empty Activity named loggedin

**package** com.example.exampleapp;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.widget.TextView;  
  
**public class** loggedin **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_loggedin***);  
  
 Intent intent =getIntent();  
 Log.*i*(**"inten"**,intent.getStringExtra(**"nme"**));  
 String gname = intent.getStringExtra(**"nme"**);  
 TextView t = findViewById(R.id.***wel***);  
 t.setText(**"Welcome "** +gname);  
 }  
}

**To save user info, either use intent.putExtra or use shared preferences.**

**Layout:**

If you're using a [LinearLayout](https://developer.android.com/reference/android/widget/LinearLayout.html), you can also expand the child views with [layout weight](https://developer.android.com/guide/topics/ui/layout/linear.html#Weight) so that each view fills the remaining space proportional to their weight value. However, using weights in a nested LinearLayout requires the system to perform multiple layout passes to determine the size for each view, slowing your UI performance. Fortunately, ConstraintLayout can achieve nearly all layouts possible with LinearLayout without the performance impacts, so you should [try converting your layout to ConstraintLayout](https://developer.android.com/studio/write/layout-editor.html#convert-%0Aconstraint). Then, you can [define weighted layouts with constraint chains](https://developer.android.com/training/constraint-layout/index.html#constrain-chain).

**Note:** When using **[ConstraintLayout](https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout.html)**, you **should not use match\_parent**. Instead, set the dimension to **0dp** to enable a special behavior called "match constraints," which is generally the same as what you expect from **match\_parent**. For more information, see how to [adjust the view size in ConstraintLayout](https://developer.android.com/training/constraint-layout/index.html#adjust-the-view-size).

**Responsive :**

1. Using match\_constraints and Wrap\_Content in width
2. Use layout\_weight
3. Use Constraint layout and its widget to get flexible layout.