**Assignment-6**

**1.Difference Between Relative And Constraint Layout**

**Relative Layout:**

A [RelativeLayout](https://developer.android.com/reference/android/widget/RelativeLayout.html) is a very powerful utility for designing a user interface because it can eliminate nested view groups and keep your layout hierarchy flat, which improves performance. If you find yourself using several nested [LinearLayout](https://developer.android.com/reference/android/widget/LinearLayout.html) groups, you may be able to replace them with a single [RelativeLayout](https://developer.android.com/reference/android/widget/RelativeLayout.html).RelativeLayout is awesome at situating things in respect to other. Suppose you have undetermined view width and you need to put a thing to one side of it and to the base of it in the meantime.

[RelativeLayout](https://developer.android.com/reference/android/widget/RelativeLayout.html) is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements (such as to the left-of or below another view) or in positions relative to the parent [RelativeLayout](https://developer.android.com/reference/android/widget/RelativeLayout.html) area (such as aligned to the bottom, left or center).

**Constraint Layout:**

The center is RelativeLayout where things can overlay. Be that as it may, what makes ConstraintLayout awesome is it can supplant the various designs: Frame, Relative, Linear. Truly, there are numerous different formats, yet those are wrappers What I like about

ConstraintLayout is that it's extremely awesome at utilizing as a part of Design mode before I didn't care for it by any means. For the most part in view of the poor execution of my fair equipment (for Android Studio) and regular rendering bugs ConstraintLayout properties really are so long and appalling that it's better utilizing it Design mode.

It can control position and measurements (not at all like RelativeLayout) of perspectives. Keep in mind, that parameters beginning with layout\_ are implied for wrapping format. The view itself doesn't deal with those .is similar to a RelativeLayout in that it uses relations to position and size widgets, but has additional flexibility and is easier to use in the Layout Editor.

So you can utilize your most loved weight from LinearLayout in ConstraintLayout by imperative chains, land left/right/top/base of from RelativeLayout, overlay things like in straightforward FrameLayout I think no one uses GridLayout, it's far excessively disappointing, constantly favored settled LinearLayout, with ConstraintLayout you can do that effortlessly.

**2.Why we use dp instead of px in android studio**

**dp or dip**

Density-independent Pixels-an abstract unit that is based on the physical density of the screen.These units are relative to a 160 dpi screen, so one dp is one pixel on a 160 dpi screen. The ratio of dp-to-pixel will change with the screen density,but not necessarily in direct proportion. Note: The compiler accepts both “dip”and “dp”,though “dp” is more consistent with “sp”.

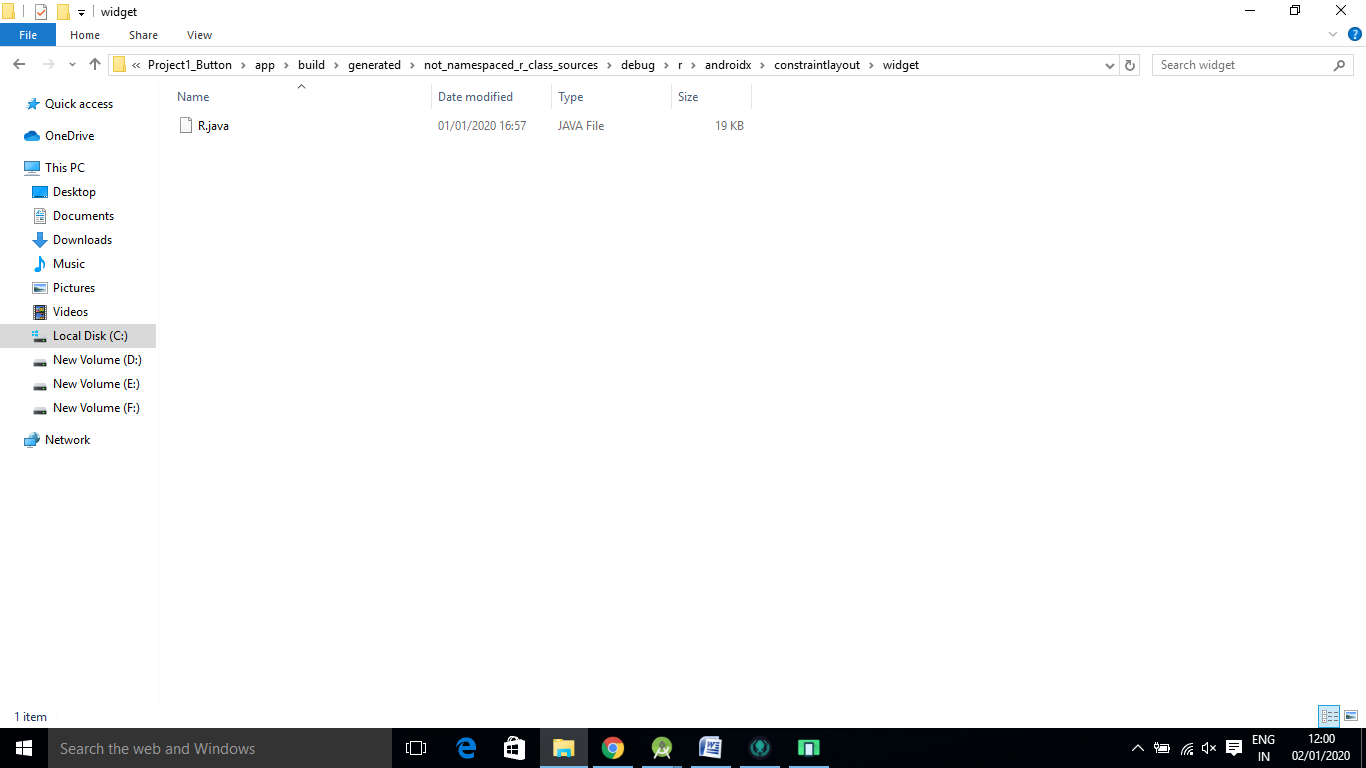
**Density-independent pixel (dp)**

A virtual pixel unit that you should use when defining UI layout,to express layout dimensions or position in a density-independent way. The density-independent pixel is equivalent to one physical pixel on a 160 dpi screen,which is the baseline density assumed by the system for a medium density screen.At runtime the system transparently handles any scaling of the dp units, as necessary, based on the actual density of the screen in use.The conversion of dp units to screen pixels is simple:px=dp\*(dpi/160).

For example on a 240 dpi screen, 1 dp equals 1.5 physical pixels

**3.Where is R.java in android**

C:\Users\unnati\AndroidStudioProjects\Project1\_Button\app\build\generated\not\_namespaced\_r\_class\_sources\debug\r\androidx\constraintlayout\widget

****

**References**

stackoverflow.com

developer.android.com