**Yashashwi Patel - Assignment# Android UI**

**Q) Cite 5 methods of the Activity class**

**onCreate()** - Called when the activity is first created. This is where you should do all of your normal static set up: create views, bind data to lists, etc. This method also provides you with a Bundle containing the activity's previously frozen state, if there was one. Always followed by onStart().

**OnRestart()** - Called after your activity has been stopped, prior to it being started again. Always followed by onStart().

**onStart()** - Called when the activity is becoming visible to the user. Followed by onResume() if the activity comes to the foreground, or onStop() if it becomes hidden.

**onResume()** - Called when the activity will start interacting with the user. At this point your activity is at the top of the activity stack, with user input going to it. Always followed by onPause().

**onPause()** - Called when the system is about to start resuming a previous activity. This is typically used to commit unsaved changes to persistent data, stop animations and other things that may be consuming CPU, etc. Implementations of this method must be very quick because the next activity will not be resumed until this method returns. Followed by either onResume() if the activity returns back to the front, or onStop() if it becomes invisible to the user.

**onStop() –** Called when the activity is no longer visible to the user, because another activity has been resumed and is covering this one. This may happen either because a new activity has been started, an existing one has been brought in front of this one or this one is being destroyed. Followed by either onRestart() if this activity is coming back to interact with the user or onDestroy() if this activity is going away.

**Q.) What methods relate to the menu in the Activity class?**

**void closeContextMenu()**

Programmatically closes the most recently opened context menu, if showing.

**void closeOptionsMenu()**

Progammatically closes the options menu.

**void invalidateOptionsMenu()**

Declare that the options menu has changed, so should be recreated.

**void onContextMenuClosed(Menu menu)**

This hook is called whenever the context menu is being closed (either by the user canceling the menu with the back/menu button, or when an item is selected).

**void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo)**

Called when a context menu for the view is about to be shown.

**void onOptionsMenuClosed(Menu menu)**

This hook is called whenever the options menu is being closed (either by the user canceling the menu with the back/menu button, or when an item is selected).

**boolean onPrepareOptionsMenu(Menu menu)**

Prepare the Screen's standard options menu to be displayed.

**void openContextMenu(View view)**

Programmatically opens the context menu for a particular view.

**void openOptionsMenu()**

Programmatically opens the options menu.

**void unregisterForContextMenu(View view)**

Prevents a context menu to be shown for the given view.

**Q.) How do the classes Activity and ActionBarActivity relate to each others?**

**Class Activity** -- An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with setContentView(View). While activities are often presented to the user as full-screen windows, they can also be used in other ways: as floating windows (via a theme with windowIsFloating set) or embedded inside of another activity (using ActivityGroup). There are two methods almost all subclasses of Activity will implement:

onCreate(Bundle) is where you initialize your activity. Most importantly, here you will usually call setContentView(int) with a layout resource defining your UI, and using findViewById(int) to retrieve the widgets in that UI that you need to interact with programmatically.

onPause() is where you deal with the user leaving your activity. Most importantly, any changes made by the user should at this point be committed (usually to the ContentProvider holding the data).

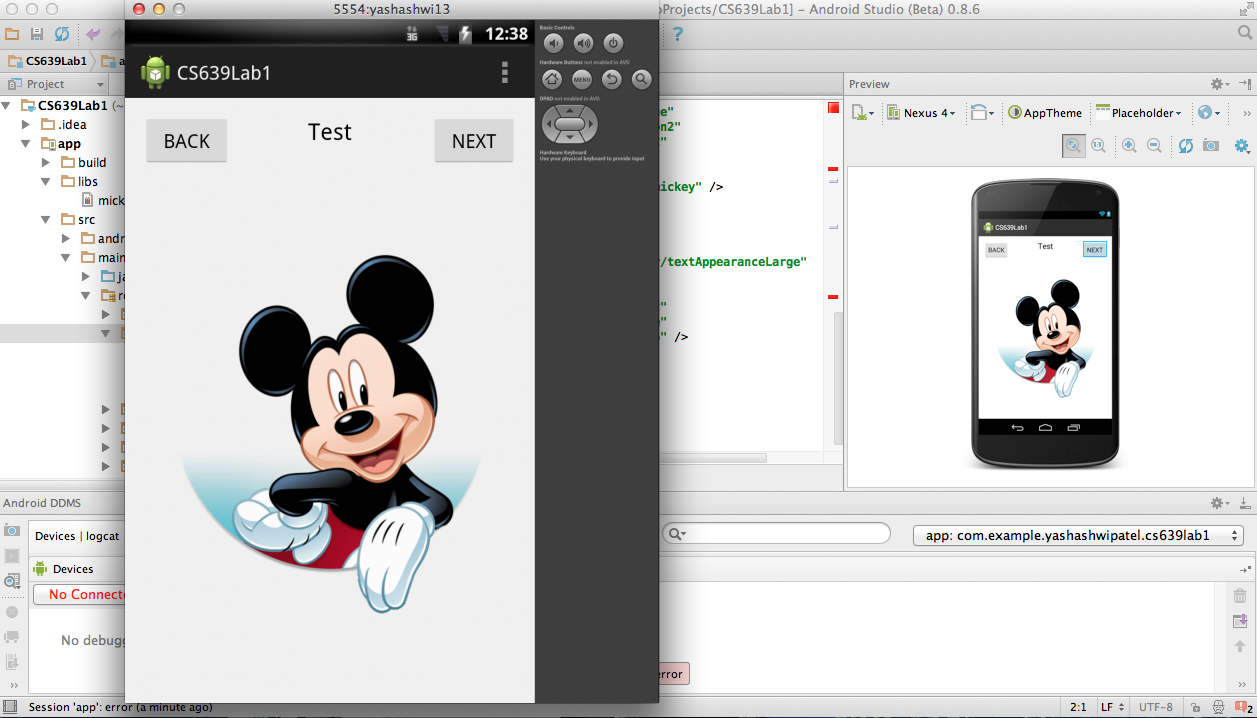
To be of use with Context.startActivity(), all activity classes must have a corresponding <activity> declaration.

**ActionBarActivity** just has more support libraries and better usage of the newer themes available from api 11. In its most basic form, the action bar displays the title for the activity and the app icon on the left. Even in this simple form, the action bar is useful for all activities to inform users about where they are and to maintain a consistent identity for your app.

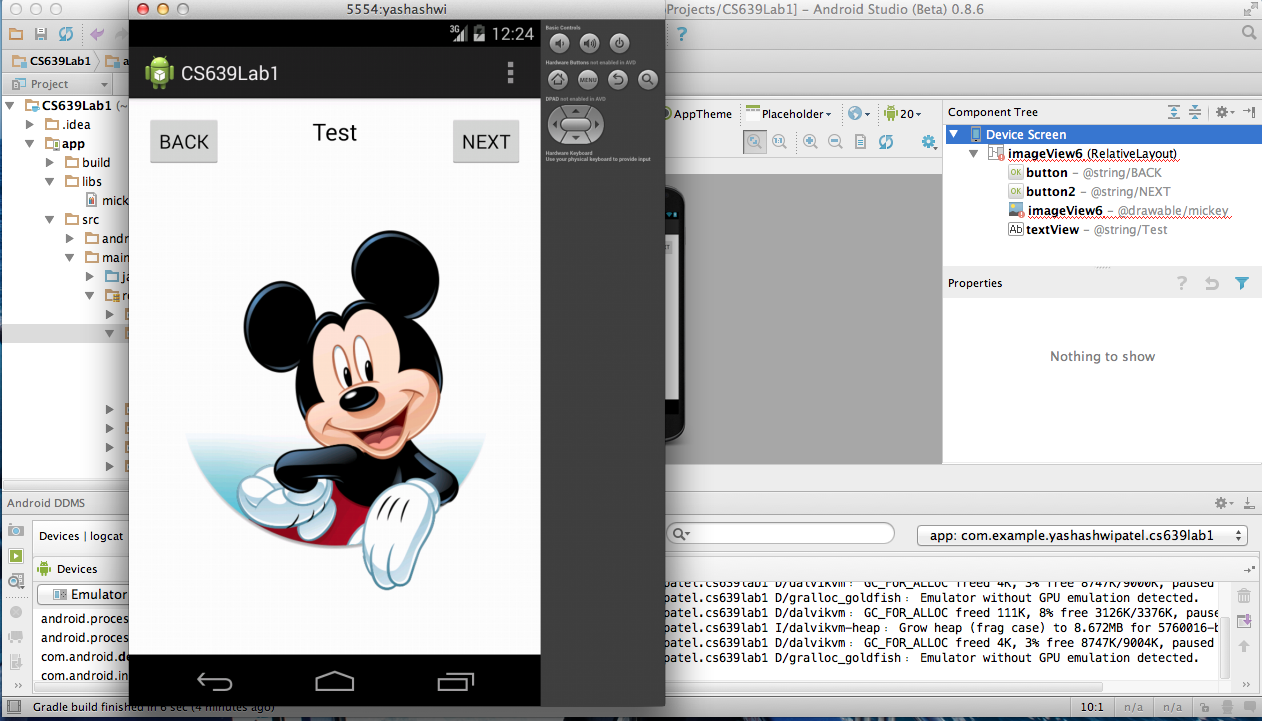
Actionbar is introduced in API level 11. com.android.support:appcompat-v7:+ is a support library which allows you to have an ActionBar in your app for devices running on Android 3.0 or below. So, if you need actionbar below api level 11 your Activity needs to extend ActionBarActivity. If you are targeting api level 11 and above then you don't need to extend ActionBarActivity and reference AppCompat. You can simply extend Activity and you will have actionbar by default. Android Studio default project includes it automatically in dependencies and extends ActionbarActivity instead of Activity in order to use it.

**# ScreenShot of Lab2:**

**3.2 AVD**

****

**4.3 AVD**

****