|  |  |  |
| --- | --- | --- |
| **Mod-3: Activity(26-31)** | **Start: 1:30 pm** | **Stop: 3:00 pm** |
| * The Loader API lets you load data from a content provider or other data source for display in an FragmentActivity or Fragment. * Fetching the data directly in the activity or fragment, from another thread will leads to loose responsiveness. * Loaders solve these problems and includes other benefits. * Loaders run on separate threads to prevent janky or unresponsive UI. * Loaders simplify thread management by providing callback methods when events occur. * The LoaderManager manages one or more Loader instances within an FragmentActivity or Fragment. There is only one LoaderManager per activity or fragment. * When you use initLoader(), t uses an existing loader with the specified ID if there is one. If there isn't, it creates one. But sometimes you want to discard your old data and start over. | | |

|  |  |  |
| --- | --- | --- |
| **Activity(32-40)** | **Start: 3:20 pm** | **Stop: 6:00 pm** |
| * The basic Preference UI building block displayed by a PreferenceActivity in the form of a ListView. * This class provides the View to be displayed in the activity and associates with a SharedPreferences to store/retrieve the preference data. * When specifying a preference hierarchy in XML, each element can point to a subclass of Preference, similar to the view hierarchy and layouts. * The Visualizer class enables application to retrieve part of the currently playing audio for visualization purpose. * It is not an audio recording interface and only returns partial and low quality audio content. However, to protect privacy of certain audio data, the use of the visualizer requires the permission android.permission. * Android provides several options for you to save your app data. * The preference hierarchy can be formed in multiple ways:   + From an XML file specifying the hierarchy   + From different Activity that each specify its own preferences in an XML file via Activity meta-data. * If you are developing apps for the enterprise market, you may need to satisfy particular requirements set by a organization's policies. * Managed configurations, previously known as application restrictions, allow the organization's IT admin to remotely specify settings for apps. * This capability is particularly useful for organization-approved apps deployed to a work profile. | | |

|  |  |  |
| --- | --- | --- |
| **Activity(41-47)** | **Start: 6:30 pm** | **Stop: 9:00 pm** |
| * Interface for accessing and modifying preference data returned by Context#getSharedPreferences. * For any particular set of preferences, there is a single instance of this class that all clients share. * Modifications to the preferences must go through an Editor object to ensure the preference values remain in a consistent state and control when they are committed to storage. * Objects that are returned from the various get methods must be treated as immutable by the application. * Resources are the additional files and static content that your code uses, such as bitmaps, layout definitions, user interface strings, animation instructions, and more. | | |

|  |
| --- |
| **Deliverable Status** |