Intents and Activities

Activity:

An **`Activity`** is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map.

Each activity is given a window in which to draw its user interface. The window typically fills the screen, but may be smaller than the screen and float on top of other windows. The floating window can be considered as dialog screen pop up which are also a kind of Activity.

An application usually consists of multiple activities that are loosely bound to each other. One activity in an application is specified as the "main" activity, which is presented to the user when launching the application for the first time.

Each activity can then start another activity in order to perform different actions. Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack (the "back stack"). When a new activity starts, it is pushed onto the back stack and takes user focus. The back stack abides to the basic "last in, first out" stack mechanism, so, when the user is done with the current activity and presses the *Back* button, it is popped from the stack (and destroyed) and the previous activity resumes.

Creating an Activity

To create an activity, we need to create a subclass of Activity. After this we need to register the Activity in the Manifest file. There should be method implementation in class for OnCreate.

```
public class Auditdataview extends Activity {
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Implementing a user interface

The user interface for an activity is provided by a hierarchy of views—objects derived from the `View` class. Each view controls a particular rectangular space within the activity's window and can respond to user interaction. For example, a view might be a button that initiates an action when the user touches it.

Android provides a number of ready-made views that can be used to design and organize layouts. "Widgets"- also called as controls, are views that provide a visual (and interactive) elements for the screen, such as a button, text field, checkbox, or just an image.

"Layouts" are views derived from `ViewGroup` that provide a unique layout model for its child views, such as a linear layout, a grid layout, or relative layout.

The most common way to define a layout using views is with an XML layout file saved in your application resources. This way, the design of the user interface can be maintained separately from the source code that defines the activity's behavior.

`setContentView()` method can be used to bind layout and Activity in the application.

Resource taken from: http://developer.android.com/guide/components/activities.html

Intents:

An `Intent` object is a bundle of information. It contains information about the component with which it is bound, such as the action to be taken and the data to be passed between components.

Three of the core components of an application — activities, services, and broadcast receivers — are activated through messages, called *intents*. Intent messaging is a facility for late run-time binding between components in the same or different applications.

Intents are instances of the android.content.Intent class.

Working of Intents:

Intents are asynchronous messages which allow Android components to request functionality from other components of the Android system. For example an Activity can send an *Intents* to the Android system which starts another Activity.

Therefore *Intents* allow to combine loosely coupled components to perform certain tasks.

Intents can be used to signal to the Android system that a certain event has occurred. Other components in Android can register to this event and will get notified.

Types of Intents:

Intents in Android are of two types: 1) Explicit Intents and 2) Implicit Intents

Explicit Intents:

- Explicit Intents *defines the component which should be called by the Android system, by using the Java class as identifier.*

  The following shows an Explicit Intent. If that Intent is correctly send to the Android system and the class is accessible, it will start the associated class.

```
Intent i = new Intent(this, ActivityTwo.class);
i.putExtra("Value1", "This value one for ActivityTwo ");
i.putExtra("Value2", "This value two ActivityTwo");
```

Implicit Intents:

- Implicit *Intents* do not directly specify the Android components which should be called. They specify the action which should be performed and optionally an URI which should be used for this action.
- If these *Intents* are send to the Android system it searches for all components which are registered for the specific action and the data type.
- If only one component is found, Android starts this component directly. If several components are identifier by the Android system, the user will get an selection dialog and can decide which component should be used for the Intent.

```
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.vogella.com"));
```

Code to send an image as Email Attachment:

```
final Intent emailIntent = new Intent(android.content.Intent.ACTION_SEND);
emailIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
emailIntent.putExtra(Intent.EXTRA_STREAM, screenshotUri);
emailIntent.setType("image/png");

startActivity(Intent.createChooser(emailIntent, "Send email using"));
```

Code to browse an image from Photo Gallery:

```
Intent intentBrowseFiles = new Intent(Intent.ACTION_GET_CONTENT);
intentBrowseFiles.setType("image/*");
intentBrowseFiles.setFlag(Intent.FLAG_ACTIVITY_NEW_TASK);
startActivity(intentBrowseFiles);
```

Code to dial a number programmatically:

```
startActivity(new Intent(Intent.ACTION_CALL, Uri.parse("tel:" + mNumber)));
```