Menus

Creating Android Menu

- Android offers three fundamental types of application menus:
  - **Options Menu**
  - **Context Menu**
  - **Submenu**

Option Menu

- This is the primary set of menu items for an Activity.
- It is revealed by pressing the device MENU key.
- Within the Options Menu are two groups of menu items:
  - ***Icon Menu***
    - This is the collection of items initially visible at the bottom of the screen at the press of the MENU key.
    - It supports a maximum of six menu items.
    - These are the only menu items that support icons and the only menu items that *do not* support checkboxes or radio buttons.
  - ***Expanded Menu***
    - This is a vertical list of items exposed by the "More" menu item from the Icon Menu.
    - It exists only when the Icon Menu becomes over-loaded and is comprised of the sixth Option Menu item and the rest.

Option Menu

- The Options Menu is where you should include basic application functions and any necessary navigation items.
- You can also add Submenus for organizing topics and including extra menu functionality.
- When this menu is opened for the first time, the Android system will call the Activity onCreateOptionsMenu() callback method.
- Override this method in your Activity and populate the Menu object given to you.
- You can populate the menu by inflating a menu resource that was defined in XML, or by calling add() for each item.
- This method adds a MenuItem, and returns the newly created object to you.

Option Menu

- When a menu item is selected from the Options Menu, you will receive a callback to the onOptionsItemSelected() method of your Activity.
- This callback passes you the MenuItem that has been selected.
- You can identify the item by requesting the *itemId*, with getItemId().

Option Menu - Example

```
/* Creates the menu items */
    public boolean onCreateOptionsMenu(Menu menu) {
      menu.add(0, MENU_NEW_GAME, 0, "New Game");
      menu.add(0, MENU_QUIT, 0, "Quit");
      return true;
    }

    /* Handles item selections */
    public boolean onOptionsItemSelected(MenuItem item) {
      switch (item.getItemId()) {
      case MENU_NEW_GAME:
        newGame();
        return true;
      case MENU_QUIT:
        quit();
        return true;
      }
      return false;
    }
```

The add() method takes four arguments: *groupId*, *itemId*, *order*, and *title*.

Option Menu

- Adding icons
  - Icons can also be added to items that appears in the Icon Menu with setIcon().
    For example:

```
menu.add(o, MENU_QUIT, o, "Quit").setIcon(R.drawable.menu_quit_icon);
```

- Modifying the menu
  - If you want to sometimes re-write the Options Menu as it is opened, override the onPrepareOptionsMenu() method, which is called each time the menu is opened.
  - This will pass you the Menu object, just like the onCreateOptionsMenu() callback.

Option Menu

Refer the example OptionMenuDemo

Context Menu

- The Android context menu is similar, in concept, to the menu revealed with a "right-click" on a PC.
- When a view is registered to a context menu, performing a "long-press“ on the object will reveal a floating menu that provides functions relating to that item.
- Context menus can be registered to any View object, however, they are most often used for items in a ListView.
- To create a context menu, you must override the Activity's context menu callback methods: onCreateContextMenu() and onContextItemSelected().
- Register a ContextMenu for the View, with registerForContextMenu().

Context Menu - Example

```
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    menu.add(0, EDIT_ID, 0, "Edit");
    menu.add(0, DELETE_ID, 0,  "Delete");
}
    public boolean onContextItemSelected(MenuItem item) {
     AdapterContextMenuInfo info = (AdapterContextMenuInfo)
    item.getMenuInfo();
     switch (item.getItemId()) {
     case EDIT_ID:
       editNote(info.id);    return true;
     case DELETE_ID:
       deleteNote(info.id);     return true;
     default:
       return super.onContextItemSelected(item);
     }
    }
```

Context Menu Example

Refer the Examples:

- *ContextMenuDemo*
- *ContextMenuWithListView*

Submenus

- A sub menu can be added within any menu, except another sub menu.
- A sub menu is created by adding it to an existing Menu with addSubMenu().
- This returns a SubMenu object. You can then add additional items to this menu, with the normal routine, using the add() methods.

Submenus - Example

```
public boolean onCreateOptionsMenu(Menu menu) {
    boolean result = super.onCreateOptionsMenu(menu);

    SubMenu fileMenu = menu.addSubMenu("File");
    SubMenu editMenu = menu.addSubMenu("Edit");
    fileMenu.add("new");
    fileMenu.add("open");
    fileMenu.add("save");
    editMenu.add("undo");
    editMenu.add("redo");

    return result;
}
```

Define Menus in XML

- You can define application menus in XML, then inflate them in your menu's onCreate...() callback method.
- To start, create a new folder in your project res/ directory called menu. This is where you should keep all XML files that define your application menus.
- In a menu XML layout, there are three valid elements: <menu>, <group> and <item>.
- The item and group elements must be children of a menu, but item elements may also be the children of a group, and another menu element may be the child of an item.

Define Menus in XML

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
        <item android:id="@+id/new_game"
            android:title="New Game" />
        <item android:id="@+id/quit"
            android:title="Quit" />
    </menu>
```

The getMenuInflater() method returns the MenuInflater for our activity's context. We then call inflate(), passing it a pointer to our menu resource and the Menu object given by the callback.

```
public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.options_menu, menu);
        return true;
    }
```

Menu Features

Menu Group

- A menu group is a collection of menu items that can share certain traits, like whether they are visible, enabled, or checkable.
- A group is defined by an integer (or a resource id, in XML). A menu item is added to the group when it is added to the menu, using one of the add() methods that accepts a *groupId* as an argument, such as add(int, int, int, int).
- You can show or hide the entire group with setGroupVisible(); enable or disable the group with setGroupEnabled(); and set whether the items can be checkable with setGroupCheckable().

Menu Features

- **Shortcut keys**
  - Quick access shortcut keys using letters and/or numbers can be added to menu items with setAlphabeticShortcut(char) (to set char shortcut), setNumericShortcut(int) (to set numeric shortcut), or setShortcut(char,int) (to set both).
  - Case is *not* sensitive.

Note: Shortcuts cannot be added to items in a Context Menu.

Menu Item Intents

- you can perform such actions from within a menu.
- There are two ways to do this: define an Intent and assign it to a single menu item, or define an Intent and allow Android to search the device for activities and dynamically add a menu item for each one that meets the Intent criteria.

```
MenuItem menuItem = menu.add(0, PHOTO_PICKER_ID, 0, "Select Photo");
    menuItem.setIntent(new Intent(this, PhotoPicker.class));
```