Menus and Dialogs

CS 240 – Advanced Programming Concepts

MENUS

Adding a Menu to an Activity

1. Add menu strings to res/values/strings.xml resource file(s)
2. Add a menu layout file (New -> Android Resource File)
   - Select 'menu' as the resource type
   - Add item attributes
   - Example: ActivityMenuExample/res/menu/main_menu.xml
3. Override onCreateOptionsMenu(Menu)
4. Override onOptionsItemSelected(MenuItem)

- Example: ActivityMenuExample.zip

Menu Item Flags

Value	Description
`ifRoom`	Only place this item in the app bar if there is room for it. If there is not room for all the items marked "**`ifRoom`**", the items with the lowest **`orderInCategory`** values are displayed as actions, and the remaining items are displayed in the overflow menu.
`withText`	Also include the title text (defined by **`android:title`**) with the action item. You can include this value along with one of the others as a flag set, by separating them with a pipe |.
`never`	Never place this item in the app bar. Instead, list the item in the app bar's overflow menu.
`always`	Always place this item in the app bar. Avoid using this unless it's critical that the item always appear in the action bar. Setting multiple items to always appear as action items can result in them overlapping with other UI in the app bar.
`collapseActionView`	The action view associated with this action item (as declared by **`android:actionLayout`** or **`android:actionViewClass`**) is collapsible. Introduced in API Level 14.

https://developer.android.com/guide/topics/resources/menu-resource

Adding an Icon to a Menu

- From the layout file
  - **android:icon="@drawable/ic_file”**
  - Use Android Asset Studio to generate the icon (Android book pgs. 253-255)
  - Can also download and add icons (https://developer.android.com/studio/write/image-asset-studio)
- From onCreateOptionsMenu(…)
  - Get reference to the menu item and write code to add the icon (required to use FontAwesome icons)

Adding a Font Awesome Menu Icon

- Add dependencies to your build.gradle file:
  - implementation 'com.joanzapata.iconify:android-iconify:2.2.2'
  - implementation 'com.joanzapata.iconify:android-iconify-fontawesome:2.2.2'
- If the above dependencies create Gradle build errors, by bringing in incompatible transitive dependencies
  - 'implementation **(**com.joanzapata.iconify:android-iconify:2.2.2**',**
    { exclude group: 'com.android.support' })
  - implementation **(**'com.joanzapata.iconify:android-iconify-fontawesome:2.2.2**',**
    { exclude group: 'com.android.support' })

Adding a Font Awesome Menu Icon

```
protected void onCreate(…) {
…
   Iconify.with(new FontAwesomeModule());
…
}


public boolean onCreateOptionsMenu(Menu menu) {
   MenuInflater inflater = getMenuInflater();
   inflater.inflate(R.menu.main_menu, menu);

   MenuItem personMenuItem = menu.findItem(R.id.personMenuItem);
   personMenuItem.setIcon(new IconDrawable(this,
      FontAwesomeIcons.fa_user)
         .colorRes(R.color.colorWhite)
         .actionBarSize());

   return true;
}
```

Adding a Menu to a Fragment

- Same as Actvity menus with the following changes:
  - **Extra Step:** Call **setHasOptionsMenu(true)** in **onCreate(Bundle)** or your menu callbacks will never be called
  - Method signature for onCreateOptionsMenu(…) receives a MenuInflater as a parameter
    - Receives the MenuInflater from the hosting activity
- Example: CriminalIntent-Chapt13.zip
  - res/menu/fragment_crime_list.xml
  - CrimeListFragment.java

Implementing an Up Button

- Add an **`android:parentActivityName`** attribute to the manifest entry for the activity that should have an Up button
- Family Map Client Example (in AndroidManifest.xml):

```
<activity
    android:name=".view.PersonActivity"
    android:parentActivityName=".view.MainActivity" />
```

Implementing an Up Button: Using the Same Fragment or Activity Instance

- Simply adding a parent activity will cause the parent activity/fragment to be re-created
- To re-use the same instance (and therefore keep it's state), do the following
  1. Provide a parent activity (per previous slide)
  2. Override onOptionsItemSelected(MenuItem)
  3. If 'up' button selected:
     1. Create an Intent
     2. Set Intent.FLAG_ACTIVITY_SINGLE_TOP and Intent.FLAG_ACTIVITY_CLEAR_TOP flags
     3. Start the parent activity from the intent

Implementing an Up Button: Using the Same Fragment or Activity Instance

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   if(item.getItemId() == android.R.id.home)  {
      Intent intent = new Intent(this, MainActivity.class);
      intent.setFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP |
         Intent.FLAG_ACTIVITY_CLEAR_TOP);
      startActivity(intent);
   }

   return true;
}
```

DIALOGS

Creating a Simple Dialog

- Create an instance of AlertDialog
  - Use AlertDialog.Builder(Context)
- Call show(FragmentManager, String) to make it visible
  - Second param = optional tag that can be used to retrieve the fragment later from the FragmentManager with findFragmentByTag(String)
- **Recommended:** Wrap in a DialogFragment to keep it from disappearing on rotation
- Useful AlertDialog.Builder methods:
  - setTitle(…)
  - setMessage(…)
  - setView(…)
  - setPositiveButton(…)
  - setNegativeButton(…)
  - setNeutralButton(…)
  - Many others
- Example: SimpleDialogExample.java

A More Sophisticated Example

- Android book chapter 12
  - CriminalIntent-Chapt13.zip