# Android UI - Menus

Sang Shin
Michèle Garoche
www.javapassion.com
"Learn with Passion!"

## Disclaimer

- Portions of this presentation are modifications based on work created and shared by the Android Open Source Project
  - http://code.google.com/policies.html
- They are used according to terms described in the Creative Commons 2.5 Attribution License
  - http://creativecommons.org/licenses/by/2.5/

## Topics

- Types of menus
- Options menu
- Context menu
- Submenu
- Creating menu using Menu resource

## Types of Menus

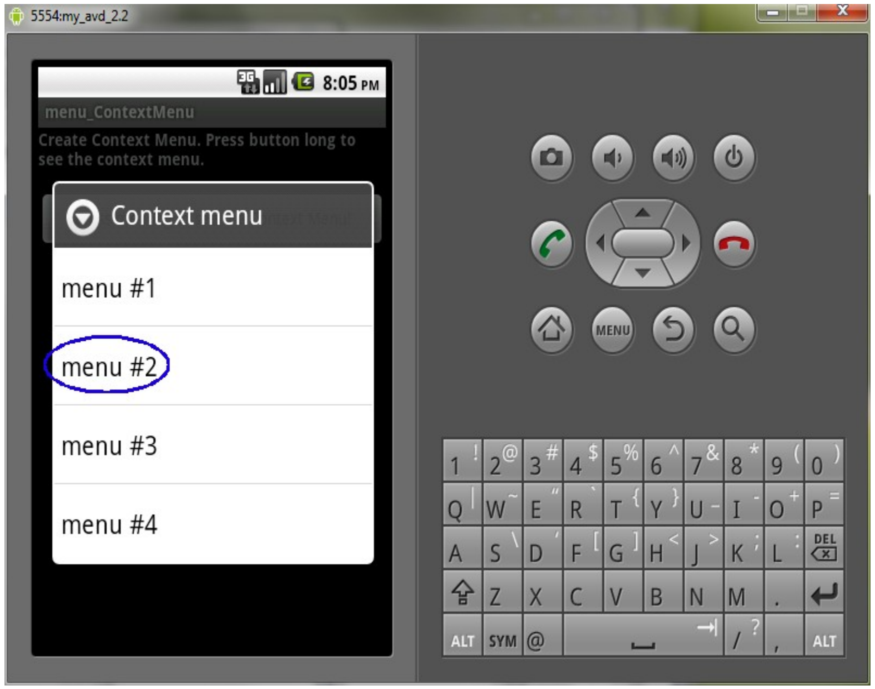
## Types of Menus

- Context menu
  - Floating list of menu items that may appear when you perform a long-press on a View
- Options menu
  - Revealed when the device MENU key is pressed
- Submenu
  - Used to organize menu items into groups
  - A Submenu item does not support nested Submenus

## Context Menu

## Context Menu

- Context menus do not support item shortcuts and item icons.



Create Context Menu. Press button long to see the context menu.

## How to Create Context Menu?

- When Context menu is opened for the first time, the Android system will call the Activity's *onCreateContextMenu(Menu menu)* callback method.
  - You, as a context menu developer, override this method in your Activity class and populate the Menu object given to you with *MenuItem*'s.
- You can populate the menu in two ways
  - Scheme #1: by calling add() for each item you'd like in the menu.
  - Scheme #2: by inflating a menu resource that was defined in XML (preferred)

## Populating Menu with Menu Items: #1

```
// Override this method of Activity class in order to create menu items.
 @Override
 public void onCreateContextMenu(
        ContextMenu menu, // Context menu that is being built
         View view, // The view for which the context menu is being built
         ContextMenuInfo menuInfo) {

     super.onCreateContextMenu(menu, view, menuInfo);
     menu.setHeaderTitle("Context menu");
     menu.add(0, Menu.FIRST     , Menu.NONE, "menu #1");
     menu.add(0, Menu.FIRST + 1, Menu.NONE, "menu #2");
     menu.add(0, Menu.FIRST + 2, Menu.NONE, "menu #3");
     menu.add(0, Menu.FIRST + 3, Menu.NONE, "menu #4");
 }
```

## How to handle User's Menu Selection?

- When a menu item is selected by a user from the Context Menu, *onContextItemSelected()* callback method of your Activity gets called
  - This callback passes you the *MenuItem* that has been selected.
  - You can identify the item by requesting the itemId, with *getItemId()*, which returns the integer that was assigned with the add(int groupId, int itemId, int order, CharSequence title) method.
  - Once you identify the menu item, you can take an appropriate action.

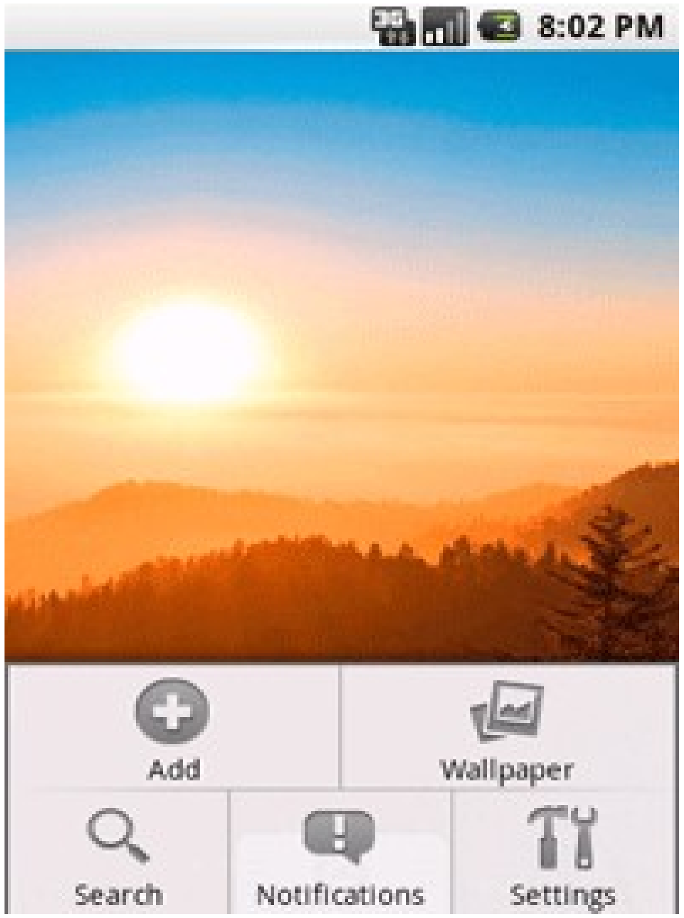
## Example: Handling Menu Selection

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

## Options Menu

## When to use Options Menu?

- The Options Menu is where you should include basic application functions and any necessary navigation items (e.g., to a home screen or application settings).



## How Options Menu Work?

- The Options Menu is opened by pressing the device MENU key.
- When opened, the Icon Menu is displayed, which holds the first six menu items.
- If more than six items are added to the Options Menu, then those that can't fit in the Icon Menu are revealed in the Expanded Menu, via the "More" menu item.

## Populating Menu with Menu Items: #1

```
/* Creates the menu items without Icons */
public boolean onCreateOptionsMenu(Menu menu) {
    // The add() method used in this sample takes four arguments:
    // groupId, itemId, order, and title.
    menu.add(0, MENU_NEW_GAME, 0, "New Game");
    menu.add(0, MENU_QUIT, 0, "Quit");
    return true;
}

/* Creates the menu items with Icons.  Note that add() method returns
    newly created MenuItem object to set additional properties like an icon,
    a keyboard shortcut, an intent, and other settings for the item. */
public boolean onCreateOptionsMenu(Menu menu) {
    // The add() method used in this sample takes four arguments:
    // groupId, itemId, order, and title.
    menu.add(0, MENU_NEW_GAME, 0,
                    "New Game").setIcon(R.drawable.menu_new_game_icon);
    menu.add(0, MENU_QUIT, 0, "Quit").setIcon(R.drawable.menu_quit_icon);
    return true;
}
```

## How to handle Menu Selection?

- When a menu item is selected from the Options Menu, *onOptionsItemSelected()* callback method of your Activity gets called
  - This callback passes you the *MenuItem* that has been selected.
  - You can identify the item by requesting the itemId, with *getItemId()*, which returns the integer that was assigned with the add() method.
  - Once you identify the menu item, you can take an appropriate action.

## Example: Handling Menu Selection

```
/* 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;
}
```

## Submenu

## When to use SubMenu?

- If you have several menu items that can be grouped together with a title, consider organizing them into a Submenu.

## Example: Creating Submenu

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

      // Create submenu "File"
      SubMenu fileMenu = menu.addSubMenu("File");
      fileMenu.add("New");
      fileMenu.add("Open File");
      fileMenu.add("Close");
      fileMenu.add("Close All");

      // Create submenu "Edit"
      SubMenu editMenu = menu.addSubMenu("Edit");
      editMenu.add("Undo Typing");
      editMenu.add("Redo");
      editMenu.add("Cut");

      return result;
   }
```

## Creating Menu using Menu Resource

## What is & Why using Menu Resource?

- Instead of instantiating Menu objects in your application code, you should define a menu and all its items in an XML menu resource, then inflate the menu resource (load it as a programmable object) in your application code.
- Defining your menus in XML is a better practice (than creating them in code) because it separates your interface design from your application code (the same as when you define your Activity layout in XML).

## When to Use Menu Resource File?

- Create <menu_resource>.xml under *res/menu/* directory
- Inflate the Menu Resource file using *inflate(<menu-resource-id>)* method of the *MenuInflator* class
  - Menu objects are created from the Menu resource file

## Example: Inflating Menu Resource

```
public boolean onCreateOptionsMenu(Menu menu) {

   // Inflate the menu XML resource.
   MenuInflater inflater = getMenuInflater();
   inflater.inflate(R.menu.title_only, menu);

   return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {

   switch (item.getItemId()) {
   case R.id.jump:
      Toast.makeText(this, "Jump up in the air!", Toast.LENGTH_LONG)
            .show();
      return true;

   case R.id.dive:
      Toast.makeText(this, "Dive into the water!", Toast.LENGTH_LONG)
            .show();
      return true;


   }
```

## Example: Menu Resource File

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">

   <item android:id="@+id/jump"
      android:title="Jump!"
      android:icon="@drawable/draw_jump" />

   <item android:id="@+id/dive"
      android:title="Dive!"
      android:icon="@drawable/draw_dive" />

</menu>
```

## Thank you!

**Check JavaPassion.com Codecamps!**
**http://www.javapassion.com/codecamps**
**"Learn with Passion!"**