ALGONQUIN COLLEGE

CST2335 GRAPHICAL INTERFACE PROGRAMMING

Week 9

Toolbar Demo

Introduction

- This slide walks you through the process of adding a Toolbar to your app

Introduction

- Toolbars are a place where you can put icons that are commonly used in your application.
- Android used to use a class called ActionBar, which is similar to Toolbar, however ActionBar can only be placed at the top of the screen.
- Toolbar is an improvement since it can be placed anywhere on the screen.

Setting up Toolbar

- To tell Android to use Toolbar and not ActionBar, go to **res/values/themes.xml**
- Add the following two lines within the style tag

```
<item name="windowActionBar">false</item>
<item name="windowNoTitle">true</item>
```

Add Toolbar to the layout of interest

- Add a Toolbar tag to the layout of interest.
- If your activity extends AppCompatActivity then use the following Toolbar tag

```
<androidx.appcompat.widget.Toolbar
        android:id="@+id/myToolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
/>
```

Setting up Toolbar in Java

- In Java, call **setSupportActionBar( );** in the onCreate() method and pass in your Toolbar object from the viewBinding:

```
setSupportActionBar(binding.myToolbar)
```

onCreateOptionsMenu()

- In your **ChatRoom.java** class, hit "Ctrl+O" and select **onCreateOptionsMenu** from the list of functions to generate.

```
@Override
   public boolean onCreateOptionsMenu(Menu menu) {
      super.onCreateOptionsMenu(menu);

      return true;
   }
```

onCreateOptionsMenu() - II

- This function loads a Menu layout file.
- It's like the LayoutInflater that you used for the RecyclerView, only now it expects a menu file in the */* **res/menu** folder.

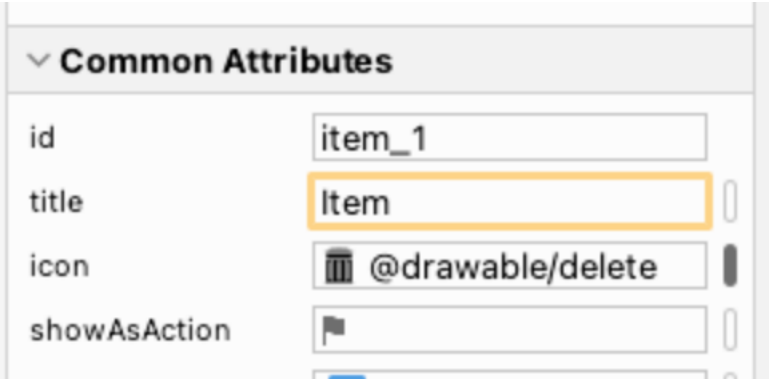
```
@Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
                getMenuInflater().inflate(R.menu.myMenu, menu);

        return true;
    }
```

Menu Layout Files

- Right-click on the "**res**" folder and select "**New**" -> "**Android Resource File**".
- From the window that appears, select "**Menu**" as the resource type and give it a filename of "**my_menu**".
- Afterwards, hit the "Ok" button.
- You should see a Menu editor, with a Code, Design, or Split view just like when you open a layout file.

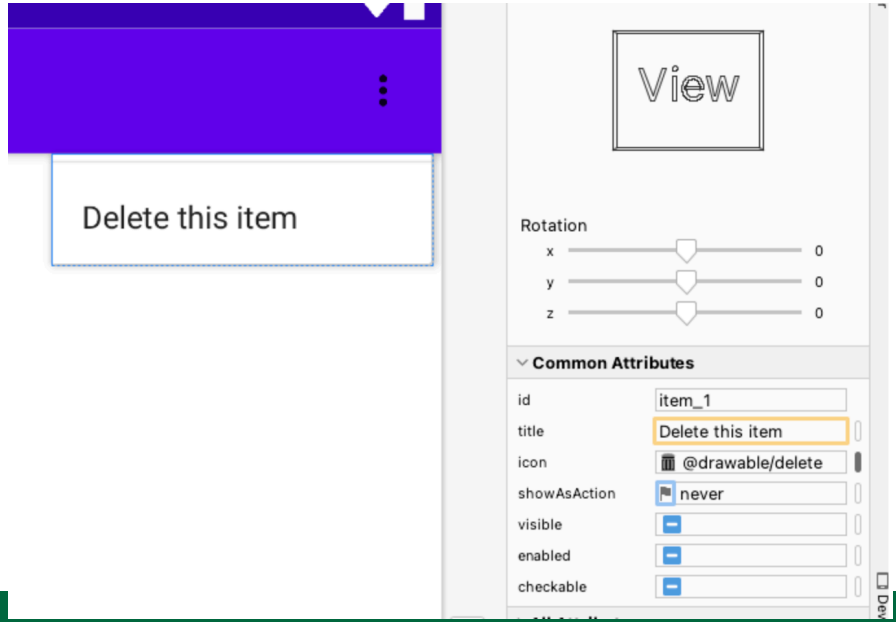
Menu Layout Files

- Open the "Palette" window and drag a MenuItem onto the toolbar.
- If you look at the Attributes window on the right side, give the item an id, "**item_1**" for example. Then let's add an icon to use as a picture for this menu item.



showAsAction attribute

- This determines how the button shows up: **always**, **never**, **ifRoom**, etc.



onOptionsItemSelected

- When the user clicks on a menu item, Android will call a function called **onOptionsItemSelected**.
- Press "Ctrl+O" and implement the **onOptionsItemSelected** function in your desired activity (e.g. ChatRoom.java).

onOptionsItemSelected

```
@Override
   public boolean onOptionsItemSelected(@NonNull MenuItem item) {
      switch( item.getItemId() )
      {
         case R.id.item_1:
            //put your code here
            break;
      }
      return true;
   }
```

Toolbar Summary

1. Add a ToolBar to your Activity layout file.
2. In the Activity's onCreate, get the ToolBar and call setSupportActionToolbar().
3. Create a Menu resource file in XML with Items in the menu.
4. In onCreateOptionsMenu(), inflate the Menu resource.
5. Handle each Item id in onOptionsItemSelected()