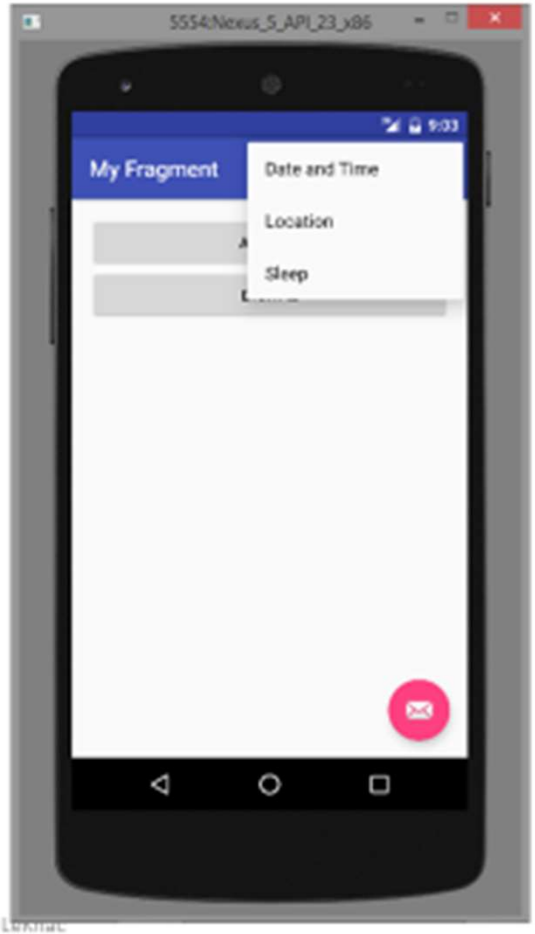
UCD School of Computer Science

Android App Development
Menu

Dr. Abraham (Abey) Campbell

Objective

- Menu



Learning Objectives

- Use Menus, one menu item per activity
- Use Sqlite to handle persistent data

Menus and Sqlite

- Menus (in the action bar)
- Menus using text, using icons
- Using Sqlite to store persistent data (SQL syntax)
- Menu item ➔ SQL operation (insert, delete, update, select)


5554:Nexus_5_API_23_x86
6:50
My Fragment
ANALOG
DIGITAL

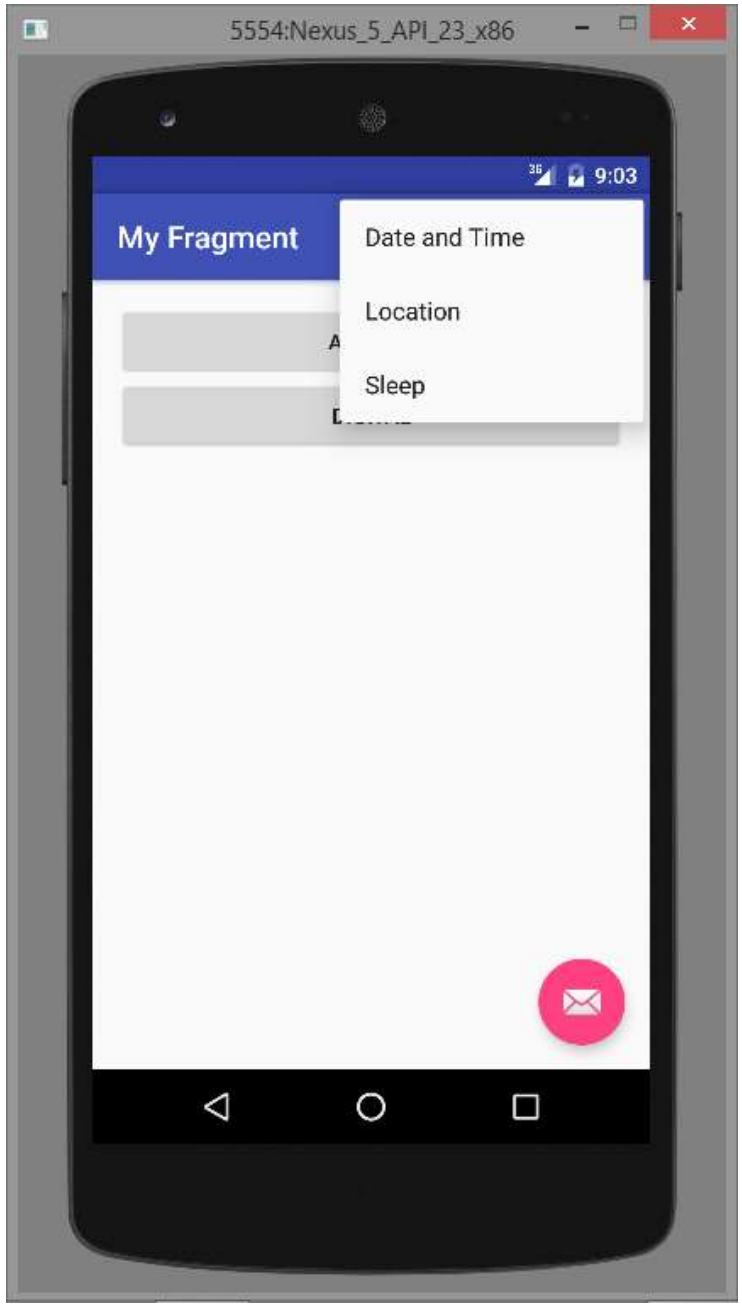
Menus

- When we start a project using the Basic Activity template, menu code is automatically generated.
- menu_main.xml file is automatically generated.
- Menu related methods are automatically coded in MainActivity.

menu_main.xml (1 of 2)

- menu element
- One item element inside
- There could be more item elements added.
- menu_main.xml defines a menu using an XML resource (and a menu can also be defined programmatically).


5554:Nexus_5_API_23_x86
9:03
My Fragment
ANALOG
DIGITAL


5554:Nexus_5_API_23_x86
9:03
My Fragment
Date and Time
Location
Sleep

menu_main.xml (2 of 2)

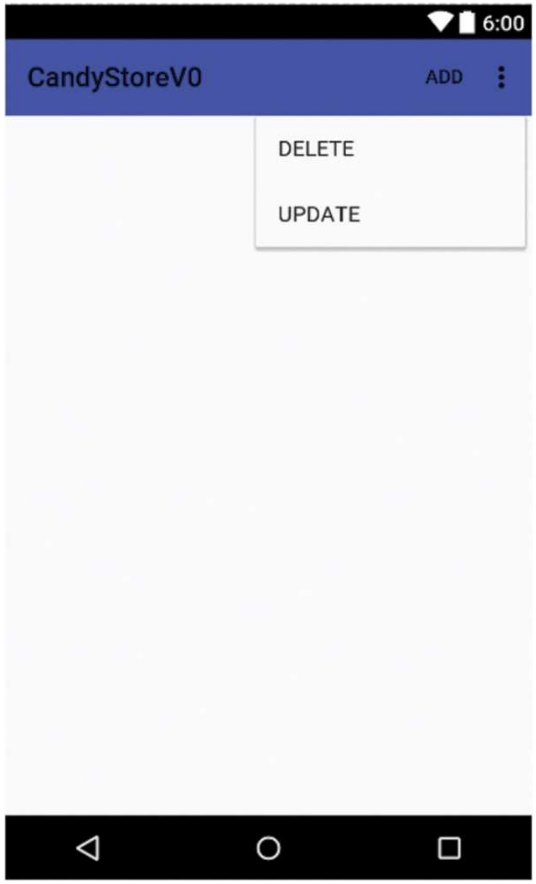
```
<item
    android:id="@+id/action_settings"
    android:title="@string/action_settings"
    android:orderInCategory="100"
    app:showAsAction="never"/>
```

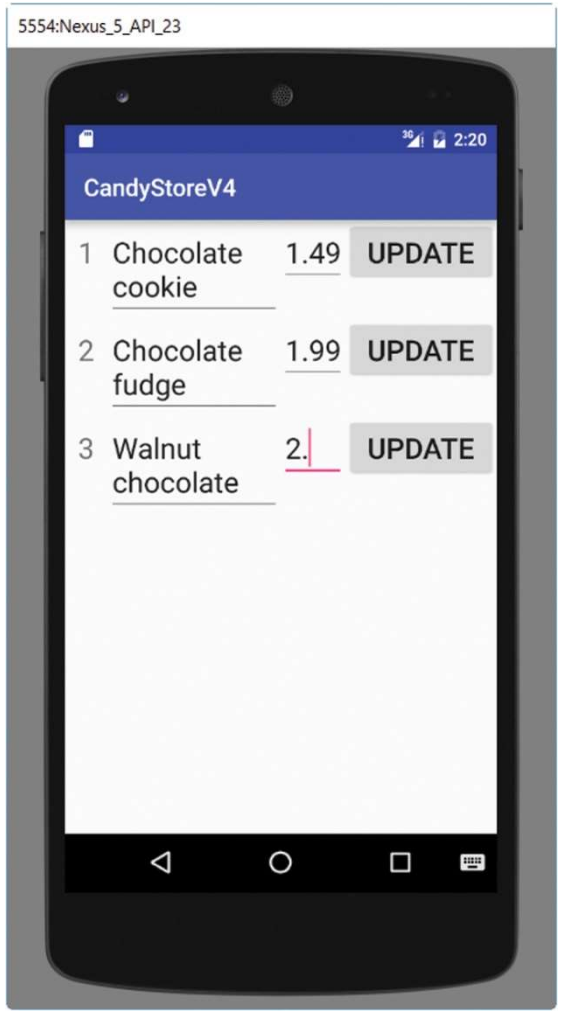
Item Element / MenuItem Class

XML attribute of item	Method of MenuItem
android:title	setTitle
android:icon	setIcon
app:showAsAction	setShowAsAction

app:showAsAction

- Possible values are: never, ifRoom, always, withText ...
- ifRoom ➔ let the system decide to show it or not based on available space.
- If there is not enough space, there will be a submenu so that the user can select it.

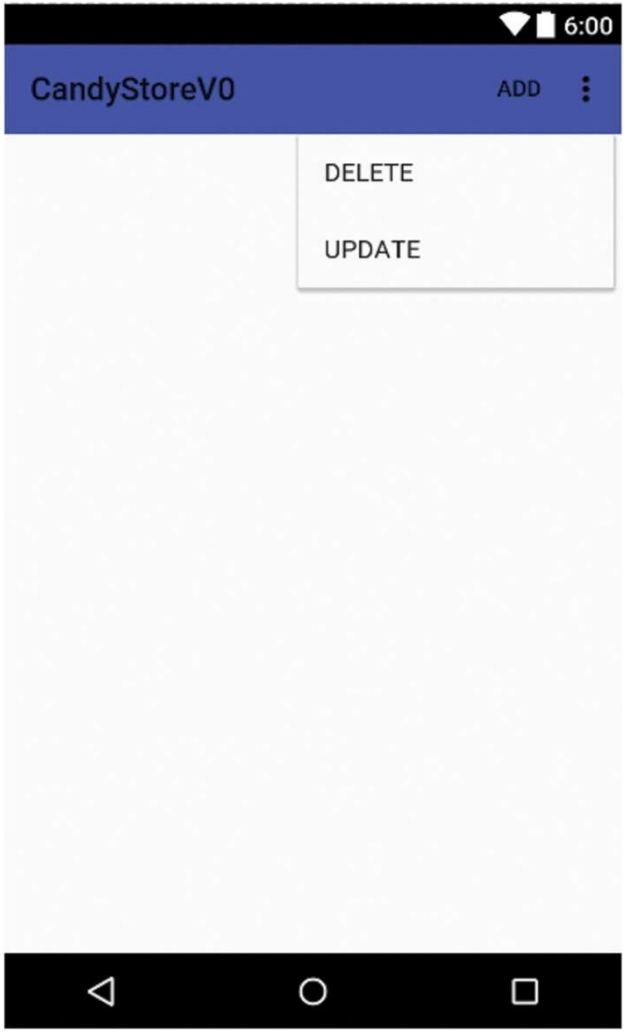

6:00
CandyStoreV0
ADD
DELETE
UPDATE


5554:Nexus_5_API_23
2:20
CandyStoreV4
1 Chocolate cookie
1.49
UPDATE
2 Chocolate fudge
1.99
UPDATE
3 Walnut chocolate
2.
UPDATE


5554:Nexus_5_API_23_x86
10:11
CandyStoreV3
1; Chocolate cookie; 1.49
2; Chocolate fudge; 1.99
3; Walnut chocolate; 2.99
BACK

menu_main.xml (1 of 3)

- We modify menu_main.xml so that it contains three menu item elements (add, delete, update).



menu_main.xml (2 of 3)

```
<item
    android:id="@+id/action_add"
    android:title="@string/add"
    app:showAsAction="ifRoom"/>
<item android:id="@+id/action_delete"
   android:title="@string/delete"
   app:showAsAction="ifRoom"/>
<item
    …
```

menu_main.xml (3 of 3)

- add and delete strings are defined in strings.xml.
- Ids are needed to identify the item the user selected (in the code).
- ifRoom ➔ the item shows in the action bar if there is room for it.

Showing the Menu (1 of 2)

- Inside MainActivity, onCreateOptionsMenu is automatically called: it displays the menu if there is one.
- The menu items show on the right side of the action bar.

Showing the Menu (2 of 2)

- The onCreateOptionsMenu method inflates menu_main.xml in order to create a menu and places that menu in the toolbar.

```
public boolean onCreateOptionsMenu( Menu
    menu ) {
    getMenuInflater( ).inflate(
        R.menu.menu_main, menu );
    return true;
}
```

Choosing from the Menu (1 of 3)

- Processing a selection from the menu
- Inside MainActivity, the onOptionsItemSelected method is automatically called when the user clicks on a menu item: we place our code here to process the user selection.

Choosing from the Menu (2 of 3)

- onOptionsItemSelected method

```
public boolean onOptionsItemSelected( MenuItem
    item ) {
  // Handle item selection
  switch ( item.getItemId( ) ) {
    case R.id.action_add:
    …
```

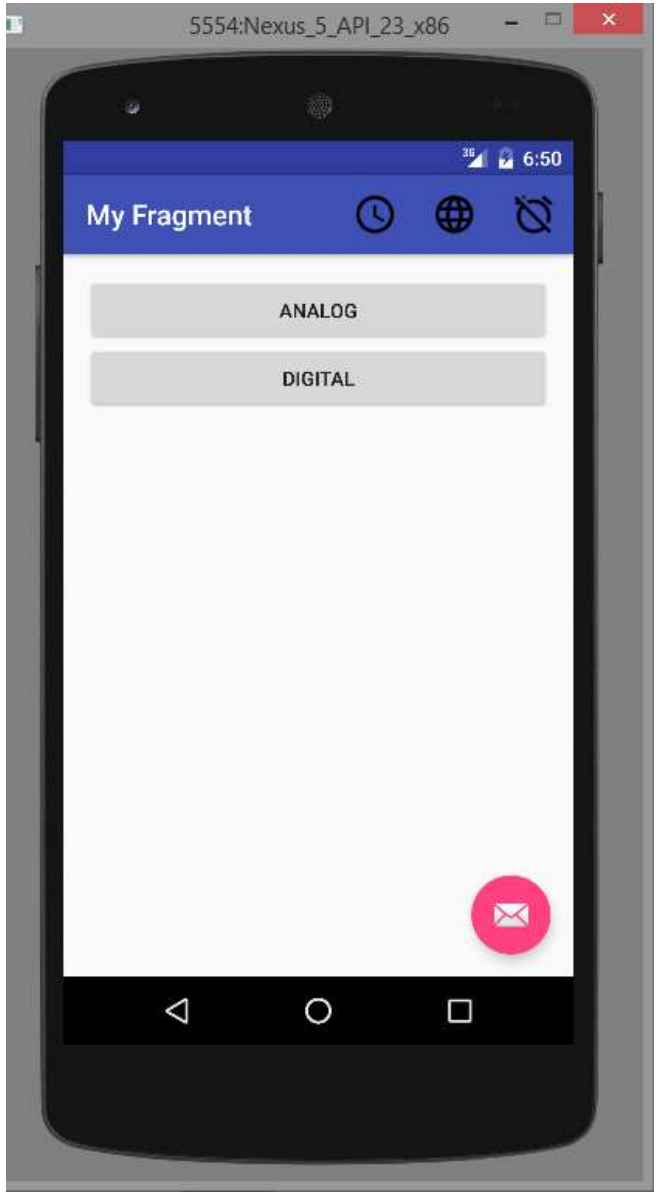
Choosing from the Menu (3 of 3)

- Add Log statements for feedback.
- ➔ we can identify what menu item was selected.

Using Icons in the Menu (1 of 2)

- In Version 1, instead of text, we can use icons.

Add Icons


5554:Nexus_5_API_23_x86
6:50
My Fragment
ANALOG
DIGITAL

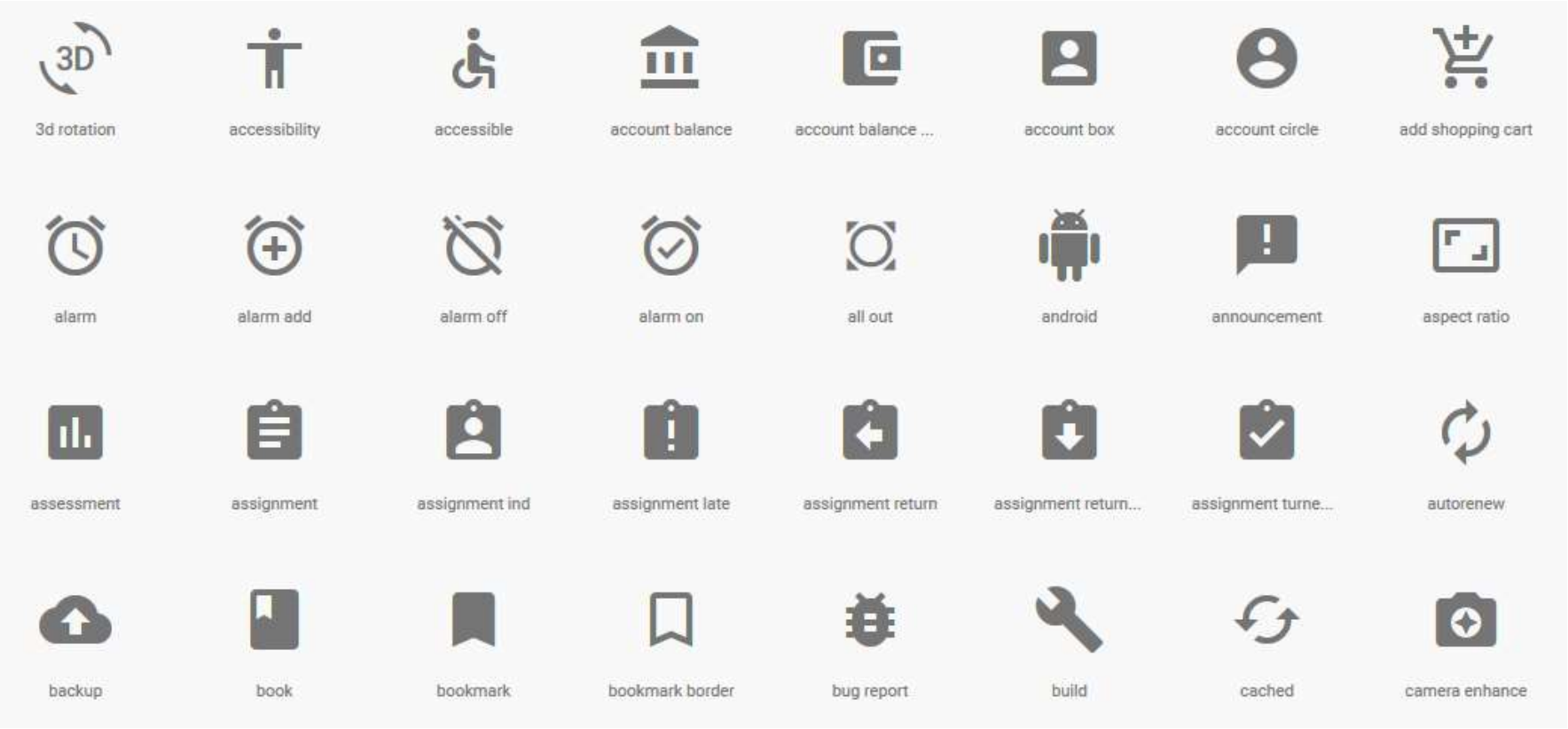

6:00
CandyStoreV1
DELETE
UPDATE

Using Icons in the Menu (2 of 2)

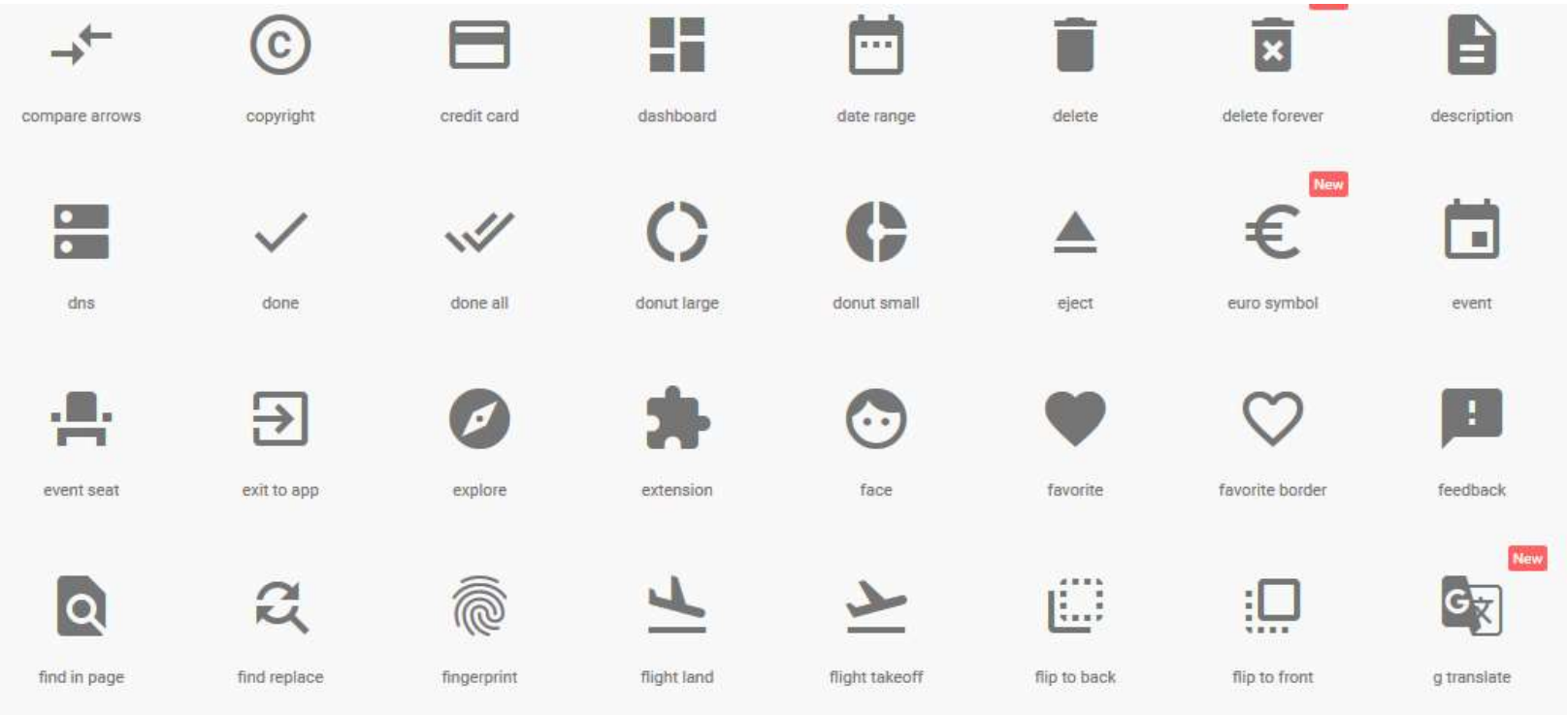
- We need either to use existing icons from the Google library or create icons (for example png files) and place them in the drawable directory.
- The size of the icons does not really matter; they will be automatically resized when placed inside the action bar.

Add Icons

http://www.google.com/design/icons/



Add Icons


compare arrows
copyright
credit card
dashboard
date range
delete
delete forever
description
dns
done
done all
donut large
donut small
eject
New
euro symbol
event
event seat
exit to app
explore
extension
face
favorite
favorite border
feedback
find in page
find replace
fingerprint
flight land
flight takeoff
flip to back
flip to front
New
g translate

menu_main.xml (1 of 4)

- Inside the item element, we use the android:icon attribute and specify an icon resource.
- We can use existing icons available in the Android library or create our own icons.

menu_main.xml (2 of 4)

- Existing icons can be referenced using the following syntax and pattern:

  @android:drawable/name_of_icon

- We use three existing icons: ic_menu_add, ic_menu_delete, and ic_menu_edit.

```
android:icon="@android:drawable/ic_menu_add"
```

menu_main.xml (3 of 4)

```
…
<item android:id="@+id/action_add"
        android:title="@string/add"
        android:icon=
            "@android:drawable/ic_menu_add"
       app:showAsAction="ifRoom"/>
…
```

menu_main.xml (4 of 4)

- If both a title and an icon are specified inside an item element, the icon has higher preference and will show.
- The title no longer shows.

Processing a Menu Selection

- When the user selects a menu item, for example ADD, we want to start a new activity.
- That new activity enables the user to enter data for a new candy and inserts that new candy in the candy database (using Sqlite).

Starting a New Activity

Two steps:

- Create an intent to start an activity
- Start the activity

Creating an Intent

- Intent class
- Intent(Context, Class) constructor:

```
Intent insertIntent = new Intent( this,
        InsertActivity.this );
```

Starting an Activity

- Once we have an Intent, we use the startActivity method of the Context class (inherited by the Activity class ...)

  startActivity(insertIntent);

Processing a Menu Selection (1 of 2)

- If the new activity is an InsertActivity, to start it:

```
Intent insertIntent = new Intent( this,
    InsertActivity.class );
this.startActivity( insertIntent );
```

Processing a Menu Selection (2 of 2)

```
public boolean onOptionsItemSelected( MenuItem
   item ) {
   switch ( item.getItemId( ) ) {
     case R.id.action_add:
       Intent insertIntent = new Intent(
           this, InsertActivity.class );
       this.startActivity( insertIntent );
       …
```

Adding a Candy

- We neeed to create and code the InsertActivity class and provide a View (an XML layout file) for it
- ➔ Create InsertActivity.java
- ➔ Create activity_insert.xml

View for Adding a Candy

activity_insert.xml (1 of 2)

- We use a RelativeLayout.
- We include two pairs of TextView/EditText.
- We include two Buttons (one to insert the data, one to go back to the main activity).
- We use ids to position the elements and retrieve them later using findViewById in InsertActivity class.

activity_insert.xml (2 of 2)

- See Example
- We also include a few Strings in the strings.xml file that we use in activity_insert.xml.

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  android:paddingBottom="@dimen/activity_vertical_margin">

  <TextView
    android:id="@+id/label_name"
    android:layout_marginTop="50dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/label_name"/>

  <EditText
    android:id="@+id/input_name"
    android:layout_toRightOf="@+id/label_name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id/label_name"
    android:layout_marginLeft="50dp"
    android:orientation="horizontal" />

  <TextView
    android:id="@+id/label_price"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/label_name"
    android:layout_marginTop="50dp"
    android:text="@string/label_price" />

  <EditText
    android:id="@+id/input_price"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id/label_price"
    android:layout_alignLeft="@+id/input_name"
    android:layout_alignParentRight="true"
    android:layout_toRightOf="@+id/label_price"
    android:inputType="numberDecimal" />

  <Button
    android:id="@+id/button_add"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_below="@+id/label_price"
    android:layout_marginTop="50dp"
    android:onClick="insert"
    android:text="@string/button_add" />

  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_below="@+id/button_add"
    android:layout_marginTop="50dp"
    android:onClick="goBack"
    android:text="@string/button_back" />

</RelativeLayout>
```

AndroidManifest.xml

We need to update AndroidManifest.xml, add an activity element for the insert activity.

```
<activity
    android:name=".InsertActivity"
   android:label="@string/app_name" >
</activity>
```

InsertActivity Methods (1 of 2)

- The goBack method takes us back to main activity.
- ➔ It pops the current activity off the stack.

```
public void goBack( View v ) {
    this.finish( );
}
```

InsertActivity Methods (2 of 2)

- The insert method retrieves user input, inserts it in the candy database, and clears the EditText fields.
- We use findViewById to retrieve EditTexts, then retrieve user input.
- ➔ to insert in database, we need to access Sqlite.

InsertActivity: insert Method (1 of 3)

```
public void insert( View v ) {
    // retrieve name and price

    // insert new candy in database

    // clear data in the two EditTexts
}
```

InsertActivity: insert Method (2 of 3)

```
// Retrieve name and price
EditText nameEditText =
    ( EditText) findViewById( R.id.input_name );
EditText priceEditText =
    ( EditText) findViewById( R.id.input_price );
String name = nameEditText.getText( ).toString( );
String priceString =
    priceEditText.getText( ).toString( );
```

InsertActivity: insert Method (3 of 3)

```
// insert new candy in database

// clear data
nameEditText.setText( "" );
priceEditText.setText( "" );
```