Menus and Dialogs

Session 4

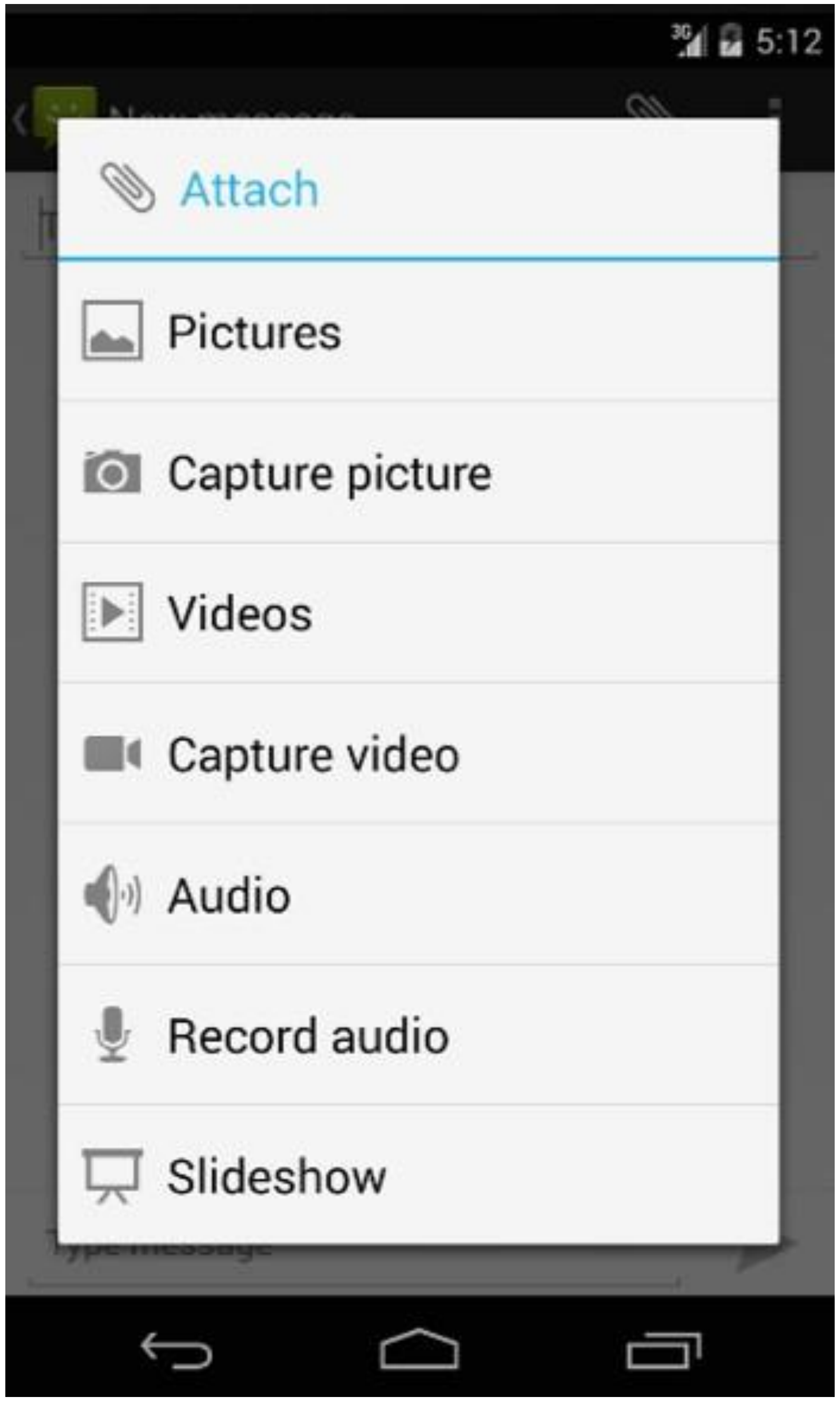
Objective

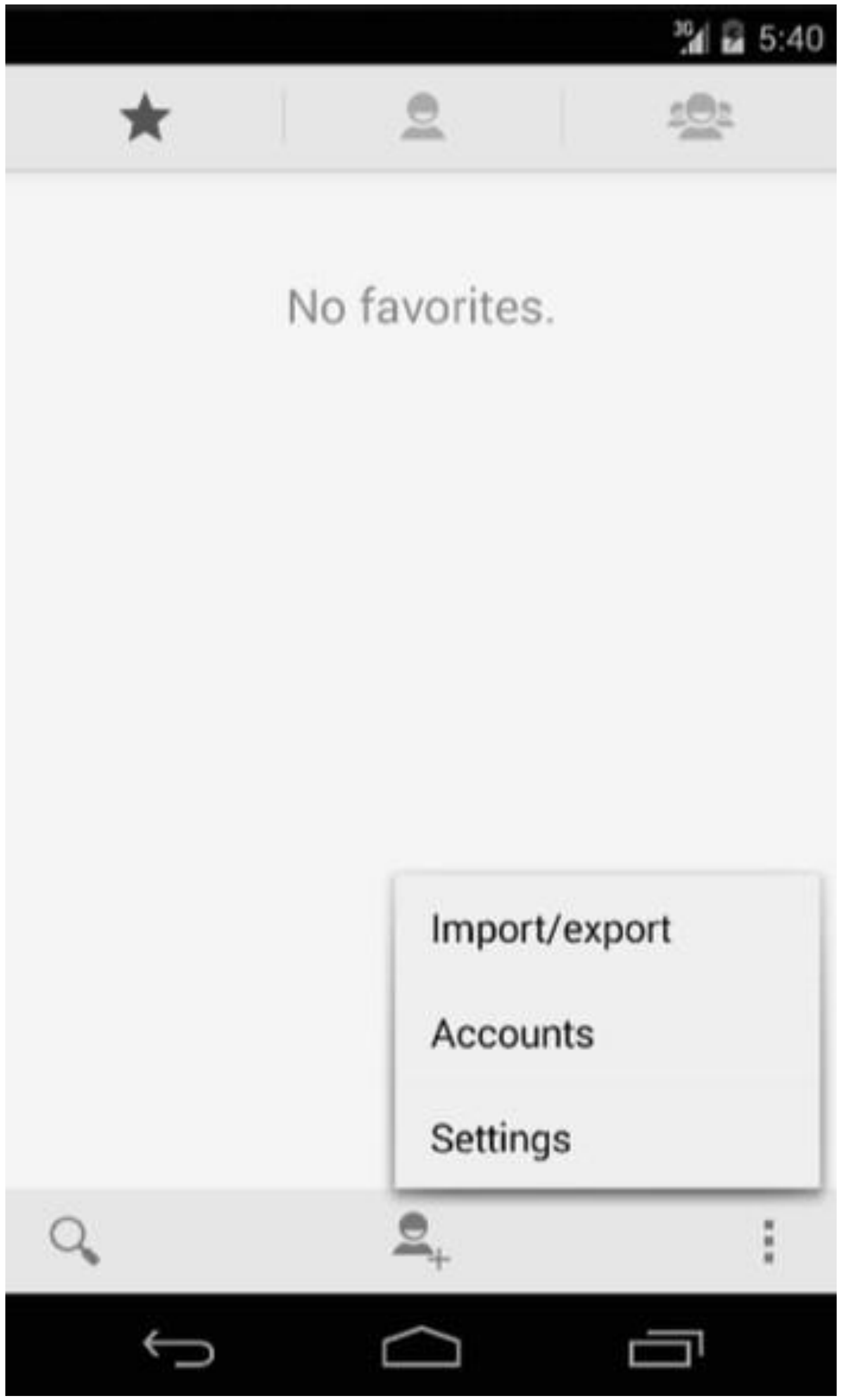
- Android menus:
  - Options menu
  - Expanded menu
  - Context menu
  - Submenus
- Dialogs:
  - AlertDialog
  - ProgressDialog
  - DatePickerDialog
  - TimePickerDialog
  - Custom Dialog

Android Menus

- Menus cung cấp cho người sử dụng những cách để thực hiện hành động (chức năng). Menu thường sử dụng khi có nhiều chức năng mà không muốn chiếm nhiều không gian
- Thông thường có 3 loại menus:
  - Options Menu: Các items mở ra khi bấm MENU hoặc sử dụng Action Bar (Android 3.0)
  - Context Menu: List chức năng xuất hiện khi người dùng bấm và giữ (long click) lên View.
  - Popup menu: Hiển thị menu bên dưới văn bản neo, nếu bấm ra ngoài popup thì menu sẽ biến mất.

Android Menus





Using Menus

- Menus may include:
  - Text
  - Icons
  - Radio Buttons
  - Check Boxes
  - Sub-menus
  - Short-cut keys
  - Expanded menu

Options Menu

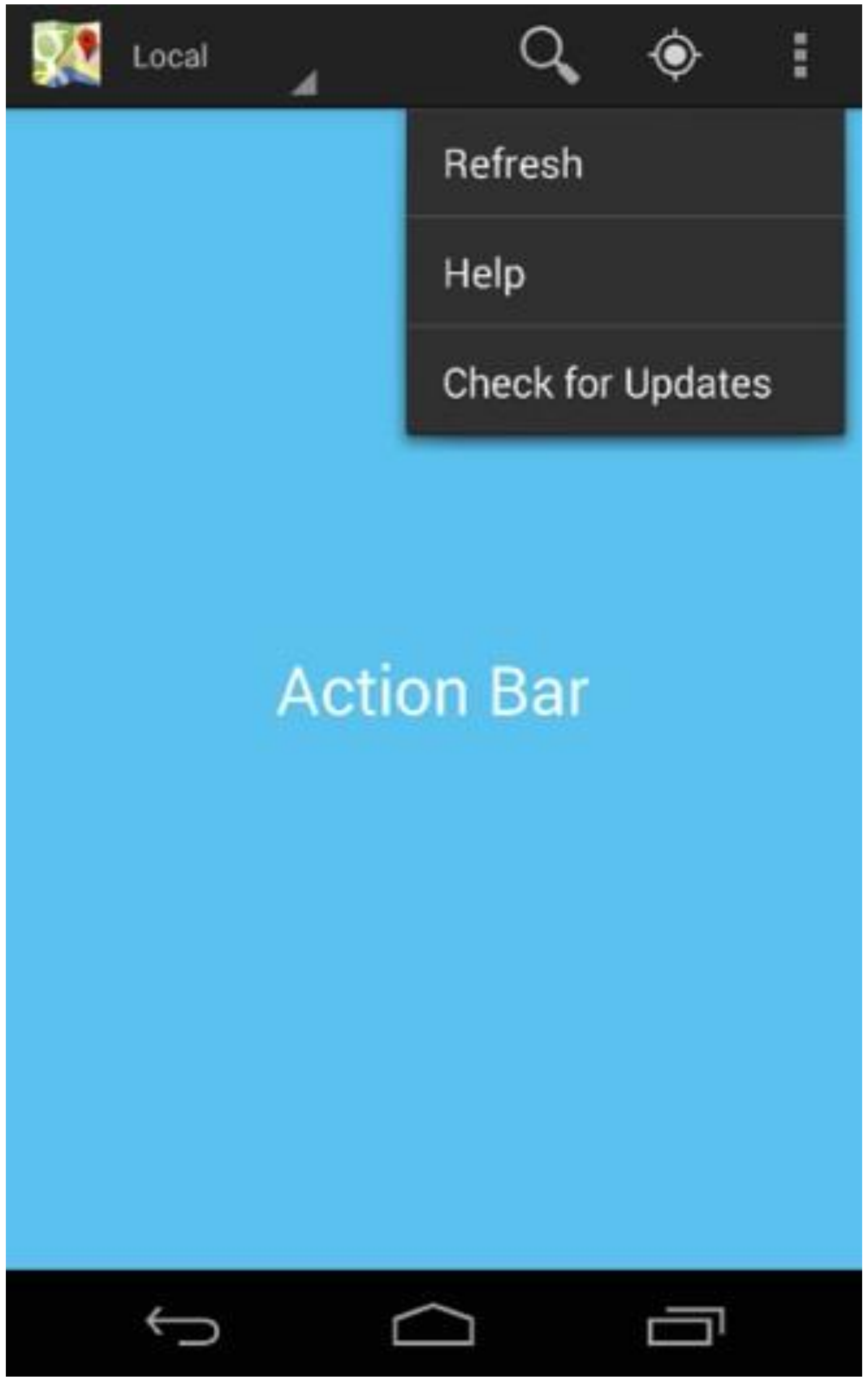
- Không hiển thị checkboxes, radio buttons, hoặc shortcut keys cho Menu Items
- Tạo xml cho menu (có thể trong res/layout hoặc res/menu)

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
      <item android:id="@+id/action_help"
            android:icon="@drawable/ic_action_refresh"
            android:title="@string/action_refresh"/>
       <item android:id="@+id/action_help"
            android:icon="@drawable/ic_action_help"
            android:title="@string/action_help"  />
</menu>
```

Options Menu

onCreateOptionsMenu() method. Sử dụng method này để inflate một menu resource định nghĩa các action trong menu:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
	// Inflate the menu items for use in the action bar
	MenuInflater inflater = getMenuInflater();
	inflater.inflate(R.menu.main_activity_actions, menu);
	return super.onCreateOptionsMenu(menu);
}
```



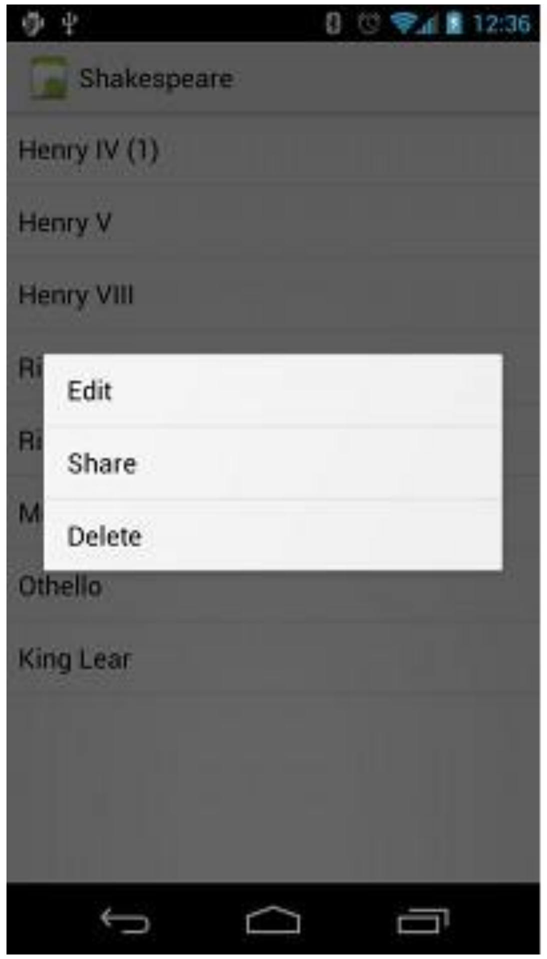
Context Menu

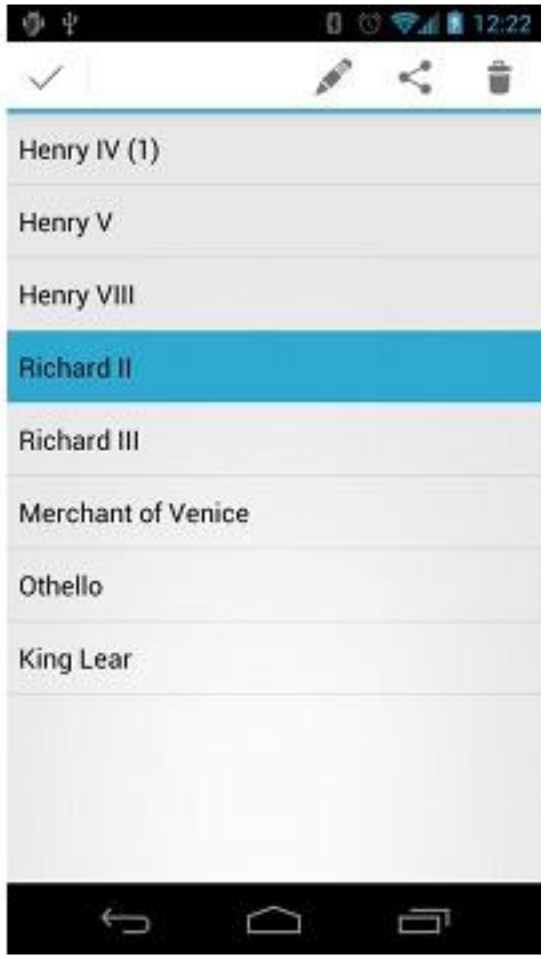
- Ta có thể cung cấp một context menu cho bất cứ view nào, các view phổ biến thường dùng cho context menu là ListView, GridView
- Tạo context menu:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                        ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.context_menu, menu);
}
```

Sample UI

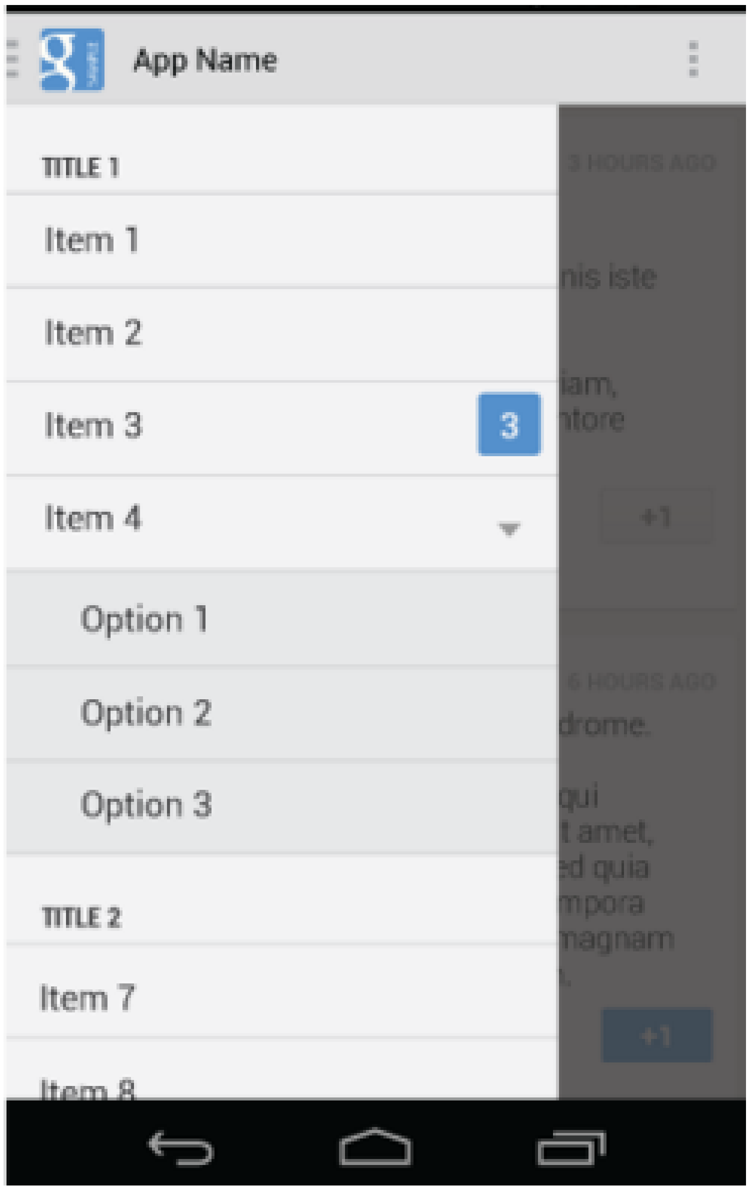
```
@Override
public boolean onContextItemSelected(MenuItem item) {
        AdapterContextMenuInfo info =
                 (AdapterContextMenuInfo)item.getMenuInfo();
        switch (item.getItemId()) {
                case R.id.edit:
                        editNote(info.id);
                        return true;
                case R.id.delete:
                        deleteNote(info.id);
                        return true;
            default:
                        return super.onContextItemSelected(item);
        }
}
```





Sub Menus

- Name of the submenu is shown in the header bar
- Display full texts, checkboxs, and shortcut keys
- Can add a submenu to asubmenu
- Press back to return main menu



Creating a Menu Resource

- Define a menu and all its items in an XML menu resource
- Using a menu resource to separate the content for the menu from your application code
- To create a menu resource, create an XML file inside your project's res/menu/ directory
- Build the menu with the following elements:
  - <menu>: define menu items
  - <item>: define single items
  - <group>: cageorize menu items

Creating a Menu Resource

- An example menu named game_menu.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/new_game" android:icon="@drawable/ic_new_game" android:title="@string/
      new_game" />
    <item android:id="@+id/help" android:icon="@drawable/ic_help" android:title="@string/help" />
</menu>
```

- Each item includes the attributes:
  - android:id - A resource ID that's unique to the item
  - android:icon - A reference to a drawable to use as the item's icon.
  - android:title - A reference to a string to use as the item's title. ...

Inflating a Menu Resource

- Inflate a menu resource (convert the XML resource into a programmable object) using MenuInflater.inflate().
- Call MenuInflater.inflate() during the onCreateOptionsMenu() callback method:

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

Creating an Options Menu

1. Creating a Menu Resource
2. Calls your activity's onCreateOptionsMenu() method to create a Option Menu
3. Override onOptionsItemSelected() to response to user action

```
4.      @Override
5.      public boolean onOptionsItemSelected(MenuItem
        item) { switch (item.getItemId()) {
        case R.id.new_game:
            newGame();
            return true;
        case  R.id.help:
            showHelp();
            return true;
        default:
            return super.onOptionsItemSelected(item);
        }
6.  }
```

Creating a Context Menu

- A context menu is conceptually similar to the menu displayed when the user performs a "right-click" on a PC
- On Android, a context menu is displayed when the user performs a "long press" (press and hold) on an item.
- Create a context menu for any View, though context menus are most often used for items in a ListView
- Step to create a Context Menu:

1. Define a menu on Menu Resource File
2. Calls your activity's onCreateContextMenu() method to create a Option Menu
3. Override onContextItemSelected() to response to user action

Creating a Context Menu

- Example:

```
@Override
public boolean onContextItemSelected(MenuItem item) {
    AdapterContextMenuInfo info = (AdapterContextMenuInfo) item.getMenuInfo();
    switch (item.getItemId()) {
        case R.id.edit:
            editNote(info.id);
            return true;
        case R.id.delete:
            deleteNote(info.id);
            return true;
        default:
            return super.onContextItemSelected(item)
    }
}
```

Create a Submenus

- A submenu is a menu that the user can open by selecting an item in another menu
- You can add a submenu to any menu (except a submenu)
- Create a submenu by adding a <menu> element as the child of an <item>

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/file"
          android:icon="@drawable/file"
          android:title="@string/file" >
        <!-- "file" submenu -->
        <menu>
            <item android:id="@+id/create_new"
                  android:title="@string/create_new" />
            <item android:id="@+id/open"
                  android:title="@string/open" />
        </menu>
    </item>
</menu>
```

Menu Groups

- A menu group is a collection of menu items that share certain traits. With a group, you can:
  - Show or hide all items with setGroupVisible()
  - Enable or disable all items with setGroupEnabled()
  - Specify whether all items are checkable with setGroupCheckable()
- A an example menu resource that includes a group:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/item1"
         android:icon="@drawable/item1"
         android:title="@string/item1" />
   <group android:id="@+id/group1">
      <item android:id="@+id/groupItem1"
            android:title="@string/groupItem1" />
      <item android:id="@+id/groupItem2"
            android:title="@string/groupItem2" />
   </group>
</menu>
```

Shortcut Keys

- This shortcut key is displayed as a tip in the menu item, below the menu item name
- Shortcut keys for menu items only work on devices with a hardware keyboard
- Shortcut keys are *not* case sensitive.
- Shortcuts cannot be added to items in a Context Menu.
- Two ways to add shortcut keys:
  - Use android:alphabeticShortcut and android:numericShortcut attributes in the <item> element
  - Use the methods setAlphabeticShortcut(char) and setNumericShortcut(char)

Dialogs

- Android provides some primitive forms of dialog boxes:
  - AlertDialog: A dialog that can manage the buttons, and/or a list of selectable items that can include checkboxes or radio buttons
  - ProgressDialog: A dialog that displays a progress wheel or progress bar
  - DatePickerDialog: A dialog that allows the user to select a date
  - TimePickerDialog: A dialog that allows the user to select a time

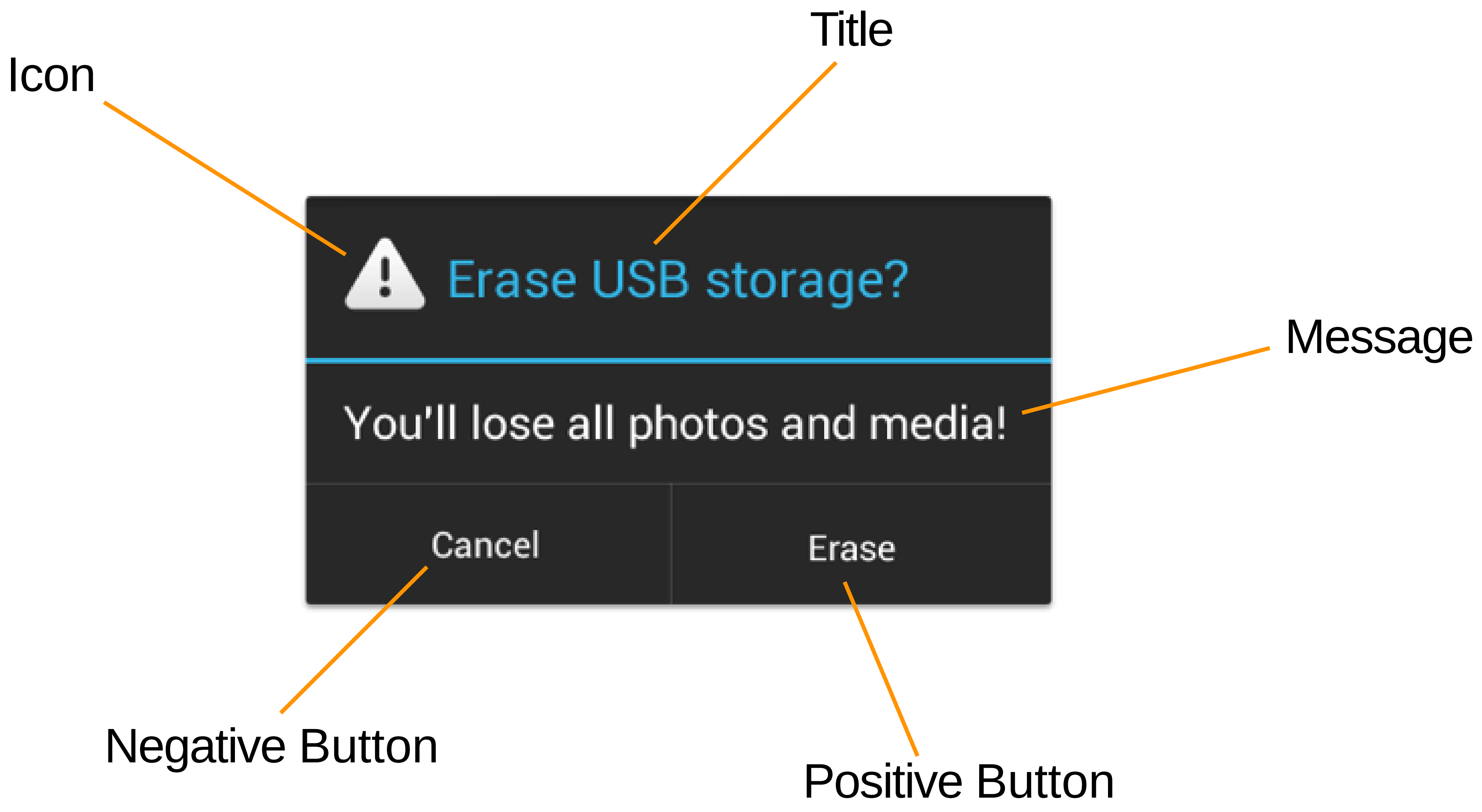
Dialogs

- Android provides some primitive forms of dialog boxes:
  - AlertDialog: A dialog that can manage the buttons, and/or a list of selectable items that can include checkboxes or radio buttons
  - ProgressDialog: A dialog that displays a progress wheel or progress bar
  - DatePickerDialog: A dialog that allows the user to select a date
  - TimePickerDialog: A dialog that allows the user to select a time

Alert Dialog

- Android provides some primitive forms of dialog boxes:
  - AlertDialog: A dialog that can manage the buttons, and/or a list of selectable items that can include checkboxes or radio buttons
  - ProgressDialog: A dialog that displays a progress wheel or progress bar
  - DatePickerDialog: A dialog that allows the user to select a date
  - TimePickerDialog: A dialog that allows the user to select a time

Alert Dialog



Alert Dialog - Adding Buttons

- To create an AlertDialog, use the AlertDialog.Builder subclass
- After you're done with the Builder, retrieve the AlertDialog object with create().
- To create an AlertDialog with buttons, use the set...Button() methods:

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage("Are you sure you want to exit?")
       .setCancelable(false)
       .setPositiveButton("Yes", new DialogInterface.OnClickListener(){
           public void onClick(DialogInterface dialog, int id){
                MyActivity.this.finish();
           }
       })
       .setNegativeButton("No", new DialogInterface.OnClickListener(){
           public void onClick(DialogInterface dialog, int id){
                dialog.cancel();
           }
       });
AlertDialog alert = builder.create();
```

Alert Dialog - Adding a List

- To create an AlertDialog with a list of selectable items like the one shown to the right, use the setItems() method:

```
final CharSequence[] items = {"Red", "Green", "Blue"};
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Pick a color");
builder.setItems(items, new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int item) {
        Toast.makeText(getApplicationContext(), items[item], Toast.LENGTH_SHORT).show();
    }
});
AlertDialog alert = builder.create();
```

Adding Checkboxes and Radio Buttons

- To create a list of multiple-choice items (checkboxes) or single-choice items (radio buttons) inside the dialog, use the setMultiChoiceItems() and setSingleChoiceItems() methods, respectively.
- Create one of these selectable lists in the onCreateDialog() callback method
- Use the same code from the previous example, but replace the setItems() method with setSingleChoiceItems():

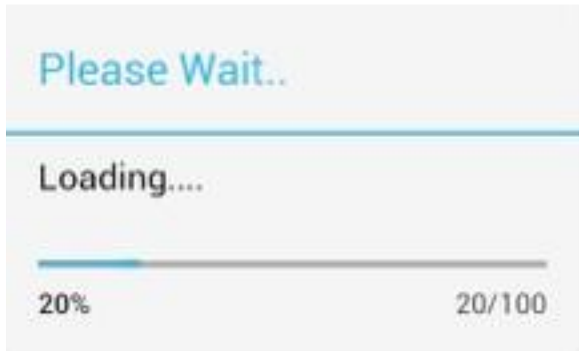
```
final CharSequence[] items = {"Red", "Green", "Blue"};
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Pick a color");
builder.setSingleChoiceItems(items, -1, new DialogInterface.OnClickListener(){
    public void onClick(DialogInterface dialog, int item) {
        Toast.makeText(getApplicationContext(), items[item], Toast.LENGTH_SHORT).show();
    }
});
AlertDialog alert = builder.create();
```

Create a Progress Dialog

- A ProgressDialog is an extension of the AlertDialog class that can display a progress animation in the form of a spinning wheel
- The dialog can also provide buttons, such as one to cancel a download.
- Opening a progress dialog can be as simple as calling ProgressDialog.show(). For example:

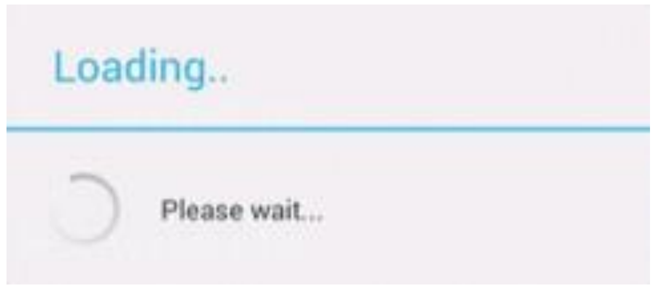
```
ProgressDialog dialog = ProgressDialog.show(MyActivity.this, "", "Loading. Please wait...", true);
```

- Using setProgressStyle, setMessage, setCancelable and so on to setting a progress


Please Wait..

Loading....

20% 20/100


Loading..

Please wait...

Create Date Picker Dialog

1. Start a new project named *HelloDatePicker*
2. Edit the res/layout/main.xml file
3. Open HelloDatePicker.java and add the following members to the class:

```
private TextView mDateDisplay;
private Button mPickDate;
private int mYear;
private int mMonth;
private int mDay;
static final int DATE_DIALOG_ID = 0;
```

4. Edit onCreate() method (code)
5. Add the updateDisplay() method
6. Initialize a new DatePickerDialog.OnDateSetListener as a member of the HelloDatePicker class
7. Add the onCreateDialog(int) callback method
8. Run the application

Create Time Picker Dialog

1. Start a new project named *HelloTimePicker*
2. Edit the res/layout/main.xml file
3. Open HelloTimePicker.java and add the following members to the class:

```
private TextView mDateDisplay;
private Button mPickDate;
private int mYear;
private int mMonth;
private int mDay;
static final int DATE_DIALOG_ID = 0;
```

4. Edit onCreate() method (code)
5. Add the updateDisplay() and pad() method
6. Add a class member for a TimePickerDialog.OnTimeSetListener that will be called when the user sets a new time
7. Add the onCreateDialog(int) callback method:
8. Run the application

Create a Custom Dialog

- If you want a customized design for a dialog, you can create your own layout for the dialog window with layout and widget elements
- After you've defined your layout, pass the root View object or layout resource ID to setContentView(View).
- For example, to create the dialog:

1. Create an XML layout saved as custom_dialog.xml
2. Set the above layout as the dialog's content view and define the content for the ImageView and TextView elements
3. Show the dialog as described in Showing A Dialog.

Create a Custom Dialog

- Here's an example, creating a custom layout in an AlertDialog:

```
AlertDialog.Builder builder;
AlertDialog alertDialog;

Context mContext = getApplicationContext();
LayoutInflater inflater = (LayoutInflater) mContext.getSystemService(LAYOUT_INFLATER_SERVICE);
View layout = inflater.inflate(R.layout.custom_dialog, (ViewGroup)findViewById(R.id.layout_root));

TextView text = (TextView) layout.findViewById(R.id.text);
text.setText("Hello, this is a custom dialog!");
ImageView image = (ImageView) layout.findViewById(R.id.image);
image.setImageResource(R.drawable.android);

builder = new AlertDialog.Builder(mContext);
builder.setView(layout);
alertDialog = builder.create();
```

Summary

- Android menus:
  - Options menu
  - Expanded menu
  - Context menu
  - Submenus
- Dialogs:
  - AlertDialog
  - ProgressDialog
  - DatePickerDialog
  - TimePickerDialog
  - Custom Dialog