

Mobile Applications (420-P84-AB)

John Abbott College

Network Communication

Aakash Malhotra

Create an Options Menu

The options menu is where you should include actions and other options that are relevant to the current activity context, such as "Search," "Compose email," and "Settings."

Where the items in your options menu appear on the screen depends on the version for which you've developed your application:

- If you've developed your application for **Android 2.3.x (API level 10) or lower**, the contents of your options menu appear at the bottom of the screen when the user presses the *Menu* button, as shown in figure 1. When opened, the first visible portion is the icon menu, which holds up to six menu items. If your menu includes more than six items, Android places the sixth item and the rest into the overflow menu, which the user can open by selecting *More*.

Create an Options Menu

- If you've developed your application for **Android 3.0 (API level 11) and higher**, items from the options menu are available in the app bar. By default, the system places all items in the action overflow, which the user can reveal with the action overflow icon on the right side of the app bar (or by pressing the device *Menu* button, if available). To enable quick access to important actions, you can promote a few items to appear in the app bar by adding `android:showAsAction="ifRoom"` to the corresponding `<item>` elements (see figure 2).

Create an Options Menu

To specify the options menu for an activity, override [`onCreateOptionsMenu\(\)`](#) (fragments provide their own [`onCreateOptionsMenu\(\)`](#) callback). In this method, you can inflate your menu resource ([`defined in XML`](#)) into the [`Menu`](#) provided in the callback. For example:

```
@Override
```

```
public boolean onCreateOptionsMenu(Menu menu) {
```

```
    MenuInflater inflater = getMenuInflater\(\);
```

```
    inflater.inflate(R.menu.game_menu, menu);
```

```
    return true;
```

```
}
```

Example:

MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int itemId = item.getItemId();
        if(itemId == R.id.getStudentAction) {
            Log.i( tag: "Main Activity", msg: "Item Selected");
        }
        else {
            return super.onOptionsItemSelected(item);
        }
        return true;
    }
}
```

Example

res > menu > main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <item android:id="@+id/getStudentAction"
        android:orderInCategory="1"
        android:title="@string/search_student">
        
    </item>
</menu>
```

Handling Click Events

When the user selects an item from the options menu (including action items in the app bar), the system calls your activity's [`onOptionsItemSelected\(\)`](#) method. This method passes the [`MenuItem`](#) selected. You can identify the item by calling [`getItemId\(\)`](#), which returns the unique ID for the menu item (defined by the `android:id` attribute in the menu resource or with an integer given to the [`add\(\)`](#) method). You can match this ID against known menu items to perform the appropriate action. For example:

Handling Click Events

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


Handling Click Events

When you successfully handle a menu item, return `true`. If you don't handle the menu item, you should call the superclass implementation of [`onOptionsItemSelected\(\)`](#) (the default implementation returns `false`).

Task

- Create an application with 2 menu options
 - Get Random Student ID
 - Get Random Student with information (Name, studentId, address)

Handling Click Event using onclick

Android 3.0 adds the ability for you to define the on-click behavior for a menu item in XML, using the **android:onClick** attribute. The value for the attribute must be the name of a method defined by the activity using the menu. The method must be public and accept a single MenuItem parameter—when the system calls this method, it passes the menu item selected.

Task: Convert previous program to use **android:onClick** attribute

Handling Click Event using onclick

```
<item
```

```
.....
```

```
android:onclick="functionToBeCalled"
```

```
.....>
```

Async Programming and Keeping your App Responsive

<https://developer.android.com/training/articles/perf-anr>

AsyncTask

AsyncTask enables proper and easy use of the UI thread. This class allows you to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.

AsyncTask is designed to be a helper class around [Thread](#) and [Handler](#) and does not constitute a generic threading framework. AsyncTasks should ideally be used for short operations (a few seconds at the most.) If you need to keep threads running for long periods of time, it is highly recommended you use the various APIs provided by the `java.util.concurrent` package such as [Executor](#), [ThreadPoolExecutor](#) and [FutureTask](#).

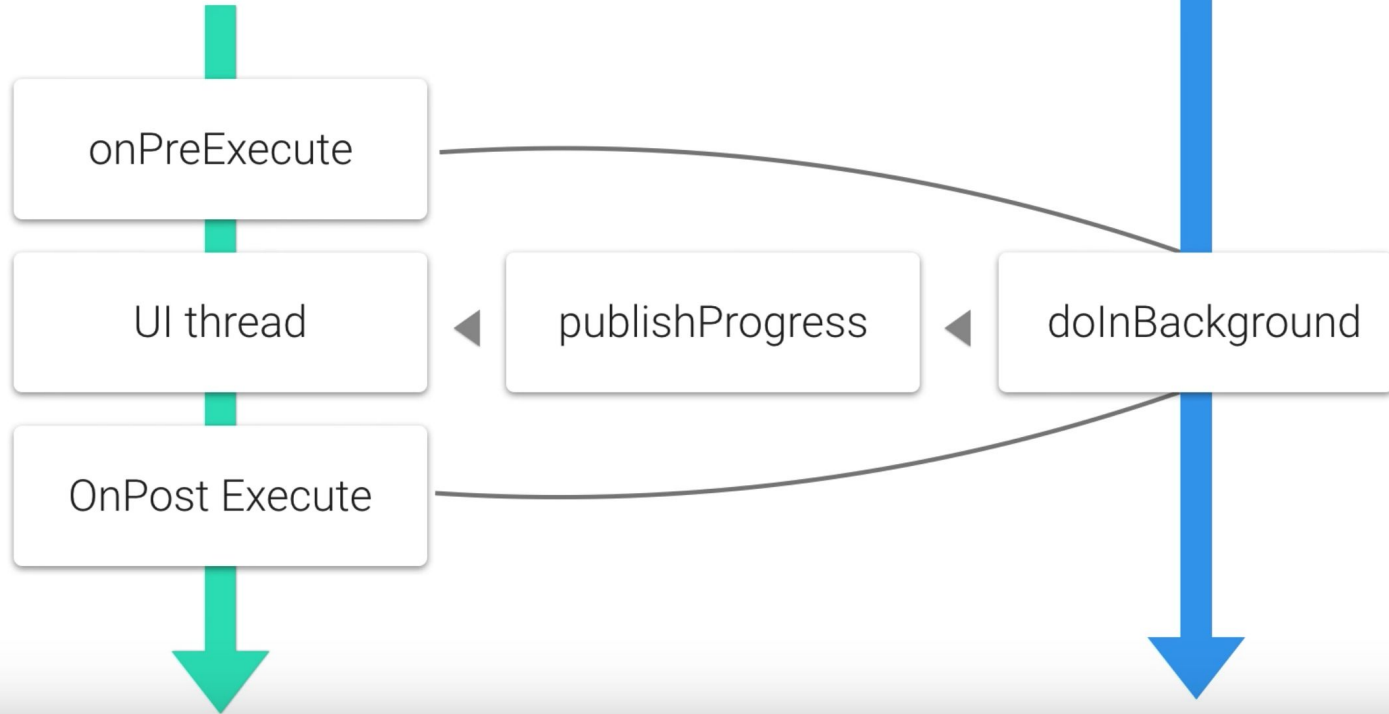
An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread. An asynchronous task is defined by 3 generic types, called `Params`, `Progress` and `Result`, and 4 steps, called `onPreExecute`, `doInBackground`, `onProgressUpdate` and `onPostExecute`.

AsyncTask

AsyncTask must be subclassed to be used. The subclass will override at least one method ([`doInBackground\(Params...\)`](#)), and most often will override a second one ([`onPostExecute\(Result\)`](#).)

Main/UI thread

Background thread



Task:

Get Json from a REST API and display it on the app

<https://jsonplaceholder.typicode.com/> may be used to fake the rest API

Task:

Get values from the received JSON object and show selected value to user

Task:

Use pre execute and post execute hooks to show loading of data

Task:

Create an app with 3 menu items:

- List of all students (atleast 30)
- Get random studentID
- Get random studentID with information

All the data is to be received from the backend

References:

- <https://community.oracle.com/blogs/pat/2004/10/23/stupid-scanner-tricks>
- <https://developer.android.com/reference/android/os/AsyncTask>