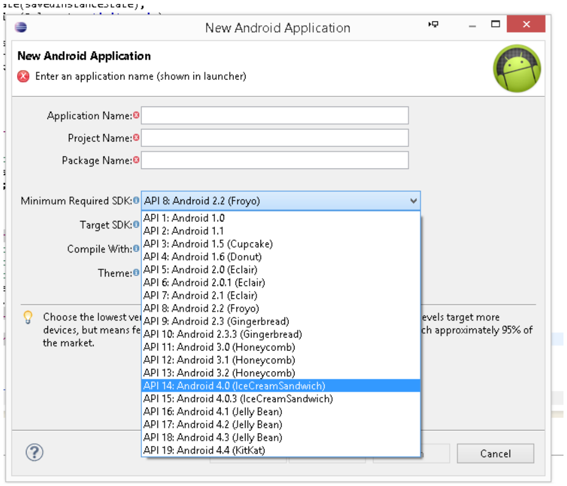
March 2014 - Changes in the Android Developer Tools (ADT)

The newest version of the ADT creates a skeleton Android project that's quite different from the project described in my book. The new project has methods for managing fragments. You can bring your this project back to the projects that I describe in my book by following these steps:

First, when creating a new project, choose a Minimum Required SDK with API Level 14 or greater:



New Android Application

Enter an application name (shown in launcher)

Application Name:

Project Name:

Package Name:

Minimum Required SDK: API 8: Android 2.2 (Froyo)

Target SDK:

Compile With:

Theme:

API 1: Android 1.0
API 2: Android 1.1
API 3: Android 1.5 (Cupcake)
API 4: Android 1.6 (Donut)
API 5: Android 2.0 (Eclair)
API 6: Android 2.0.1 (Eclair)
API 7: Android 2.1 (Eclair)
API 8: Android 2.2 (Froyo)
API 9: Android 2.3 (Gingerbread)
API 10: Android 2.3.3 (Gingerbread)
API 11: Android 3.0 (Honeycomb)
API 12: Android 3.1 (Honeycomb)
API 13: Android 3.2 (Honeycomb)
API 14: Android 4.0 (IceCreamSandwich)
API 15: Android 4.0.3 (IceCreamSandwich)
API 16: Android 4.1 (Jelly Bean)
API 17: Android 4.2 (Jelly Bean)
API 18: Android 4.3 (Jelly Bean)
API 19: Android 4.4 (KitKat)

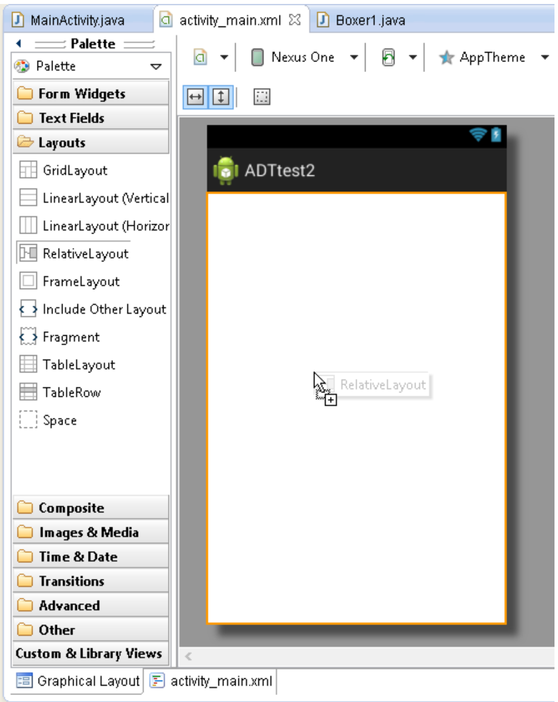
Choose the lowest ve evels target more
devices, but means fe ch approximately 95% of
the market.

Next, after the project has been created, open MainActivity.java in the editor and delete (or at least comment out) all the code that isn't in my book's examples:

```
package com.example.adttest2;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;

public class MainActivity extends Activity {

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

//    if (savedInstanceState == null) {
//      getFragmentManager().beginTransaction()
//          .add(R.id.container, new PlaceholderFragment()).commit();
//    }
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {

    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
  }
//  @Override
//  public boolean onOptionsItemSelected(MenuItem item) {
//    // Handle action bar item clicks here. The action bar will
//    // automatically handle clicks on the Home/Up button, so long
//    // as you specify a parent activity in AndroidManifest.xml.
//    int id = item.getItemId();
//    if (id == R.id.action_settings) {
//      return true;
//    }
//    return super.onOptionsItemSelected(item);
//  }
//
//  /**
//   * A placeholder fragment containing a simple view.
//   */
//  public static class PlaceholderFragment extends Fragment {
//
//    public PlaceholderFragment() {
//    }
//
//    @Override
//    public View onCreateView(LayoutInflater inflater, ViewGroup container,
//        Bundle savedInstanceState) {
//      View rootView = inflater
//          .inflate(R.layout.fragment_main, container, false);
//      return rootView;
//    }
//  }

}
```

(You might also want to select Source→Organize Imports to get rid of the now-unnecessary import declarations.)

Finally, in activity_main.xml, the new layout manager is a FrameLayout by default. To change back to a RelativeLayout or a LinearLayout (both of which I use in the book), go to the Graphical Layout view of the activity_main.xml file. Then drag a RelativeLayout or LinearLayout from the Layouts section of the Palette onto the preview of your activity. (See the next figure.)



If you have questions, send me an e-mail.