Java programming (including 📱 Android), under 💻 Eclipse or Android Studio

Based on (1) GRCC CO117 Java 2007-09-12; and Grand Rapids Java User's Group GRJUG events (2) "Intro to Eclipse" by Carlus Henry, SageTech LLC 2007-02-20, and (3) "Android Xperience" 2013-06-18/10-15 www.meetup.com/gr-jug/events. For date this file last updated, please see page footer. For information on green or other programming subjects, please see a list of this document's **sister docs**.

Contents

Android Studio
Launch
If you get "Error launching Android Studio" "No JVM installation found. Please install a 64-bit JDK. If you already have a JDK installed, define a JAVA_HOME variable in Computer > System Properties > System Settings > Environmental Variables.", then do so from https://google.com/search?q=jdk.
C:\Program Files\Java\jdk1.7.0_51\
JDK info http://docs.oracle.com/javase/
https://plus.google.com/s/gr-jug

New Project…

2014-07-15 uninstalling Android Studio 0.4.2, so I can install new 0.8.2.
It got rid of everything under C:\Apps\Android\android-studio.
Deleted Eclipse from C:\Apps\EclipseAndroidDeveloperToolsADT. Now up to 100 Gb free.

Need to install Android Studio 0.8.2. from http://developer.android.com/sdk/installing/studio.html#download and http://developer.android.com/sdk/installing/index.html?pkg=studio and http://developer.android.com/sdk/installing/adding-packages.html.

Done.
Put in C:\Users\EricPiehl\AppData\Local\Android\android-studio\sdk instead of C:\Apps\Android\android-studio\sdk.

Do Help > Check for Updates.
Do button SKD Manager.

1 Architecture

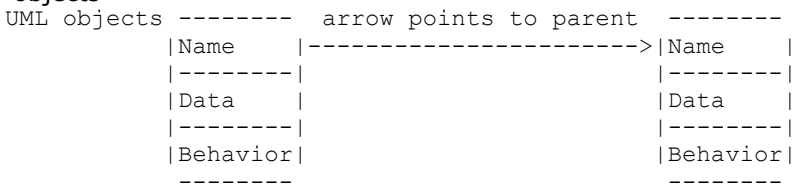
OSGi Open Services Gateway interface
Bundle providers
Innoopract
IBM
BEA
Actuate
...

2 Ideas

Eclipse—the greatest **Java IDE** of all time.
"Java in a Nutshell", by O'Reilly is paper Javadoc.
A Brief History of Java and JDBC.

Reference = pointer.
Methods = member functions, or routines.

Workspace
Projects
classes
files
objects



"Has-A" = Composition Relationship.
Needs more data to go from one class to the other.
Hierarchical, but not in Generalization-Specialization Relationship.
`Car ---- SteeringWheel.`
Side-by-side, no arrow, in UML.
If your class "has-a"nother class, implement by instantiating an object of that other class.

"May-be-associated-with" = Aggregation Relationship. Fuzzier than Composition Relationship.
`Ponds ----- Ducks.`

"Is-like-a" = Substitution Relationship.
DVD "is-like-a" VideoTape.

Could both extend a Player class.

"Is-A" = Generalization-Specialization Relationship = Inheritance Relationship = Substitution. Polymorphism. Overriding. Subtyping.
Hierarchical.
`Circle ----> Shape.`
Parent over the other, with arrow to the parent, in UML.
Helps reuse code with little or no mods.
If your class "is-a"nother class, implement by child class "extends" the parent class.

```
upcast    Primitive Types   downcast
 A        ---------------      |lossy
 |not                         |
 |lossy*      double          |allowed
 |             float          |only by
 |allowed       int           |casting
 |by Java      bool           V
```

* = more room to store all information in lower types, and more.

```
 upcast       Complex Types, ancestor class       downcast
  A          ---------------------------------      |allows use of override**
  |not           door                shape          |
  |lossy*         A                    A           |allowed
  |             __|______            ___|____       |only by
  |allowed     |         |          |        |      |casting
  |by Java  slider securityDoor  circle rectangle   V
 Is-A                    derived class            Inherit
```

* = keeps the attributes and methods of the child type even if stored in a parent type object.
** allows use of override methods and non-applicable attributes?

3 Install Eclipse (or other IDE listed in Tools)

http://cs.calvin.edu/curriculum/cs/112/resources/installingEclipse/videos/ nice **video instructions**, thanks to Calvin College Prof. **Joel Adams**.

Better for professionals—put only what you need in each Workspace
http://download.eclipse.org/eclipse/downloads/ > Latest Release > Platform Runtime Binary > *yourOS*, etc.
-OR-
Easy—all-at-once www.eclipse.org > Download > tab Packages > Eclipse IDE for Java [EE] Developers > *yourOS*, etc. *[Was Europa 3.2 when I wrote this. Later Helios 3.6.1. Now 4.2.1.]*
-OR-

3.1 Android

If you want to do Android development, download the **Android SDK with Android Developer Tools ADT** Bundled for Windows http://developer.android.com/sdk/ > "Download the Android SDK ADT Bundle for Windows".
-OR-
Install the new **Android Studio** from http://developer.android.com/sdk/installing/studio.html, but as of 2013-06-18, the Android Studio is said to be "Early Access Release", have bugs, and crash every 15 minutes.

Follow other Android prerequisites from https://groups.google.com/forum/#!topic/grand-rapids-java-user-group/eIwrk4nXdnE:

- Follow other installation steps from "C:\Users\EricPiehl\Documents\EricPiehlEdu\EclipseAndroidDeveloperToolsADT\Quick Installation Guide for ADT - Google Drive.pdf".
  - "Unpack the files into a directory and take note of the path to the SDK.
  - Open Eclipse.
  - Install **Eclipse Plugin:**
    - **Help** > **Install New Software**.

        - Click the **Add** button in the upper right corner.
        - Enter Name=**ADT**, and Location=https://dl-ssl.google.com/android/eclipse/, and click OK.
        - Next check the checkbox next to **Developer Tools** and click **Next** > **Next** > accept terms > **Finish**.
        - **Restart** Eclipse. When Eclipse opens you will be asked to point to the SDK.
        - Select Use Existing SDKs, and navigate to the path where the SDK was unpacked (above).
    - **Configure Virtual Device**
      In order to run your Android Applications locally; you are going to need an emulator for an Android Device. Here are the steps necessary to configure the emulator that we will be using.
        - Eclipse > Window > **Android Virtual Device Manager**.
        - Click **New** to add a new Virtual Device.
        - If you don't have something better, set AVD Name=Nexus_One, Device=Nexus One (3.7", 480 x 800 hdpi), Target: Android 4.2.2 – API Level 17, Hardware keyboard present, Display a skin with hardware controls, RAM=512, VM Heap=32, Internal Storage=200 MiB, and select OK.
    - **Check Environment**
      Let's verify that we have everything working by running one of the example Android applications from Google. Once this has been verified, you are all set to go.
      Install Samples:
        - Eclipse > Window > **Android SDK Manager**.
        - Select for your "Android v.v.v" of interest, "**SDK Platform**" and "**Samples for SDK**".
        - Select "Extras" "**Google USB Driver**" and anything else you need.
        - Click button **Install packages** > **Accept License** > **Install**.
        - When installation is complete, you can view the samples under path `<sdk>/samples/android<version>/` (for me, "`C:\Apps\EclipseAndroidDeveloperToolsADT\sdk\samples\android<version>`").
    - **Run ContactManager Sample**
      There are many sample projects that come with Samples for SDK. I chose ContactManager. The steps below will walk you through importing a project, start the Virtual Device, and run an Android Application.
        - From Eclipse right click in **Package Explorer** > **Import...**.
        - **Android** > **Existing Android Code into workspace** > **Next**.
        - The samples folder is installed under sdk. Open the samples folder "`C:\Apps\EclipseAndroidDeveloperToolsADT\sdk\samples\android<version>`" > ContactManger > OK > select > Finish.
        - Compile and run as below.

3.2 Other

http://download.**eclipse**.org/**webtools**/downloads/
http://code.**google**.com/**webtoolkit**/ GWT Google web toolkit
www.csszengarden.com
web page debugging Firebug (see Tools)
source control Subversion or Git (see Tools)
www.springsource.com GUI

Javatester, including a nice tester of which Java version your browser is using.
Another Java tester. Old Java version uninstall tool.

4 Run Eclipse

Launch **Eclipse.exe**:

- **Android Studio** is at "file:///C:\Apps\Android\android-studio\bin\studio64.exe".
  -OR-
- A long time ago, I put Eclipse somewhere else. For me after reinstall mid-2013, Eclipse is at "file:///C:\Apps\EclipseAndroidDeveloperToolsADT\eclipse\eclipse.exe".
  - ***TODO:*** *Actually, that doesn't work, but "file:///C:\Apps\EclipseAndroidDeveloperToolsADT\SDK Manager.exe" does.*

If portable, put **Workspace** on a thumb-/jump-/USB-/flash-drive F:\eclipse\workspace.
Else maybe %USERPROFILE%\Documents\GRLUG\eclipse\workspace. (For me after reinstall mid-2013, is at "file:///C:\Users\EricPiehl\Documents\EricPiehlEdu\EclipseAndroidDeveloperToolsADT".)

Periodically (monthly?) **update** at:

- Help > **Check for Updates**.
- Eclipse > Window > Android **SDK Manager** > check items with Status="Update available:…" > "Install n packages…".

4.1 Android

Start your Virtual Device via Eclipse > **Window** > **Android Virtual Device Manager**.
Highlight the Android Virtual Device you created earlier, click button **Start...** on the right > **Launch**.
Do it now—you will need it later, and it takes forever to initialize itself.
If problems, look at Eclipse pane **Console**, or "**LogCat**" for application errors.

4.2 All

Save a test Perspective.
If delete a frame, bring back with Window > Show View. Can bring back to default, with Window > Reset Perspective.
Something like:

```
Package Explorer   Text Editor   Outline    Help
 Hierarchy                        Ant

Problems
 Javadoc
  Declaration
   Console
```

Put `Javadoc` in SE pane.

To change Eclipse Options so that Ctrl-Shift-F (right-click Source > Format) will put my "{" on a new line, set Window > Preferences > Java > Code Style > Formatter > New Profile > Edit… > Braces tab > set every dropdown box to Next line > correct settings on tab White Space, Blank Lines, New Lines, Control Statements, Line Wrapping and Comments > OK > OK.

5 Create a new Java Project

Either here, or Import whole tree as in two sections down.

In pane **Package Explorer**, [right-click|File >] New > Project...
In the **New Project...** dialog, select:

- **Java** Project (your choice on ordinary, or "from Existing Ant Buildfile"), or
- **Android Project** (your choice on "Application", "Sample" or "from Existing Code").

In the New <projectType> Project dialog, give it:

Project name: in **`UpperCamel`**, e.g., ***`YourProject`*** or **`PiehlAssignementZero`**.
Create new project in workspace.
Contents: Use default JRE
Project Layout: Create separate folders for sources and class files.
Working set: no Add project to working set.
In Android, uncheck Create customer launder, check Create Activity.
`Finish`.
Use **Package Explorer** to open and manage all project files.

When Create Activity, blank Activity, name=***`YourProject`***.

```
package com.ericpiehl.co117.assigmentzero
```

```
Packages com.polarseal.database.window v. com.polarseal.gui.window
```

6 To find Classes

www.gotapi.com for HTML, SQL, etc.
www.javapolis.com.
http://jakarta.apache.org/commons for http client, net, I/O, database, JDBC html CCS.
www.google.com/codesearch
www.archive.org

7 To import code or library (e.g., .jar file) into workspace /lib folder

7.1 Non-Android?

1. In the Package Explorer pane, click on ***`yourProject`***, and select File > New > Folder > lib.
2. Preferably in Navigator pane, drop in the .jar file from wherever into the workspace/project/lib folder.

If you didn't do the above in the Navigator pane, F5 or Right-click project > Refresh.

3. Back in the Package Explorer pane, right-click on the .jar > Build Path > Add to Build Path.

Note in the Package Explorer pane, that the .jar moved from the lib folder to Referenced Library.

4. Right-click on the .jar > Properties > Javadoc Location > select "Javadoc in archive" > select "Workspace file" > Archive path Browse... to the .jar > OK > Path within Archive Browse... > select the documentation, which is usually the /doc folder > OK > Validate > OK > OK.

Javadocs will now appear when a class, object or method is selected or when you hover.
Window > ShowView > Javadoc: the Javadoc view also works with this and gives a bit more room to view the documentation and the source.

5. If the jar has source, right-click on the .jar > Properties > Java Source Attachment > Workspace... > select the jar > OK > OK.

6? Right-click on the .jar > "Add to Classpath"? *<DataAbstraction.ppt sl13 Finding Classes.*

7.2 Android

To use Sample that came from Android:

- From Eclipse right click in **Package Explorer** > **Import...**.
- **Android** > **Existing Android Code into workspace** > **Next**.

- The samples folder is installed under sdk. Open the samples folder "**C:\Apps\EclipseAndroidDeveloperToolsADT\sdk\samples\android<version>**" > ***yourSample*** > OK > select > Finish.

If you want code or samples from **GitHub**:

- **One-time**: Install **GitHub Desktop** https://desktop.github.com/. [I did so on 2018-02-26 and 2016-10-02 (different machines). I installed the older **GitHub for Windows** http://windows.github.com/ on 2016-04-21 and 2013-08-20 (different machines).]
- For each project's **GitHub repository**:
  - **GRJUG** at https://github.com/grjug such as week one "Favorite Colors",
  - **iNaturalist** at https://github.com/inaturalist repertories for Rails, Android, iOS, etc.,
  - or in general, https://github.com/ . . .
- ~~**Fork**~~ > **Clone or download** > **Open in Desktop** ~~**Clone in Desktop**~~ > wait > Save prj to your computer and use it in GitHub Desktop.
  - Hover over the repository to see where it is.
  - In Android Studio or Eclipse, **Import** Project? that at its internal .gradle file.
  - Build it and run it.
- If you want the Master branch (tip-of-the-tree):
  - Drag the repository blue-on-white name to your **Start** > **Github, Inc.** > **Github** > little icon "drag a repository here to add". [I did that, then copied to "***C:\Users\EricPiehl\Documents\EricPiehlEdu\EclipseAndroidDeveloperToolsADT\YourProject_GITHUB_GRJUG***".]
- If you want another branch, click "releases", chose the release you want, then download the `zip` or `tar.gz` version, unzip it yourself, then Eclipse > File > **Import**.

I got project **DroidSheep** v36 via:

- **One-time**: Install SVN via Tools,
- http://droidsheep.de > https://code.google.com/p/droidsheep/source/checkout "***svn checkout http://droidsheep.googlecode.com/svn/trunk/ droidsheep-read-only***" > copied to "***C:\Users\EricPiehl\Documents\EricPiehlEdu\EclipseAndroidDeveloperToolsADT\droidsheep-read-only_DROIDSHEEP_DE***".

To get Android assets, such as icons, see http://android-ui-utils.googlecode.com/ or Android Asset Studio - Icon Generator - Launcher http://android-ui-utils.googlecode.com/hg/asset-studio/dist/icons-launcher.html#foreground.space.trim=1&foreground.space.pad=0&foreColor=33b5e5%2C0&crop=0&backgroundShape=bevel&backColor=ffffff%2C100.

8 To create a Class

In the Package Explorer panel, src folder, [right-click|File >] New > Class.
Package #package "com.ericpiehl.project" or "edu.grcc.co117.examples"
Name=***YourProject***.
Select Public as the modifier.
The superclass should be **java.lang.Object** by default (usually, exceptions below).
Select **public static void main(String args[])** as a method stub to create.
If you want to create a constructor stub in the code, select "Constructors from superclass".
Select Generate comments.

Create new **Application Class** via File > New > Java Class > Name=***YourProject*****Application** and Superclass=**android.app.Application** > Finish.

Leave all other fields blank.
Click all checkboxes > Finish.

```
#import
public class YourProject {
 public static void Main() {
 }}
```

Use Package Explorer > package > Properties > Java Build Path > Source tab, put "eclipse" in output folder somewhere to remove confusion.
Update **main()**, by…

9 To hack your source file

Example string array:

```
String[] myColors = new String[]{"Blue", "Red", "Yellow", "Orange"};
```

Not `String Array:String[]` ?

If you see wavy red lines under something:

- Hover > choose an **Import** option.
- If that doesn't work, see lots of stuff in last subsection in this section.

9.1 External Strings

Externalize your strings in *yourProject*`\res\values\strings.xml`, by adding XML element `string` with attribute `name`, such as `<string name="myColors">Red Yellow Green Blue</string>`.

9.2 To add an Activity (Android screen)

An **Activity** is a combination of a **Class** and a **Layout**.
Roughly, a **screen showing right now** is a combination of an **instantiation of an Activity class**, and a **Layout**.

If you created an Activity while creating the project, set tbd=`blank`, name=*yourActivity*. Then:

- To *yourProject*`\src\`*packageValueFromRootAndroidManifestXmlWithSlashesInsteadOfDots*`\`*yourActivity*`.java`, verify that it includes:

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.yourActivity);
        ListView lv = (ListView) findViewById(R.id.yourProjectListView);
        String[] myColors = new String[]{"Blue", "Red", "Yellow", "Orange"};
        ListAdapter fcAdapter = new ArrayAdapter<String>(this,
                R.layout.yourActivity_list_item, myColors);
        lv.setAdapter(fcAdapter);
    }
```

- Clone *yourProject*`\src\`*packageValueFromAboveWithSlashesInsteadOfDots*`\`*yourExistingActivity*`.java`, to *yourNewActivity*`.java`.
- Edit its class name from *YourExistingActivity* to *YourNewActivity*.

*[For week one 2013-06-18 Android Xperience ListView Challenge, this was called **FavoriteColor**. At first, I misspelled FavoriateColor, but later refactored.]*

To add items to screen layout, add one or more Controls to the screen in *yourProject*`\res\layout\activity_`*yourProject*`.xml`:, inside tag `RelativeLayout`, add one or more of:

-
```
<ListView android:id="@+id/yourProjectListView"
          android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:layout_alignParentLeft="true"
           android:layout_alignParentTop="true" />
```

-
```
<TextView android:id="@+id/textView1"
          android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="@string/hello_world" />
```

-
```
<TextView android:id="@+id/textView2"
          android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_alignLeft="@+id/textView1"
           android:layout_below="@+id/textView1"
           android:layout_marginTop="34dp"
           android:text="@string/myColors" />
```

- ```
  <EditText android:id="@+id/editText1"
            android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:layout_alignParentLeft="true"
             android:layout_below="@+id/textView2"
             android:layout_marginTop="36dp"
             android:ems="10" />
  ```

Add a list view via Create Android XML file.
~~Add Action Bar menu.~~
Folder src has .java code
Folder res\layout has .xml files
AndroidManifest.xml
ListView
ListAdapter
RelativeLayout (R.layout.activity_*yourProject*)
TextView

To **create** a new **Activity** (Android screen):

- To *yourProject*\AndroidManifest.xml, within tag application, add tag:
  ```
  <activity android:name="packageValueFromAbove.yourNewActivity"
      android:label="@string/app_name"></activity>
  ```
- Clone *yourProject*\src*packageValueFromAboveWithSlashesInsteadOfDots**yourExistingActivity*.java, to *yourNewActivity.java*.
- Edit its class name from *YourExistingActivity* to *YourNewActivity*.

9.3 To add an Action Bar menu Options button to an Activity (Android screen)

In an **Activity**, **hook up** an **Action Bar** menu **Options button**:

- To *yourProject*\src*packageValueFromAboveWithSlashesInsteadOfDots**yourActivity*.java, within class *YourActivity*, add:
  ```
  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case R.id.menu_myMenu:
              Intent intent = new Intent();
              intent.setClass(this, YourOtherActivity.class);
              startActivity(intent);
              return true;
        default:
              return super.onOptionsItemSelected(item);
        }
  }
  ```

To **show** that **Action Bar** menu **Options button** in an Activity (Android screen):

- To *yourProject*\res\menu*yourActivityMenu*.xml:
  - Change tag *item* attribute *android:showAsAction* from value "*never*" to "*always*".

If you want to **show** another **Options button** in an Activity (Android screen):

- To *yourProject*\res\menu*yourActivityMenu*.xml:
  - Clone tag *item* to another one.
  - Change tag *item* attribute *android:orderInCategory* to 100 more than its source attribute.

If you want to make that **Options button** show a **particular icon** in an Activity (Android screen):

- To *yourProject*\res\menu*yourActivityMenu*.xml:
  - Add tag item attribute android:icon="@drawable/*iconName*",
    where *iconName* is a .png file that exists in folder *yourProject*\res\drawable-size.
  ```

You may have gotten these by dragging a folder tree of these icons from Windows File Explorer over into Eclipse > Package Explorer > folder ***yourProject*\res**.

- ???Change tag *item* attribute **android:id** from value **"@+id/action_settings**" to **"@+id/menu_*myMenu*"**???

Shift-Ctrl-R = open a **Resource** via patter, e.g., "**.manifest**" or "**strings**".

TODO: *Is this supposed to be useful?* `yourProject\res\values\strings.xml` *?*

Shift-Ctrl-C = see the activity.

9.4 To externalize control off to a Service, to survive Activity destroy/create

Create new **Service Class** via File > New > Java Class > Name=***YourProject*Service** (default Superclass) > Finish.
Populate it with (in **green**):

```
public class YourProjectService {
  private List<String> colors = new ArrayList<String>();  //this needs to survive
  public YourProjectService() { //need a constructor
    colors.add("Yellow");
    colors.add("Green");
    colors.add("Red");
    colors.add("Blue");}  //initial list--more will be added at runtime by addColor()
  public List<String> getColors() {return colors;}  //getter
  public void addColor(String newColor) {colors.add(newColor);} //add at runtime
}
```

Create new **Application Class** via File > New > Java Class > Name=***YourProject*Application** and Superclass=**android.app.Application** > Finish.
Populate it with (in **green**):

```
public class YourProjectApplication extends Application {
  private YourProjectService service = new YourProjectService(); //need survive
  public String getStatusMsg() {return "in getStatusMsg()";}
  public YourProjectService getService() {return service;}  }
```

Hook up the above: select project root **AndroidManifest.xml** (if it looks like a webform, click lower-tab to show in XML format), to element **application**, add attribute "**android:name=".*YourProject*Application"**", as:

```
<application android:allowBackup="true"
             android:icon="@drawable/ic_launcher"
             android:label="@string/app_name"
             android:theme="@android:style/Theme.Holo.Light"
             android:name=".YourProjectApplication">
  <activity android:name="com.example.yourProject.YourActivity"
            android:label="@string/app_name">
```

In ***YourActivity*.java**, populate it with (in **green**):

```
public class YourActivity extends Activity {
  ArrayAdapter<String> fcAdapter; // moved here to survive Activity delete/recreate
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // get Service, so this app's data can survive Activity delete/recreate
    Application application = getApplication(); // get app > my app >
    YourProjectApplication thisApp = (YourProjectApplication) application;
    String msg = thisApp.getStatusMsg(); // optional msg and
    Log.w("in YourActivity()", msg); // optional logging >
    YourProjectService service = thisApp.getService(); // Service.
    setContentView(R.layout.yourActivity);
    ListView lv = (ListView) findViewById(R.id.yourProjectListView);
    // get these from Service instead of being hardcoded
    // String[] yourProject = new String[] {"Blue", "Red", "Yellow","Orange"};
```

```
      List<String> yourProject = service.getColors();
      ListAdapter fcAdapter = new ListAdapter ArrayAdapter<String>(this,
         R.layout.yourActivity_list_item, myColors);
      lv.setAdapter(fcAdapter);
    }
  ...
    @Override
    // added so Activity can be resurrected by Service
    protected              void              onResume()              {super.onResume();
      fcAdapter.notifyDataSetChanged();}   }
```

In NewColorActivity.java, populate it with (in green):

```
  public class NewColorActivity extends Activity {
    EditText txtColorName;
    YourProjectService service; //get Service to survive Activity create/recreate
    @Override
    protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.yourActivity);
      // get Service, so this app's data can survive Activity delete/recreate
      Application application = getApplication(); // get app > my app >
      YourProjectApplication thisApp = (YourProjectApplication) application;
      //String msg = thisApp.getStatusMsg(); // optional msg and
      //Log.w("in NewColorActivity()", msg); // optional logging >
      service = thisApp.getService(); // Service.
      txtColorName = (EditText) this.findViewById(R.id.txtFirstName);
      txtColorName.requestFocus();}
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if it is present.
      getMenuInflater().inflate(R.menu.yourActivityMenu, menu);
      return true;}
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
      switch (item.getItemId()) {
      case R.id.menu_myMenu:
        String newColor = txtColorName.getText().toString(); //get string typed by user
        service.addColor(newColor); // and add it to the list
        NewColorActivity.this.finish();
        return true;
      default:
        return super.onOptionsItemSelected(item);}   }
```

9.5 To separate Service off to frontend/backend Model–View–Controller MVC

TODO: fill this in from Compare of C:\Users\EricPiehl\Downloads\FavoriteColors-1.1 v C:\Users\EricPiehl\Downloads\FavoriteColors-1.2.

9.6 All

--package (to move a package, right-click > Refactor > Move package).
--import has an uppercase Class name at the end.
--Javadocs
--Declaration

"**.**" is the member operator.
To see methods, type "***object.***"
***object*.clone** method copies the contents.
object* = *object makes the other pointer point to the **same** object.

Ctrl-space = "help me": expand Sysout, etc.

Ctrl-1 = "fix it": show lots of stuff.
Ctrl-/ = comment out or de-comment out.
Ctrl-Shift-F = Format, auto-indent source.
Ctrl-2 undo = undo something real fancy.
Ctrl-hover = goes to that routine.
Toggle markers and **Ctrl-K** helps finds.
Help > Tips and tricks.
templates.

If you see red wavy lines under something, hover > choose an **Import** option.

If your call "***cannot be resolved***", do a **Ctrl-Shift-O** or click Source > **Organize Imports** to organize your Imports.

If having **weird troubles**, do a **Clean**.

Use `Math.PI`.

10 To compile or run your app

10.1 Non-Android

Alt-Shift-X, J run as Java App.
-OR-
Right-click source > **Run As** > **Java App**.

10.2 Android

If you haven't done so earlier:

- Start your Virtual Device via Eclipse > **Window** > **Android Virtual Device Manager**.
- Highlight the Android Virtual Device you created earlier, click button **Start...**on right > **Launch**.
- If problems, look at pane **Console**, or "**LogCat**" for application errors.

Once the VD has started…

In Package Explorer:

- select ***yourProject***, and **Ctrl-F11** or click the green arrow **Run**, or
- right-click ***yourProject*** > **Run as** > **Android Application**.

yourProject may start, or just pop up under Applications.

If **bad**, see errors in panes **Problems** or **Console**.

If you see build errors, right click ***yourProject*** > **Android Tools** > **Fix Project Properties**, and retry.

If you see on pane **Console**:
The connection to adb is down, and a severe error has occurred.
You must restart adb and Eclipse.
Please ensure that adb is correctly located at 'path\sdk\platform-tools\adb.exe' and can be executed.
then reread launching the Android Virtual Device window you should have done earlier.

If you see on pane **Console** "ActivityManager: Warning: Activity not started, its current task has been brought to the front", then exit your app on the emulator, and retry.

If you get compile errors related to file (`R.class` or `R.java`?), do a clean & build within Eclipse. Never check in file (`R.class` or `R.java`?) into a version control system—it is different for everyone. Nor check in folders `bin`, `gen`, nor `assets`.

10.3 Build

Ant > Internal ant.

11 To debug your app

***TODO**: write this part.*

More in Tools section Testing.

12 To beautify—make it look pretty

In each source file:
Click **Source** > **Clean Up...** will fix lots of optional stuff (Code Style tab: Use block in if/while/for/do statements=Always, Use parentheses around conditions=Always, Use modifier 'final' where possible=Always; Member Access tab: Use 'this' qualifier for method accesses=Always, Use declaring class as qualifier, Change all access through subtypes (indirect accesses), Change all access through instances; Unnecessary Code tab: Remove unnecessary casts, Remove unnecessary '$NON-SLS$' tabs).

Ctrl-Shift-O or
click Source > **Organize Imports** will organize your **Import**s.

Ctrl-Shift-F Format or
click Source > **Format** will auto-indent source, and make it look pretty.

13 To create Javadoc

Reruns can just right-click **`doc\javadoc.xml`** > Open Javadoc Wizard > Next > Next > Open generated index file in browser > Finish.

If first time:
Right-click your project > Export...
In the Export dialog, select Javadoc.
In the Generate Javadoc dialog, give it:
- Javadoc command: your computer's javadoc.exe file.
- Select your project and any subordinate files.
- Private.
- Use Standard Doclet in yourWorkspace***`yourProject`**\doc.
- Next.
- Document title optional, perhaps "Javadoc of Eric Piehl's CO117 application, AssignmentThree."
- Basic Options: select all
- Document these tabs: select all.
- No archives.
- No Style sheet.
- Next.
- No overview.
- No VM options.
- No Extra Javadoc options.
- Save the settings of this Javadoc export as Ant Script: ***`yourWorkspace\yourProject`*****`\javadoc.xml`**.
- Yes Open generated index file in browser.
- Finish.
- If Update Javadoc Location box seems reasonable, click Yes.

If you hit **`Refresh F5`**, you will see **`javadoc.xml`** appear in the Package Explorer pane doc folder.

Wait until Console pane show "<terminated> Javadoc Generation".

In batch: F:\eclipse\workspace\HelloDate>javadoc -classpath . java

14 To publish: create a .jar file

TODO: *add* ***Android*** *section.*
Reruns can just right-click ....jar > Delete, right-click ....jardesc > Create JAR.

If first time:
Refresh F5.

library jar dependent on

Right-click your project > Export...
In the Export dialog, select JAR file.
In the JAR File Specification dialog, give it:
- Select the project source and default package.
- Select Export generated class files and resource.
- No select Export all output folders for checked project.
- Select Export java source files and resources
- No select Export refractoring for checked projects.
- Select the export destination yourWorkspace***yourProject***\PiehlAssignmentZero.jar.
- There is no need to include the .classpath or .project files or any additional library jars (??).
- No compress
- No Add directory entries.
- Click Next.
- Select Export class files with compile errors
- Select Export class files with compile warnings.
- Save the description of this JAR in the workspace ***yourProject***/PiehlAssignmentZero.jardesc.
- Next.

From the JAR Manifest Specification dialog,
- Select Generate the Manifest file and save it in the project as ***yourProject***/MANIFEST.MF. Always use the Browse button to name the file.
- Always select the Main() class using the Browse button.
- Click Finish. The jar file will appear in the project.

To paranoia-check:
- Run the app with:
  `java -classpath <directory location of the jar>/co117Utilities.jar;<directory location of the jar>/<YourLastName>Assignment<Number>.jar <MainClassName>`
- For example:
  `java -classpath C:\Java\co117\code\AssignmentZero\lib\co117Utilities.jar;.\MishAssignmentOne.jar Temperature`

Or view with any zip utility, including the Windows compressed file or 7-Zip.
My favorite:
1. Copy the `.jar` to your favorite trash folder.
2. Rename the `.jar` to `.zip`.
3. Right-click > Extract to All… to somewhere in this trash folder.
4. Drill down this directory to see:
   a. that the source and class files are all there, and
   b. Open the manifest file `META-INF\MANIFEST.MF` in your favorite text editor, and see that it has a `Main-Class` specified.
5. If in a trash folder, leave it, cleaning out the folder occasionally. If live, rename it back to `.jar`.

15 To submit to Instructor

Submit to instructor with BB Course Tools > Digital Dropbox > Add File then Send File.

16 TODO

~~**TODO**: Read discussions at Google+ communities for GR-JUG https://plus.google.com/communities/103314203601881861807 or https://plus.google.com/u/0/communities/103314203601881861807. Set "Notifications on". Done 2013-08-20.~~

TODO: For week one 2013-06-18:

- Watch screencast of Android Xperience round 1: ListView Challenge (Favorite Color) www.youtube.com/watch?v=va9oX5IITtU. Or www.youtube.com/v/va9oX5IITtU.

TODO: For week two 2013-07-16 EE Actionbarexperience on ActionBar and how to add custom menu items:

- See slides https://docs.google.com/presentation/d/11RHs9yKUrFtSZGShMSKRhCd2uwy5-V4CrmUM4ohgzpw/edit?forcehl=1&hl=en#slide=id.p13.
- Watch video www.youtube.com/watch?&v=fhi1_FzIwfo.

TODO: If needed, see "So, You Want to Develop Android Apps? Here's How to Learn" http://makeuseof.com/tag/so-you-want-to-develop-android-apps-heres-how-to-learn/.

At least one other **TODO:** sprinkled above.

TODO: convert other **.doc* to *X*PDF.doc* and **.pdf*, including links from *index.html*:

- Foreach of 6 **.doc* ApacheMySqlPhp.doc, CSharp.doc, Java.doc, syncPalmOutlook64Bit.doc, tools.doc and VBA.doc:
  - rename to *X*PDF.doc*
  - change *.doc* inside to *.pdf*, perhaps with a *www.ericpiehl.com/*
  - style Title
  - Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
  - Save As **.pdf*
- Update *index.html*: lowercase, change `.doc` to `.pdf`.
- FTP up, delete 6 or more old **.doc*s.

-End.- send comments to the author.