Android apps lessons

This is an outline of what I did with my students. I did not take as many notes on the timing I took so I am going on approximate memory :) I meet with my students daily for two and quarter hours all year long. So adjust the timing on these as needed. If you have questions please feel free to contact me. With the presentation and the mini-apps it took about an hour to introduce.

1. MVC reading - I had my students read the Wikipedia entry and then look for an article about MVC in general. We were then able to discuss the idea of this paradigm. (45-60 minutes)

2. Class discussion of apps: good bad and the ugly (25 minutes/homework)
   (a) what are the worst apps out there? Have students search through the app stores if they have a phone with access or there are some good websites rating apps and a few articles on bad apps.
      i. Why were these apps bad?
      ii. How to make them better?
      iii. What apps still need to be made?

3. GoogleLabs AppInventor
   (a) Structure (25 minutes)
      i. Go over the environment, and the basic capabilities
      ii. Widgets and properties
      iii. Blocks editor
   (b) KISS principle for the first version (60 minutes)
      i. Click a button and the screen or picture changes/ start and stop a sound
         A. Change screens by setting a layout to invisible as only one screen is currently possible in AppInventor
      ii. There is a limit of 3 MB per file uploaded

   (c) Have students build from there (3 hours homework)
      i. More complex apps
         A. My class started with at least 3 self defined methods as well as the use of 4 predefined methods from AppInventor
         B. 8 widgets/objects in the code
         C. Interaction with the user
         D. .ico files do NOT work , causing weird errors

4. Using the methodology of creating blocks of code and calling them as needed (15 minutes)
   (a) Organization is key as always
   (b) Grouping of blocks based on type
      i. Methods
      ii. Predefined methods

5. Switch to eclipse environment (30-60 minutes)
   (a) First make a skeleton/bare bones app
   (b) Then adjust to include a button that changes text or time when clicked
      i. Introduce/reuse the idea of a listener object
      ii. Event driven programming
      iii. What are we listening for?

6. Go back to AppInventor (4 hours homework)
   (a) Take your more complex AppInventor app and identify the objects, methods and data member components so it can be recreated in code within eclipse.
   (b) Review this document with either teacher or second level students.
   (c) Rebuild the app using java instead of blocks
   (d) Comparison of code to blocks as a written review

7. Install app to the phone (totally up to you and the methods used)
   (a) Bar code
      i. By far the easiest method to transfer the completed app to your phone
      ii. Just press the button, wait and use a bar code scanner app to install it
      iii. Same idea with eclipse connect .apk file to a QR bar code and it is ready to go.
   (b) APK
      i. Save the file, use the ADB command line and install it. Not hard but DOS based so if there are restrictions on your systems probably not the easiest choice
   (c) USB
      i. Not recommended as it is totally dependent on the driver availability for the phone. Some of the phones do not have easy to access drivers

8. XML Overview:
   (a) Rules for the files
      i. Lowercase file names
      ii. properties in the XML tags
      iii. android:id="@+fdsfsd" vs android:id="fdsfsd"
      iv. Formatting of XML for ease of use
         A. Eclipse settings

9. Designing an interface (30 minutes)
   (a) Basic design elements
   (b) Dragging elements from the pallet onto the screen
   (c) Changing view to the XML side to add specifics to code vs using the properties view
   (d) Separate files for each activity

10. Android JUnit testing (40 minutes)
   (a) Start with the original application
      i. make it fail
      ii. add to the code
      iii. make a new test
      iv. make sure it passes
      v. Repeat...
   (b) The built in testing is not easy to do with complex apps. I need to work on this specifically. If I come up with anything fabulous I will post it.