**Kim**

1. Debug source code

**Response:**

1. Example on how to debug the database application
2. More resources: <http://developer.android.com/tools/debugging/index.html>

**Cesar**

1. A simple drawing app with basic UI components for drawing

**Mason**

1. AWT, Swing background - 2D, Canvas

**Response:**

1. Sample project created in github: **00\_Paint\_2D** ( based on Marakana article paint 2D - <http://marakana.com/tutorials/android/2d-graphics-example.html> )
2. Sample project created in github: **00\_CustomViewDraw\_2D** ( based on official Android API samples )
3. Sample project created in github: **OaiaAnimata** ( based on article from Alin Berce )
4. Please check the following resources:

* Canvas, View, Surface View - <http://developer.android.com/guide/topics/graphics/2d-graphics.html>
* Custom drawing - <http://developer.android.com/training/custom-views/custom-drawing.html> . Sample project taken from the original documentation **00\_CustomViewDraw**
* Additional resource signature signing: <http://corner.squareup.com/2010/07/smooth-signatures.html>
* Improvement smooth draw: <http://stackoverflow.com/questions/8287949/android-how-to-draw-a-smooth-line-following-your-finger>

**Ed**

1. Integrate with images, contacts
2. Image caching

**Kirk**

1. Image caching
2. Lazy loading

**Response:**

1. Sample project created in github: **02\_ContentProvider** ( this demo the integration with the phone contacts and gallery; more great examples on <http://developer.android.com/guide/topics/providers/contacts-provider.html> )
2. Straight forward image caching, please use the system caching mechanism –

URL url = new URL(strUrl);

URLConnection connection = url.openConnection(); connection.**setUseCaches**(true);

Object response = connection.getContent();

if (response instanceof Bitmap) {

Bitmap bitmap = (Bitmap)response; }

More on this mechanism on <http://pivotallabs.com/users/tyler/blog/articles/1754-android-image-caching>

1. SD card caching images – Lazzy loading listview library <https://github.com/thest1/LazyList>

**Kirk**

1. Unit testing, ui testing
2. SQLite best practices
3. Custom UI components

**Response:**

1. For the custom UI components demo via DS, MB, WK

**SQLite best practices**

1. In case you have a multi threaded application, you cannot afford to have more than a 1 SQLiteDatabase object pointing to a single database. Use SqliteOpenHelper object because it holds on to one database connection (one helper instance, one db connection)
2. Store as little data as possible. SQLite caches frequently accessed database pages.
3. Prepare generic statements that use named variables. Execute the statements when they are required by iterating through the variable values, binding the values to the named variables in each iteration.
4. Avoid subqueries. By default, the SQLite® library stores the subquery results in a temporary file.
5. Close database connections and resultsets, cursors as soon as you're done with them.
6. Batch inserts/updates by wrapping them in a transaction.
7. Don't ship a large pre-populated database with your app, download it instead or break it in chuncks
8. Store a version number in your database for easier app updating.
9. Dao objects maybe - <http://greendao-orm.com/>

**Unit testing**

1. Introduction to the framework and all related classes, Junit test: - <http://developer.android.com/guide/topics/testing/testing_android.html>
2. A sample of how to test Activities: - <http://developer.android.com/tools/testing/activity_test.html>
3. Mock the Android projects - <http://code.google.com/p/powermock/> ( sample <https://sites.google.com/site/androiddevtesting/> )

**UI testing:**

1. For final integration testing - **Robotium** ( like Selenium for Android ) - <http://code.google.com/p/robotium/>
2. Other alternatives - <http://pivotal.github.com/robolectric/>
3. The SDK provides two tools for functional-level application testing: **monkey** (is a command-line tool that sends pseudo-random streams of keystrokes, touches, and gestures to a device) and **monkeyrunner** (monkeyrunner tool is an API and execution environment for test programs written in Python with functions for connecting to a device, installing and uninstalling packages, taking screenshots, comparing two images, and running a test package against an application).

**Tom**

1. Push notifications
2. Location based
3. POI and locations around you

**Push notification response:**

1. Google Cloud Messaging for Android - <http://developer.android.com/google/gcm/index.html> ( server and app sample at <http://developer.android.com/google/gcm/demo.html> ), you will need a server side
2. Alternatives? – websocket, poll, sms, weird solutions ( <http://tokudu.com/2010/how-to-implement-push-notifications-for-android/> ) . What is better crossplatform: Apple Push Notification, Google Cloud Messaging, etc etc

**Location based - MapView:**

1. Sign your application - <http://developer.android.com/guide/publishing/app-signing.html>
2. Get the your certificate MD5 fingerprint - *keytool -list -alias androiddebugkey -keystore ~/.android/debug.keystore -storepass android -keypass android*
3. Generate a API key - <https://developers.google.com/maps/documentation/android/v1/maps-api-signup?hl=en>
4. Copy the key in the mapview
5. Send me the location in which you are via <http://itouchmap.com/latlong.html>

**Location based – POI-s around you**

1. Official info about Overlays: <https://developers.google.com/maps/documentation/android/v1/hello-mapview>
2. Sample overlays - <http://android-er.blogspot.de/2009/11/display-marker-on-mapview-using.html>