CS 653 : Mobile Computing

Instructor : Prof. Vinayak Naik

# Assignment 5 : UIActivityClassifier

Name : Debanjan Das

Roll : 173050069

1. **A traditional way to implement GUI it to hardcode UI elements in the source code. The Android provides a different approach. It separates the UI elements, its layout, source code, and its linkage. How does this approach play out? (6 marks) The new approach will give better efficiency, for some scenarios, when the app is getting upgraded. Give one such scenario. (2 marks)**

   Ans. The Android separates the UI elements, its layout, source code and linkage. This approach takes place by the following way :

   - First the following resource files (strings.xml, layout.xml etc.), manifestation file (AndroidManifest.xml) and source code file (MainActivity.java) are created.
   - Then the resource files and the manifestation file are given to the Android Asset Packaging Tool (AAPT) which compile the resources into binary assets and R.java file which contains all the resource ids for all resources in the *res/* directory.
   - Now, our source code (MainActivity.java) and R.java gets compiled together and corresponding Java bytecode (.class file) got generated. Now, this Java bytecode is cross-compiled to dalvik and it will generate Dalvik bytecode (.dex file).
   - The compiled resources and .dex file will be compressed into a zip-like file called Android Package (.apk) file. Now to distribute the app we need to sign the app and signing an app package means that you store an additional file in the .apk that is based on a checksum of the contents of the .apk and a separately generated private key.
   - Finally, after going through these many steps we get a verified android app.

   This approach will be efficient while upgrading the app. Such scenarios are :

   a. As the UI resources and source code are separated thus whenever source code gets updated, only the source code gets compiled not the resources files again. This will reduce the building and execution time. On the other hand, in traditional approach we have to compile the whole code again causing unnecessary redundancy.
   b. If we want to add the functionality of orientation changing while the orientation of the phone changes; then we just have to add one/two lines in AndroidManifest.xml file, while in traditional approach we have to write a whole function in the logic part of the source code to handle this situation.

2. **A mobile phone is constrained in terms of its screen size as compared to PCs. A typical phone displays one UI screen of one app at a time. Such a UI screen is called an Activity. How does the Android OS keep track of the Activities and in what order will these Activities be presented to the user by the OS? (4 marks) One app can have multiple Activities. Under what scenarios will such an app result in code duplication? (2 marks) How does Android solve this issue? (2 marks)**

   Ans : Android OS keeps track of the activities by using the Least Recently Used (LRU) list. The activities which are at the beginning of the list will be killed by the OS. The applications which has been restarted they will be added to the end of the list. The Android OS keep track of the activities based on a priority system :
   - Foreground activity are given priority 1. That is user is currently interacting with this application.
   - Visible activity are given priority 2. Here the activity is in *Paused state*. But, it is partially visible.
   - Services are given priority 3.
   - Background activities are given priority 4. Here the activity is in *Stopped state.*

   Android OS presents the activities in LRU order to the users.

   In our assignment 2, I implemented spinner for navigating to different activities whereas all the activities had the same action-bar. I had to copy that action-bar code to other activities layout files. So, the spinner navigation technique results in code duplication.

   Android solved this issue by introducing fragments. Fragment is a reusable segment of Android UI which can appear in an activity. We can implement the action bar as fragment which will eradicate the code duplication for the above case.

3. **A classifier can be graded in terms of its accuracy and precision. When would you regard a classifier to be good in its accuracy? (2 marks) How about precision? (2 marks) Give a scenario when a sensor has poor precision but good accuracy. (2 marks)**

Ans : Whenever a classifier is classifying the data points into the correct corresponding classes then it is said that it has high accuracy. But it also include another case. E.g, Let's say there are three classes A,B and C. The number of data points belonging to each of these classes are 95%, 2% and 3% respectively. Then if the classifier labels all the data points as class A then also it achieves higher accuracy of 95%.

But, on contrary the precision is low for the above case. As we know that if for a definite range of inputs the classifier outputs similar results then it has high precision but if the outputs are different while the input is in that range then the precision is said to be low.

A highly sensitive sensor can have high accuracy but low precision. Because, as it is highly sensitive it can correctly classify the data points. But because of its high sensitiveness its output might significantly vary on small change of input values. E.g, accelerometer which can change while walking in a queue and slow walking. Sensitive accelerometer can detect this with high accuracy but small change in the value of acceleration can predict different results, hence the precision will be low.

## References :

[1] http://www.vogella.com/tutorials/AndroidLifeCycle/article.html
[2] www.github.com/dogriffiths/HeadFirstAndroid/wiki