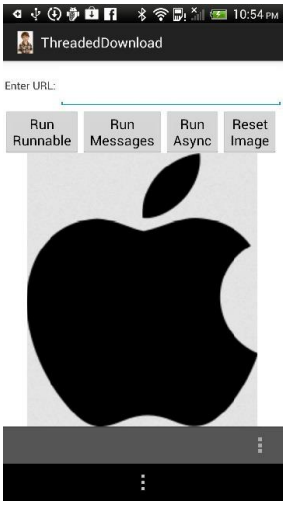
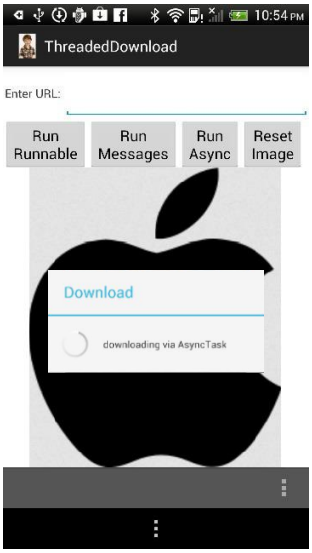
Assignment #5
Mobile Device Development
Android Multithreading
Due Date: 12/02/2021@11:59pm

The purpose of this assignment is to give you experience using various Android concurrency models to download bitmap images from a web server. The application works as follows:

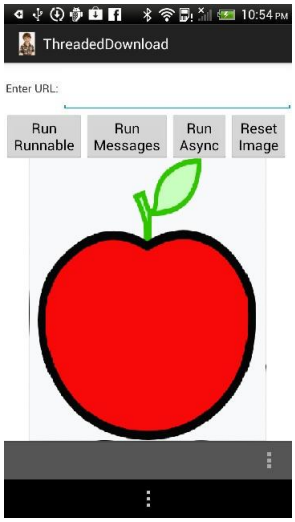
- The Activity provides a menu of buttons and displays a default image (configured via the XML resource files). The user is prompted to enter the URL for a new bitmap image, as shown in the following screenshot:



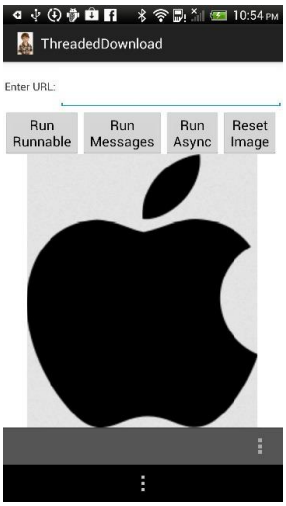
- After entering the desired URL, the user can select one of several buttons that provide different ways to download the image concurrently. The below screenshot shows how a progress dialog is displayed when downloading the image via the threading model:



- After the URL download has completed it will be displayed in an ImageView. The following screenshot shows the displayed image:



- The user can reset the image to its default contents by clicking the "Reset Image" button. The following screenshot shows the originally displayed image:



Program Description

This assignment involves writing an Android program that has the following features:

- Its AndroidManifest.xml contains the following:

```
<uses-permission
    android:name="android.permission.INTERNET">
</uses-permission>
```

- It contains a ThreadedDownloadActivity class that inherits from Activity and uses the XML layout containing a TextView object that prompts for the URL of the bitmap file and stores the entered URL in an EditText object.

- It uses four Button objects with the label "Run Runnable", "Run Messages", "Run Async", and "Reset Image" to run the corresponding hook methods that use the URL provided by the user to download the designated bitmap file via one of the following three Android concurrency models:

  1. Handlers and Runnables model ("Run Runnables").
  2. Handlers and Messages model ("Run Messages").
  3. AsyncTask model ("Run Async").

- Should any of these downloads fail you should produce a warning indicator via a Toast and/or EditText field. As this is most likely called from the background thread performing the download the toast or field update must forced to the UI Thread via one of the means we discussed in class.

- The Button objects that download the bitmap file must be connected to the corresponding ThreadedDownloadActivity.run*() methods via the appropriate android:onClick="..." attributes.

Skeleton Code

The following code is intended to give you some ideas of how to structure your program. Feel free to change this skeleton code if you have a better solution.

```
package edu.sjsu.android.threadeddownloads;

import <…>
public class ThreadedDownloadActivity extends Activity {
    // ... you fill in here

    @Override
    public void onCreate(Bundle savedInstanceState) {
        // ... you fill in here
    }

    Bitmap downloadBitmap (String url) {
        // ... you fill in here
    }

    public void runRunnable(View view) {
        // ... you fill in here
    }

    public void runMessages(View view) {
        // ... you fill in here
    }

    public void runAsyncTask(View view) {
        // ... you fill in here
    }

    public void resetImage(View view) {
        // ... you fill in here
    }
}
```

HttpURLConnection

HttpURLConnection is a powerful class that allows you to retrieve any type of data and content including images. I encourage you to read through the developer documents: https://developer.android.com/reference/java/net/HttpURLConnection.html

To use HttpURLConnection, we create an URL object and call openConnection() on it. We then evaluate the response in the form of an InputStream:

```
URL url = new URL("http://www.android.com/my.png");
HttpURLConnection urlC = (HttpURLConnection) url.openConnection();
try {
   InputStream in = new BufferedInputStream(urlC.getInputStream());
   Bitmap myBitmap = BitmapFactory.decodeStream(in);
   ……
} finally {
   urlC.disconnect();
}
```

Submission

1. Push your project directory along with the source to remote bitbucket repository by the due date.
2. Share your project repository the Grader (mohibkhanayubkhan.pathan@sjsu.edu) and Instructor (ramin.moazeni@sjsu.edu).
3. Submit a Readme.pdf to Canvas including your name, repository access link, instructions to run your program (if any), snapshot of your running program
4. Your project directory will be graded according to the state your project directory was in at due time when fetched.