Android Networking

EE5415

Mobile Apps Design and Development

Message

- Friendly reminder, please prepare the group project proposal and presentation in week 5.
- Due to the outbreak of the novel coronavirus, the proposal presentation is changed to video clip and PPT submissions.
  - Each project team must use a PPT file to record a 5-minute video presentation. You can combine multiple video clips of some members of your team as a 5-minute video presentation for satisfying the requirement.

Proposal Submission Requirements

- Submit your project proposal in MS-Words format, presentation in PPT format, and video presentation in MPEG-4 (.mp4) format to ee5415@gmail.com with subject
  - EE5415 Project Proposal: Group 1 (Project Name)
- Only attachments are accepted for the submission of MS-Words and PPT files.
- On-line storages are only accepted for video file submissions:
  - For example, Google Drive, Dropbox, etc.
  - Please try to not using China's on-line storages for the video submission, as we may has difficulty to download from these sites.

Agenda

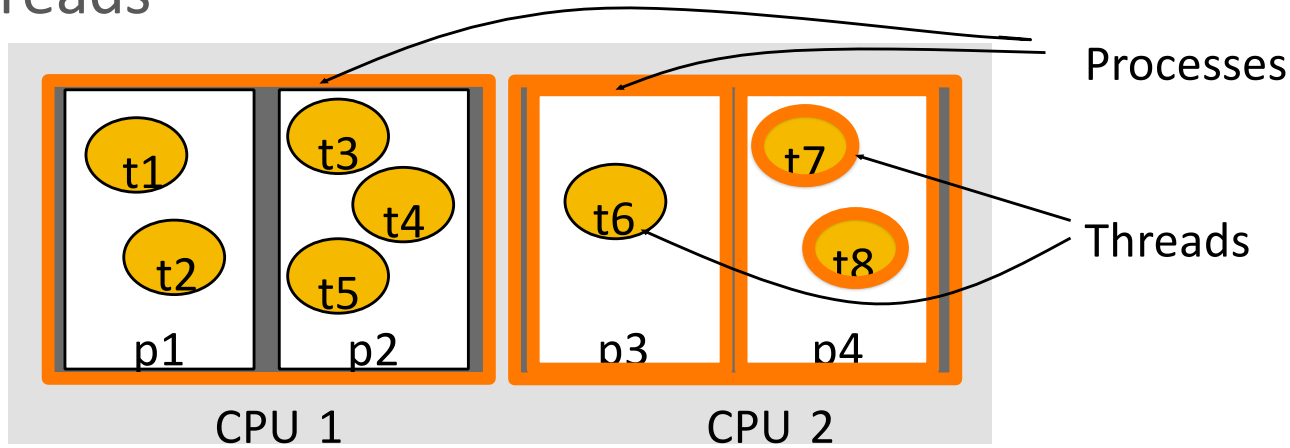
- Threading
  - Android's UI Thread
  - The Async Task class
- Networking
  - Socket and HttpURLConnection
  - Parse of JSON data
- Android Menus
  - Options Menu
  - Context Menu

- **Lab05: Localization, Multiple Screen and Menus**

Threading

What is a Thread?

- The Handler class Conceptual view
  - Parallel computation running in a process
- Implementation view
  - A program counter and a stack
  - with heap and static areas that are shared with other threads



Java Threads

- Represented by an object of type java.lang.thread
- Threads implement the runnable interface
  - void run()
-
- See: http://docs.oracle.com/javase/tutorial/essential/concurrency/threads.html

Thread Methods

Basic Thread Methods:

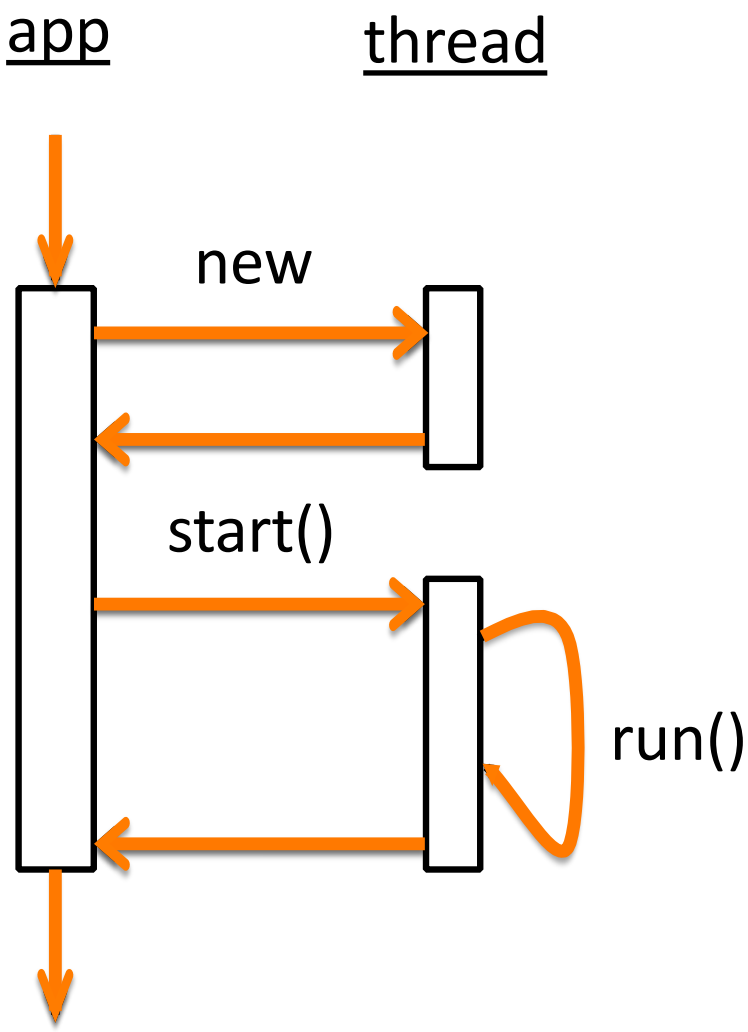
- void run()
  - Runs the Thread
- void start()
  - Starts the Thread void sleep(long time)
- Void sleep (long time)
  - Sleeps for the given period

Some Object Methods:

- void wait()
  - Current thread waits until another thread invokes notify() on this object
- void notify()
  - Wakes up a single thread that is waiting on this object

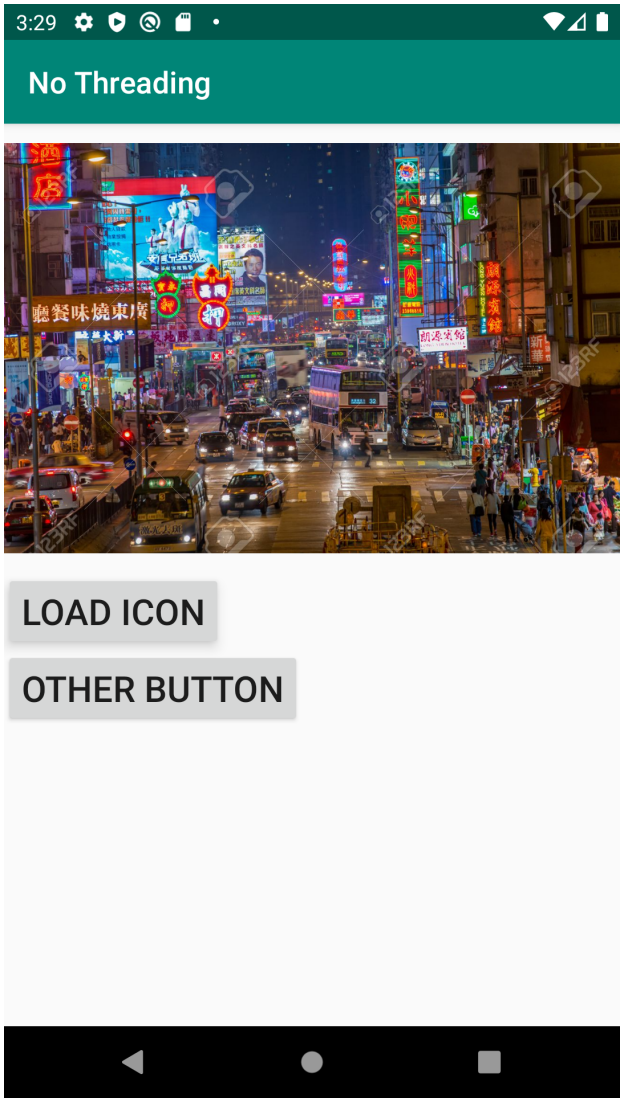
Basic Thread Use Case

- Instantiate a Thread object
- Invoke the Thread's start() method
- Thread's run() method get called
- Thread terminates when run() returns



Threading Examples

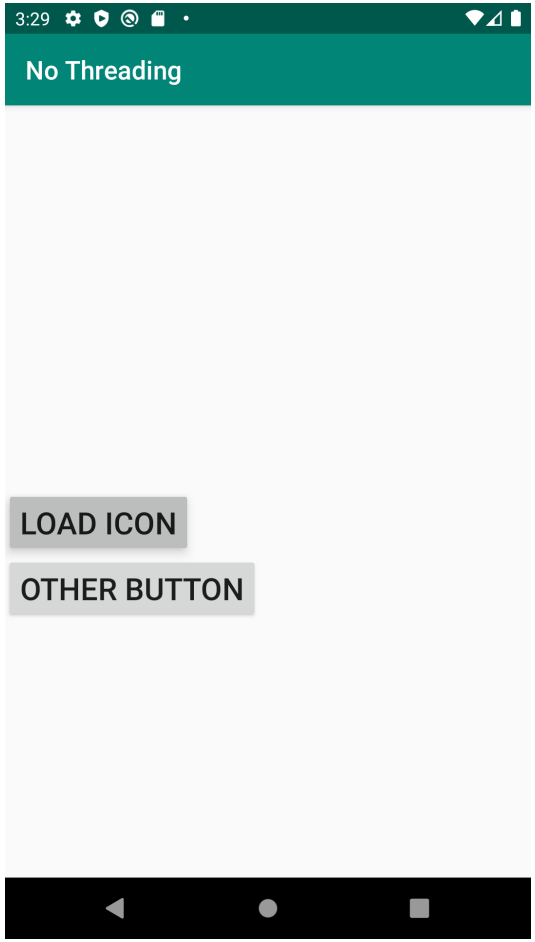
- Application displays two buttons
- LoadIcon
  - Load a bitmap from a resource file & display
  - Show loaded bitmap
- Other Button
  - Display some text



```java
public class MainActivity extends AppCompatActivity {

    private Bitmap mBitmap;
    private ImageView mImageView;
    private int mDelay = 5000;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mImageView = (ImageView) findViewById(R.id.imageView);

        final Button button = (Button) findViewById(R.id.loadButton);
        button.setOnClickListener((v) -> { loadIcon(); });

        final Button otherButton = (Button) findViewById(R.id.otherButton);
        otherButton.setOnClickListener((v) -> {
                Toast.makeText( context: MainActivity.this,  text: "The button is Working",
                        Toast.LENGTH_SHORT).show();
        });
    }
```

No Threading Example

- The “Other Button” cannot work until the image is loaded.
- It is because the loadIcon() is running in the same UI thread

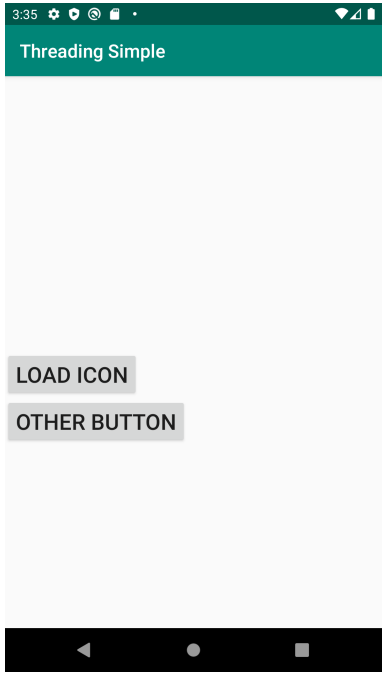



```
private void loadIcon() {
    try {
        Thread.sleep(mDelay);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    mBitmap = BitmapFactory.decodeResource(getResources(), R.drawable.hongkong);
}
```

Simple Threading

- The loadIcon() method runs as a new Thread.
- But the app crash when set the loaded image to the ImageView
  - Error Message: Only the original thread that created a view hierarchy can touch its views.

```
private void loadIcon() {
    new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                Thread.sleep(mDelay);
            } catch (InterruptedException e) {
                Log.e(TAG, e.toString());
            }
            mBitmap = BitmapFactory.decodeResource(getResources(),
                    R.drawable.hongkong);

            // This doesn't work in Android
            mIView.setImageBitmap(mBitmap);
        }
    }).start();
}
```



The UI Thread

- Applications have a main thread (the UI thread)
- Application components in the same process use the same UI thread
- User interaction, system callbacks & lifecycle methods handled in the UI thread
- In addition, UI toolkit is not thread-safe

Implications

- Blocking the UI thread hurts application responsiveness
  - Long-running operations should run in background threads
- Don’t access the UI toolkit from a non-UI thread

Improved Solution

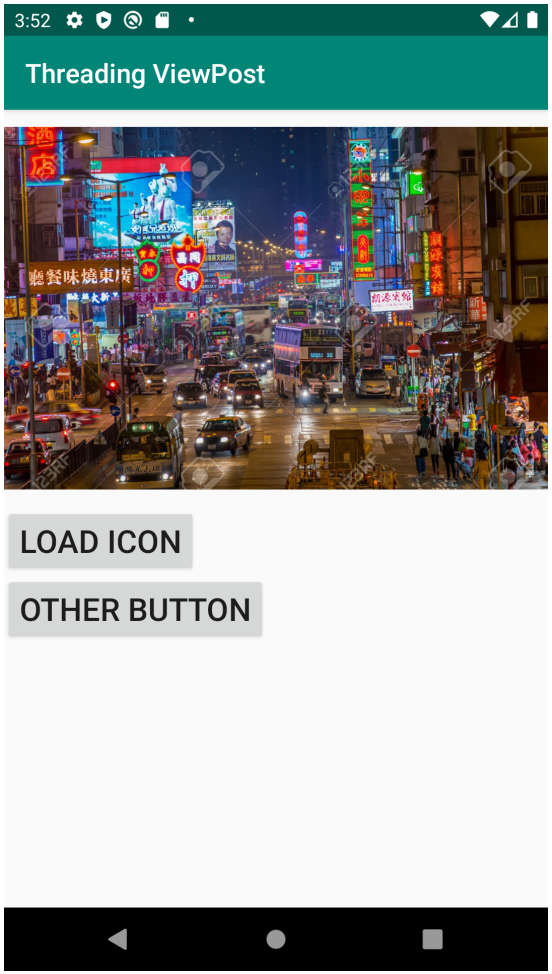
- Need to do work in a background thread, but update the UI in the UI Thread
- Android provides several methods that are guaranteed to run in the UI Thread

    boolean View.post (Runnable action)

    void Activity.runOnUiThread (Runnable action)

Threading ViewPost

```
private void loadIcon() {
    new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                Thread.sleep(mDelay);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            mBitmap = BitmapFactory.decodeResource(getResources(),
                    R.drawable.hongkong);
            mImageView.post(new Runnable() {
                @Override
                public void run() {

                    mImageView.setImageBitmap(mBitmap);
                }
            });
        }
    }).start();
}
```



AsyncTask

- Provides a structured way to manage work involving background & UI threads
- **Background Thread**
  - Performs work
  - Indicates progress
- **UI Thread**
  - Does setup
  - Publishes intermediate progress
  - Uses results

AsyncTask Class

- AsyncTask class is a generic class

  ```
  class AsyncTask<Params, Progress, Result> {
  …
  }
  ```

- Generic type parameters
  - Params – Type used in background work
  - Progress – Type used when indicating progress
  - Result – Type of result

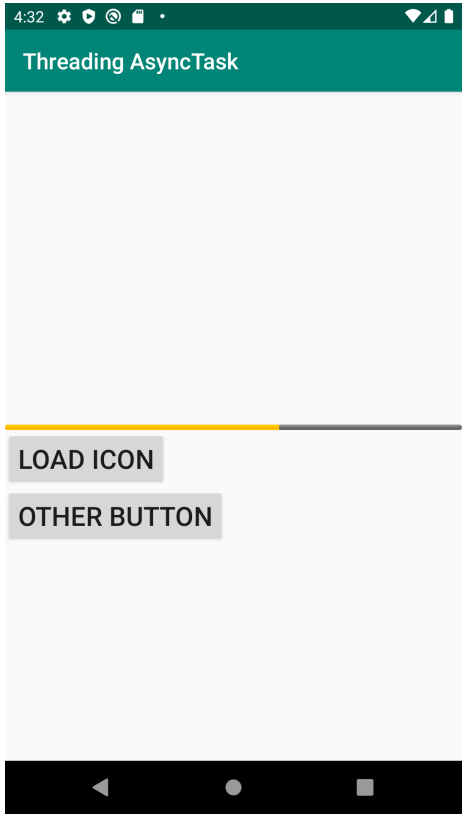
AsyncTask Workflows

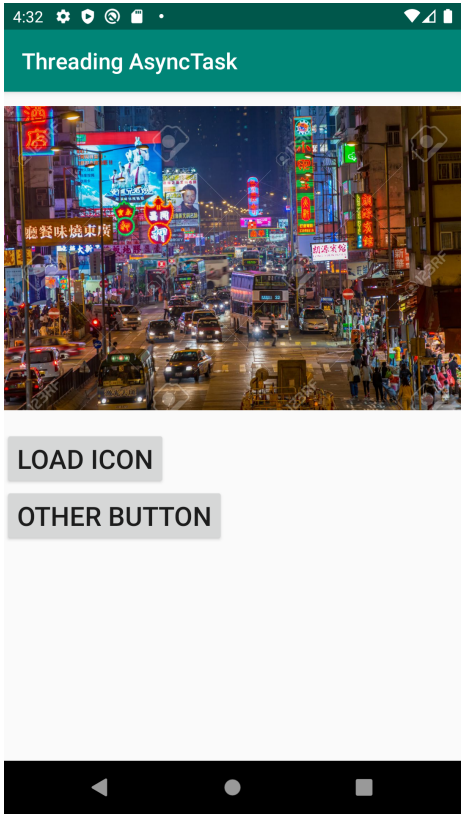
1. void onPreExecute()
   - Runs in UI Thread before doInBackground()
2. doInBackground (Params...params)
   - Performs work in background Thread
   - May call void publishProgress(Progress... values)
3. void onProgressUpdate (Progress... values)
   - Invoked in response to publishProgress()
4. void onPostExecute (Result result)
   - Runs after doInBackground()

Threading AsyncTask Example

```
final Button button = (Button) findViewById(R.id.loadButton);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        new LoadIconTask().execute(R.drawable.hongkong);
    }
});
```

Progress bar





```
class LoadIconTask extends AsyncTask<Integer, Integer, Bitmap> {

    @Override
    protected void onPreExecute() {
        mProgressBar.setVisibility(ProgressBar.VISIBLE);
    }

    @Override
    protected Bitmap doInBackground(Integer... resId) {
        Bitmap tmp = BitmapFactory.decodeResource(getResources(), resId[0]);
        // simulating long-running operation
        for (int i = 1; i < 11; i++) {
            sleep();
            publishProgress( ...values: i * 10);
        }
        return tmp;
    }

    @Override
    protected void onProgressUpdate(Integer... values) {
        mProgressBar.setProgress(values[0]);
    }

    @Override
    protected void onPostExecute(Bitmap result) {
        mProgressBar.setVisibility(ProgressBar.INVISIBLE);
        mImageView.setImageBitmap(result);
    }

    private void sleep() {
        try {
            Thread.sleep(mDelay);
        } catch (InterruptedException e) {
            Log.e(TAG, e.toString());
        }
    }
}
```

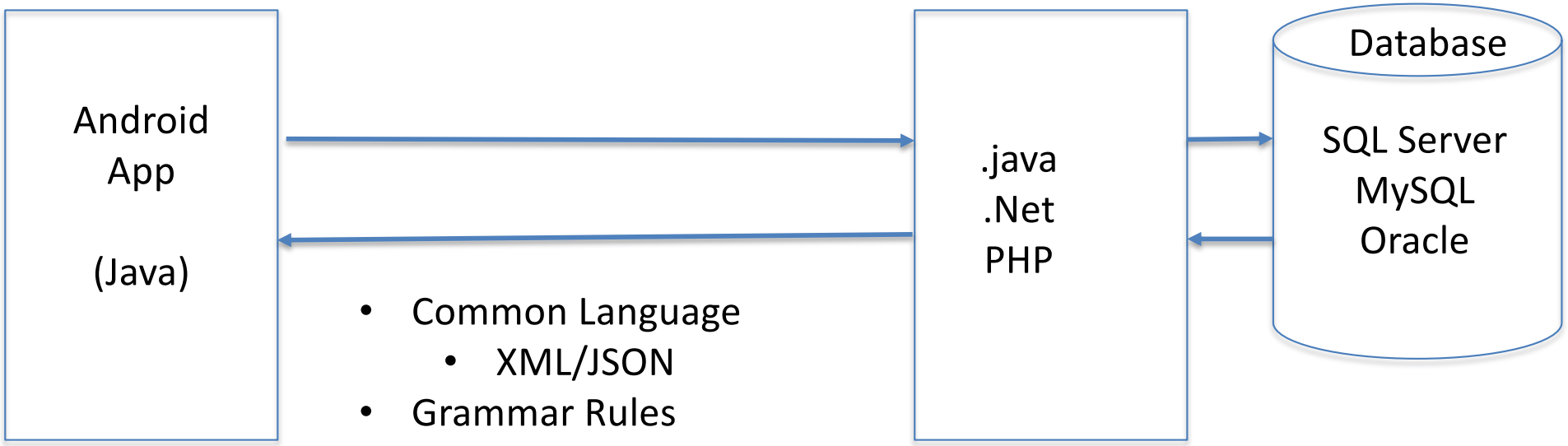
Android Networking

Why Networking?

- There should not be a limit for storing the data
- Data can be accessed by any where from the world
- Android devices have great mobility and connectivity
- Many applications use data and services via the Internet
  - HTTP: Hyper Text Transfer Protocol
  - URL: Uniform Resource Locator

Web Services

- Android cannot directly communicate with the remote database
- **Web Services is used to provide communication between two different technologies or two same technologies.**



Android Networking

- Android support 3 HTTP client packages for network connections:
  - Java.net – (Socket, URL)
  - **Java's HttpURLConnection – recommended**
  - Apache HTTP client – removed in Android 6.0

Connecting to the Network

1. Declare Permissions:

- Declare Permissions to use the network in AndroidManifest.xml

2. Make Separate Threads:

- Network operations involve unpredictable delays. Hence, it's recommended that you always perform network operations on separate threads other than UI thread (main thread).
- Another method is to use Async Task class

3. Connect and Download Data:

- A client app can use Socket or HttpURLConnection classes to send HTTP requests and receive.

Networking Permission

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.httpurlconnloadimage">

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

Manage Multithreading in Android

The Main Thread

- Each app has a single foreground thread and also known as **Main Thread** or **UI Thread**
- All visual elements are on the main thread
  - Don’t block the main thread!
  - Don’t access the user interface from other threads!

Using Background Threads

- Must run network communications in the background
- Standard Java concurrent programming
  - `Thread`, `Runnable`, `java.concurrent`
- Android-specific techniques
  - `AsyncTask`
  - `AsyncTaskLoader`
  - `IntentService`

AsyncTack

- It’s intended for **shorter tasks**, lasting a few seconds
- Each instance of an `AsyncTask` can be executed only once
- An `AsyncTask` is bound to a single activity
- `AsyncTask` object **do not survive configuration changes**

AsyncTaskLoader

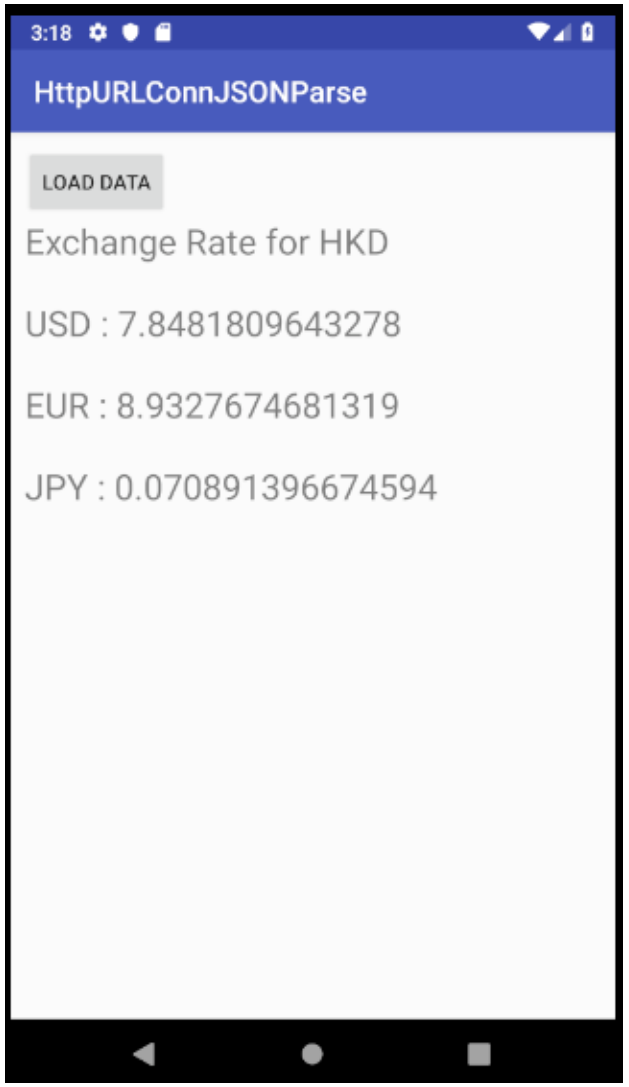
- Loader wrapper for an `AsyncTask`
- Survives configuration changes elegantly
- Works well with fragments
- Bound to a single activity

IntentService

- Decoupled from user interface
- Reliably manages **longer request**/response cycles
- Launch with Intent object that packages instructions
- Stops automatically when task is complete
- Communicate with UI with broadcast messages

Four Example Apps

- Apps send a request to a networked server for HK Dollar Exchange Rate data
- Then displays the requested data in a TextView with ScrollView



JSON Service for Exchange Rate

http://www.floatrates.com/daily/hkd.json

{

"usd":{"code":"USD","alphaCode":"USD","numericCode":"840","name":"U.S. Dollar","rate":0.12741806089147,"date":"Thu, 28 Feb 2019 00:00:01 GMT","inverseRate":7.8481809643278},

"eur":{"code":"EUR","alphaCode":"EUR","numericCode":"978","name":"Euro","rate":0.11194738960435,"date":"Thu, 28 Feb 2019 00:00:01 GMT","inverseRate":8.9327674681319},

"gbp":{"code":"GBP","alphaCode":"GBP","numericCode":"826","name":"U.K. Pound Sterling","rate":0.095742628325155,"date":"Thu, 28 Feb 2019 00:00:01 GMT","inverseRate":10.444668351947},

"cad":{"code":"CAD","alphaCode":"CAD","numericCode":"124","name":"Canadian ":

. . .

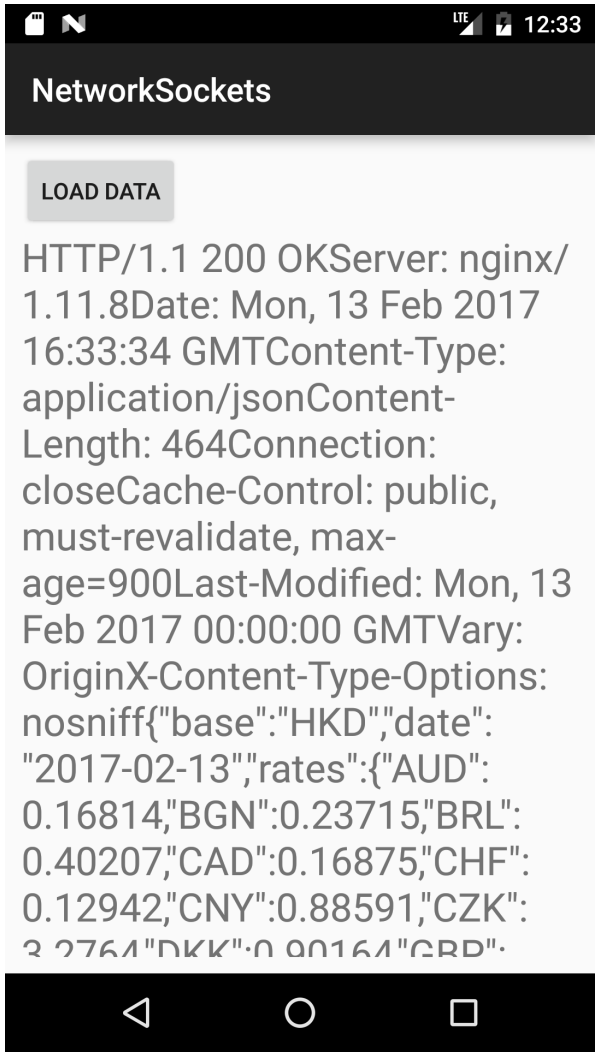
}

Sending HTTP Requests

- Socket
- HttpURLConnection

NetworkSockets : Layout XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dp"
    android:orientation="vertical" >

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Load Data"
        android:onClick="loadData" />

    <ScrollView
        android:id="@+id/scrollView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:id="@+id/textView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="24sp" >
        </TextView>
    </ScrollView>

</LinearLayout>
```



NetworkSockets (1)

```
public class MainActivity extends ActionBarActivity {

    TextView mTextView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        setContentView(R.layout.activity_main);
        mTextView = (TextView) findViewById(R.id.textView1);

    }

    public void loadData(View view) {
        new HttpGetTask().execute();
    }

:
:

}
```

NetworkSockets (2)

Host: www.floatrates.com
GET /ldaily/hdk.json HTTP/1.1
Connection: close

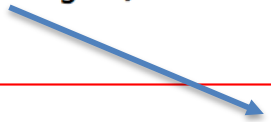
```
// The HttpGetTask Class to handle the network connection
private class HttpGetTask extends AsyncTask<Void, Void, String> {

    private static final String HOST = "www.floatrates.com";
    private static final String HTTP_GET_COMMAND = "GET /daily/hkd.json"
            + " HTTP/1.1"
            + "\n"
            + "Host: "
            + HOST
            + "\n"
            + "Connection: close" + "\n\n";
```

NetworkSockets (3)

```
@Override
protected String doInBackground(Void... params) {
    Socket socket = null;
    String data = "";

    try {
        socket = new Socket(HOST, 80);
        PrintWriter pw = new PrintWriter(new OutputStreamWriter(
                socket.getOutputStream()), true);
        pw.println(HTTP_GET_COMMAND);

        data = readStream(socket.getInputStream());

    } catch (UnknownHostException exception) {
        exception.printStackTrace();
    } catch (IOException exception) {
        exception.printStackTrace();
    } finally {
        if (null != socket)
            try {
                socket.close();
            } catch (IOException e) {
                Log.e(TAG, "IOException");
            }
    }
    return data;
}
```

NetworkSockets (4)

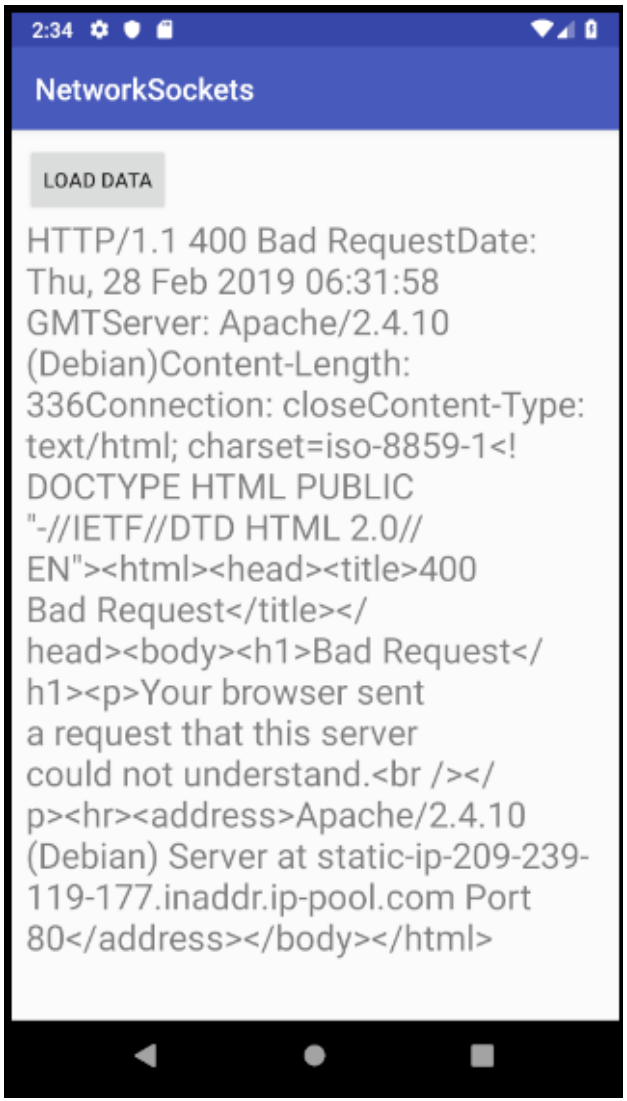
```
private String readStream(InputStream in) {
    BufferedReader reader = null;
    StringBuffer data = new StringBuffer();
    try {
        reader = new BufferedReader(new InputStreamReader(in));
        String line = "";
        while ((line = reader.readLine()) != null) {
            data.append(line);
        }
    } catch (IOException e) {
        Log.e(TAG, "IOException");
    } finally {
        if (reader != null) {
            try {
                reader.close();
            } catch (IOException e) {
                Log.e(TAG, "IOException");
            }
        }
    }
    return data.toString();
}
```



```
@Override
protected void onPostExecute(String result) {
    mTextView.setText(result);
}
```

NetworkSockets (5)

- The main drawback of using network socket connection is the returned data with header information, which is not easy to read and handle for further processing.



HttpURLConnection

- Higher-level than Sockets
- This example app loads the JSON with use of HttpURLConnection from http://www.floatrates.com/daily/hkd.json
- Only the JSON text is extracted while the headers are removed.

HttpURLConn Load JSON (1)

```
public class MainActivity extends AppCompatActivity {

    //Connectivity Manager instance
    private ConnectivityManager mConnMgr;

    //Text View for displaying the downloaded JSON data
    private TextView mTextView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Store the Connectivity Manager in the member variable
        mConnMgr = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);

        //Image View reference to update the image later
        mTextView = (TextView) findViewById(R.id.textView);

    }
```

HttpURLConn Load JSON (2)

```
public void loadData(View v) {

    //Website URLtowhich a network request will be sent
    String path = "http://www.floatrates.com/daily/hkd.json";

    //Bullet proofing test to make sure connection manager reference is not null
    if (mConnMgr != null) {
        //Get active network info
        NetworkInfo networkInfo = mConnMgr.getActiveNetworkInfo();
        //If any activie network is available and inernet connection is available
        if (networkInfo != null && networkInfo.isConnected()) {
            //Start to data download Async Task
            new DownloadDataTask().execute(path);
        } else {
            //If network is off of Internet is not availble, inform the user
            Toast.makeText( context: this,  text: "Network Not Avaialble", Toast.LENGTH_LONG);
        }
    }
}
```

HttpURLConn Load JSON (3)

```
private class DownloadDataTask extends AsyncTask<String, Void, String> {

    //doInBackground is executed on background thread
    @Override
    protected String doInBackground(String...urls) {
        return downloadData(urls[0]);
    }

    protected void onPostExecute(String data) {
        //Set the newly downloaded data to the text view on the activity
        if (mTextView != null) {
            mTextView.setText(data);
        }
    }
}
```

HttpURLConn Load JSON (4)

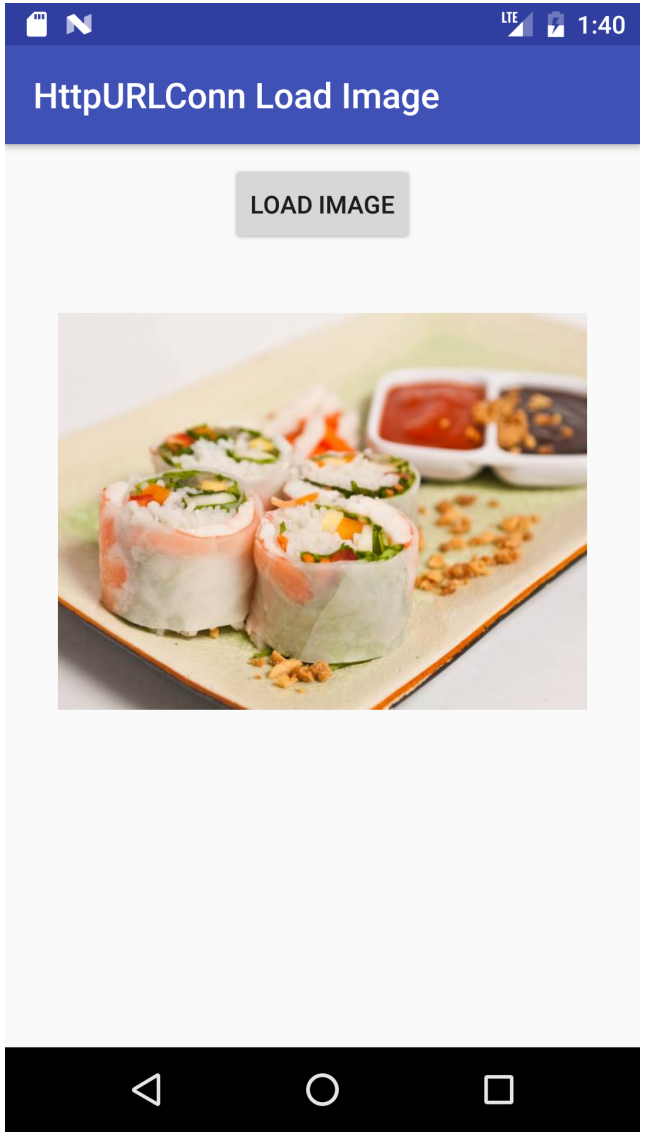
```
public String downloadData(String path) {
    String data = null;
    InputStream inStream;

    try {
        //Create a URL Connection object and set its parameters
        URL url = new URL(path);
        HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
        //Set connection time out of 5 seconds
        urlConn.setConnectTimeout(5000);
        //Set read connection time out of 2.5 seconds
        urlConn.setReadTimeout(2500);
        //Set HTTP request method
        urlConn.setRequestMethod("GET");
        urlConn.setDoInput(true);

        //Perfrom network request
        urlConn.connect();

        //After the network response, retrieve the input stream
        inStream = urlConn.getInputStream();
        //Convert the input stream to String Bitmap
        data = readStream(inStream);

    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

    return data;
}
```

HttpURLConn Load JSON (5)

```
private String readStream(InputStream in) {
    BufferedReader reader = null;
    StringBuffer data = new StringBuffer("");
    try {
        reader = new BufferedReader(new InputStreamReader(in));
        String line = "";
        while ((line = reader.readLine()) != null) {
            data.append(line);
        }
    } catch (IOException e) {
        Log.e("HttpGetTask", "IOException");
    } finally {
        if (reader != null) {
            try {
                reader.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    return data.toString();
}
```

HttpURLConn Load Image

- This example app loads image with use of `HttpURLConnection` approach from the website
  - http://lorempixel.com/644/480/
- The downloaded image is displayed on the `ImageView`



```
public Bitmap downloadImage(String path) {
    Bitmap bitmap = null;
    InputStream inStream;

    try {
        //Create a URL Connection object and set its parameters
        URL url = new URL(path);
        HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
        //Set connection time out of 5 seconds
        urlConn.setConnectTimeout(5000);
        //Set read connection time out of 2.5 seconds
        urlConn.setReadTimeout(2500);
        //Set HTTP request method
        urlConn.setRequestMethod("GET");
        urlConn.setDoInput(true);

        //Perfrom network request
        urlConn.connect();

        //After the network response, retrieve the input stream
        inStream = urlConn.getInputStream();
        //Convert the input stream to Bitmap object
        bitmap = BitmapFactory.decodeStream(inStream);

    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

    return bitmap;
}
```

Processing Http Responses

- Several popular formats including
  - JSON
  - XML

Javascript Object Notation (JSON)

- Intended to be a lightweight data interchange format
- Data packaged in two types of structures:
  - Maps of key/value pairs
    - { “username” : “Peter Chen” }
  - Ordered lists of values
    - [{ “username” : “Peter Chen” }, { “username” : “Ryan Li” }, { “username” : “Emily Zhang” }]

See: http://www.json.org/

Earthquake Data (JSON Output)

http://api.geonames.org/earthquakesJSON? north=44.1&south=-9.9&east=-22.4&west=55. 2&username=demo


```
{ "earthquakes”:
  [
    {
        "eqid":"c0001xgp","magnitude":8.8,"lng":142.369, " "src":"us”,
        "datetime":"2011-03-11 04:46:23","depth":"
        24.4,"lat":38.322
     },
    {"eqid":"2007hear","magnitude":8.4,"lng":101.3815,"
      "src":"us”,"datetime":"2007-09-12 09:10:26","depth": " 30,"lat":-4.5172},
...
    {"eqid":"2010xkbv","magnitude":7.5,"lng":91.9379,"
      "src":"us”,"datetime":"2010-06-12 17:26:50","depth":" 35,"lat":7.7477}
  ]
}
```


The value of the “earthquakes” map is an order List

JSON Service for Exchange Rate

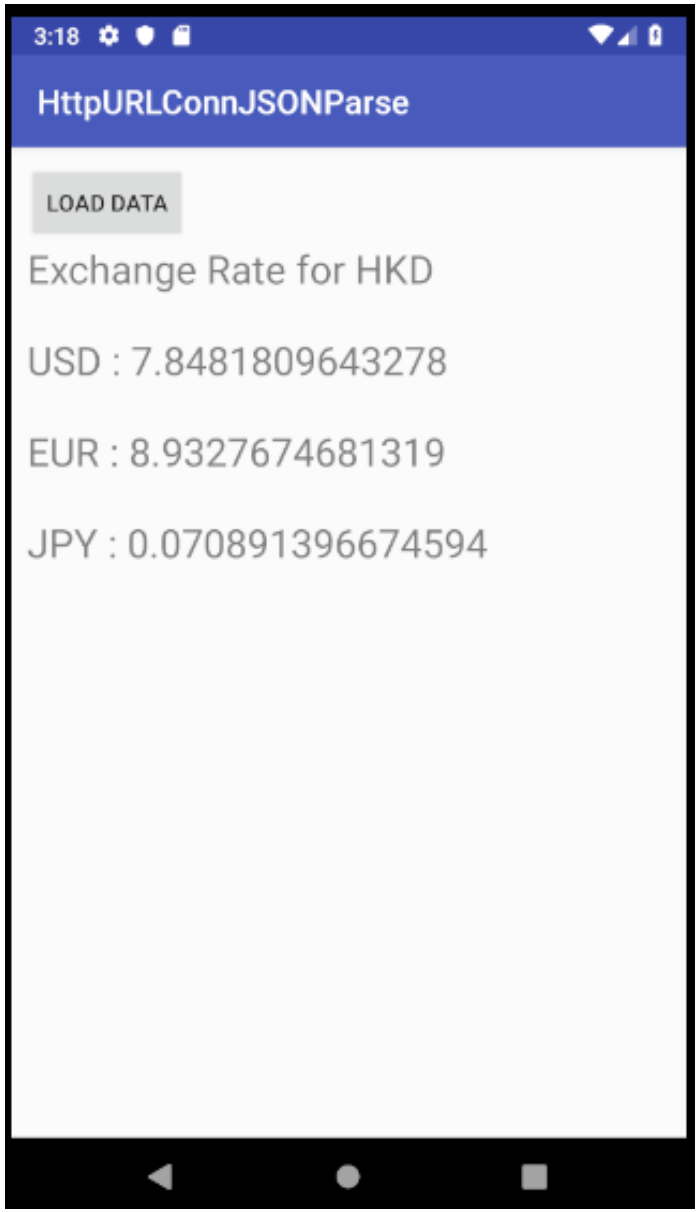
- http://www.floatrates.com/daily/hkd.json

{"usd" : {"code":"USD","alphaCode":"USD","numericCode":"840","name":"U.S. Dollar","rate":0.12741806089147,"date":"Thu, 28 Feb 2019 00:00:01 GMT","inverseRate":7.8481809643278},
"eur":{"code":"EUR","alphaCode":"EUR","numericCode":"978","name":"Euro","rate":0.11194738960435,"date":"Thu, 28 Feb 2019 00:00:01 GMT","inverseRate":8.9327674681319},
"gbp":{"code":"GBP","alphaCode":"GBP","numericCode":"826","name":"U.K. Pound Sterling","rate":0.095742628325155,"date":"Thu, 28 Feb 2019 00:00:01 GMT","inverseRate":10.444668351947},
. . .
}

- *There are many maps for each currency*
- *For USD, key = "usd" and value is **collection of object** for representing the detail of the rate such as code, alphaCode, mumericCode, etc.*

Parse the JSON Data

- This example app loads the JSON with use of HttpURLConnection from http://www.floatrates.com/daily/hkd.json
- Parse some of the useful data for display



HttpURLConn Parse JSON

```
protected void onPostExecute(String data) {
    //.........Process JSON DATA.................
    if(data !=null){
        try{
            // Get the full HTTP Data as JSONObject
            JSONObject reader= new JSONObject(data);
            JSONObject rateReader;
            String mapString, rateString;

            mTextView.setText("Exchange Rate for HKD\n\n");

            // Get the date of the USD Exchange Rate
            mapString = reader.getString( name: "usd");
            rateReader = new JSONObject(mapString);
            rateString = rateReader.getString( name: "inverseRate");
            mTextView.setText(mTextView.getText()+ "USD : " + rateString +"\n\n");

            // Get the date of the EUR Exchange Rate
            mapString = reader.getString( name: "eur");
            rateReader = new JSONObject(mapString);
            rateString = rateReader.getString( name: "inverseRate");
            mTextView.setText(mTextView.getText()+ "EUR : " + rateString +"\n\n");


            // Get the date of the JPY Exchange Rate
            mapString = reader.getString( name: "jpy");
            rateReader = new JSONObject(mapString);
            rateString = rateReader.getString( name: "inverseRate");
            mTextView.setText(mTextView.getText()+ "JPY : " + rateString +"\n\n");

            // process other data as this way.............

        }catch(JSONException e){
            e.printStackTrace();
        }
    } // if statement end
}
```