CPEN 475/575
Project 3: A multithreaded networked application

Deliverables: Please clean your Android Studio project, then zip the containing directory (into a .zip file) and submit.

Gradle Info (see build.gradle for the app):

```
compileSdkVersion 30
minSdkVersion 15
targetSdkVersion 30
//with minimum dependencies on the following (can be later)
dependencies {
  implementation fileTree(dir: 'libs', include: ['*.jar'])

  implementation 'androidx.appcompat:appcompat:1.1.0'

  implementation "androidx.preference:preference:1.1.0"

  implementation 'com.google.android.material:material:1.1.0'

  implementation
'androidx.constraintlayout:constraintlayout:1.1.3'

}
```

Target SDKs I will compile and test it on a Google Pixel running API 30.

Overview:
This is just an exercise to familiarize you with network communications, AsyncTask, and JSON. Please install and run the example app to see behavior. Please see the sample projects on the course website.
Do not worry about screen rotations for this project.

Features Create an application that has/does the following;

1. Please create an application with an appbar.
2. The appbar should have a spinner and an overflow icon that leads to settings.
3. Please create a settings screen with one ListPreference. It contains a choice of 2 URLs (see Helpful Bits). These sites are where both the json list of pet information and individual pet pics are stored.

4. Please create a preference change listener on this preference(see Helpful Bits). On preference change, this listener repopulates the spinner, and the applications main activity.
5. Please make all network calls on a separate thread. You will have to download both text (JSON) and images.

Typical Use

1. On startup the app checks for network connectivity, if there is none it alerts the user.
2. Next the app will download the json file pets.json from the site chosen by the user in the settings (or if initial app install, default to the CNU site)
3. If this site cannot be reached (and the Teton Software one cannot) the app should tell the user what the server status code is and what link was tried. This should consists of an appropriate error graphic (see apk below for demo).

4. The spinner on the appbar is dynamically populated from information downloaded and parsed from pets.json using standard Android JSONObject and JSONArray. This is a flexible design that allows updates to the pets list.
5. The app downloads whichever pet the user specifies in the spinner and displays it in the background.
6. Any errors should be reported to the user on the main page (see apk)

Grading

5	gradle files correct
20	Preferences and Preference Change listener
15	behavior when no networking available
20	networking calls
20	threading
20	coding practices and style for instance how objects in the pets list are implemented refactoring repetitive code , no magic numbers

Gotchas:

1. Set device on airplane mode, start app, get a warning ‘network unreachable’, uncheck airplane mode, it should start working when you change a preference

Helpful bits

- Install and test the demo apk
- Dont forget to request appropriate permissions in the manifest
- Sites where JSON and images are stored

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="JSON_URL_NAME">
        <item>CNU - Defender</item>
        <item>Pair - Teton Software - Pets</item>


    </string-array>
    <string-array name="JSON_URL">
        <item>https://www.pcs.cnu.edu/~kperkins/pets/</item>
        <item>https://www.tetonsoftware.com/pets/</item>

    </string-array>
</resources>
```

- **Getting the JSON info**
  The file containing the json data is named pets.json. Download pets.json with this URL.

https://www.pcs.cnu.edu/~kperkins/pets/pets.json

it will return the following JSON text (I will probably change this for testing submissions)

```
{
"pets":[
        {
        "name":"Winston",
        "file":"p0.png"
        },
        {
        "name":"Higgens",
p0.png  "file":"p1.png"
        },
        {
        "name":"Broccoli",
        "file":"p2.png"
        }
        ]
}
```

Use the info in the name and file fields to populate the spinner. Please use 1 list only to hold this data, something like List<pet> myList. It will be part of the adapter associated with the spinner. The pet object just holds the name and its associated file.

- **Downloading the image associated with the pet the user choose**
  Once the user selects a pet by name download its associated image. For example if the user selected Winston above, you would download p0.png using the following link.
  https://www.pcs.cnu.edu/~kperkins/pets/p0.png

Preference listener – Make sure your preference listener is a member of your main activity (See http://stackoverflow.com/questions/2542938/sharedpreferences-onsharedpreferencechangelistener-not-being-called-consistently for why). This means you have something like the following example;

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_class_lab2);
    myPreference=PreferenceManager.getDefaultSharedPreferences(this);

    // listen for a change to the listPref key,
    //its the key part of the key value pair that holds the URL to load
    //when this key changes then the URL has changed to so reload it
    //make this listener an instance var so it is not GCed due to it being
    //saved as a weak reference
    listener = new SharedPreferences.OnSharedPreferenceChangeListener() {
      public void onSharedPreferenceChanged(SharedPreferences prefs, String key) {
          if (key.equals("listPref")) {
              loadImage();
          }
      }
    };

    //gotta register the listener
    myPreference.registerOnSharedPreferenceChangeListener(listener);
```

Want to hide the title? In activity onCreate…

```
getSupportActionBar().setDisplayShowTitleEnabled(false);
```

What should I use for activity_main.xml

I used a combination LinearLayouts, and a Framelayout. Here are the widgets I used:

```
<com.example.perkins.proj3.WebImageView_KP
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:id="@+id/imageView"
  android:scaleType="fitXY"/>
<TextView
  android:id="@+id/tvLarge"
  android:layout_width="match_parent"
  android:layout_height="60sp"
  android:text="404!"
  android:textSize="40sp"
  android:textStyle="bold"
  android:gravity="center"
  android:layout_gravity="center_horizontal|bottom"
  android:background="#00cccccc"/>
<TextView
  android:id="@+id/tvSmall"
  android:layout_width="match_parent"
  android:layout_height="70sp"
  android:text="having a spot of \nserver troubles?"
  android:textSize="25sp"
  android:textStyle="bold"
  android:gravity="center"
  android:layout_gravity="center_horizontal|bottom"
  android:background="#00cccccc"/>
```