When an app runs, get a single thread, the UI thread
Long running code should not be run on this thread
if app does not return quickly, platform may kill the app
There is a way to get more threads running
Only the main thread should update the UI
All of your code now needs to be thread safe
Note: the android UI toolkit is not thread safe!
AsyncTask class is one way to get do thread processing
There also the Thread and Handler classes, underlying mechanisms
AsyncTask helps use them with less "pain"
Required methods to overload:
doInBackground(...) -- does the actual work
Other methods to overload:
onPreExecute() -- called by the UI thread, possibly sets up a progress bar ...
onPostExecute() -- called by the UI thread, clean up
onProgressUpdate(Progress ...) -- called by the UI thread
doInBackground() calls publishProgress(Progress...)
AsyncTask class -- can be cancelled

- Call the cancel() method -- returns boolean
  - onCancelled() method will be called instead of onPostExecute(), on the UI thread
- doInBackground() should call isCancelled() if in a long loop

Issues for using the AsyncTask

- Task must be created on the UI thread
- UI threads calls execute() to run the task
- Task can be run only once. execute() called a second time => exception thrown
- getStatus() returns one value of AsyncTask.Status (PENDING, RUNNING, FINISHED)

AsyncTask <doInParam, onProgressParam, onPostParam>

- onPostParam is also the return type of doInBackground()

Book’s AsyncDemo program
Activities are destroyed and re-created in many instances

- orientation rotations
- low memory situations ...

What happens to AsyncTasks?

- Continue running ...

What happens to references to views? (think onPostExecute())

- Need a way to get new Activity to get access to "old" AsyncTask

Activity

- @Override Object onRetainNonConfigurationInstance()
  - Gets called just before onDestroy()
  - return the AsyncTask

- getLastNonConfigurationInstance()
  - returns null or last object returned from OnRetain...

With fragments ... setRetainInstance() ..

- But still needs to get new activity

Add a method to the class to set current context

- onActivityResult() can call this method
Chapter 47 Services

- Service Class -- to perform background tasks
  - started services, bound services and intent services
  - service tag in the manifest

- Started Services:
  - Started by an app and runs until stopped
  - Runs past the lifetime of the starting app
  - startService() method used to start them
  - Can stop self by calling stopSelf()
  - stopService() can be called by another app
  - Services have a high priority and are the last to be stopped

- Intent Services
  - Subclass of Service
  - Started by intents rather than startService()
  - onHandleIntent() queues requests to be handled
Bound Service

- Bound service allows interaction and results from starting app
- Other services don’t
- Example: audio player, player is service, UI app controls the player via IPC
- "Client" starts and binds a service
- Service can be "unbound" when no longer needed

Service Methods

- onCreate() -- as expected
- onBind() or onStartCommand() or onHandleIntent() -- depends on kinds of service
- onDestroy()

onStartCommand returns one of:

- START_NOT_STICKY -- not restarted if destroyed by framework
- START_STICKY -- restart requested
- START_REDELIVER_INTEN -- service was destroyed after returning

Service can be started on system start with appropriate manifest data
Sample programs: Chapters 48, 49 and 50

48 -- Started Service (ServiceExample)
- service is started by an intent
- service code uses an AsyncTask
- this service doesn’t update the UI
- watch this service via the Logcat

49 -- Bound Service (LocalBound)
- Activity can call methods in the service
- Service in this case has no AsyncTask
- Can the service update the UI?
  - Can give the activity a reference (setMethod)
  - Activity.runOnUiThread(Runnable action)
  - Activity needs to let service know if of changes

50 -- Remote Bound Service (RemoteBound)
- Sending messages rather than direct calls
- Sending a message may throw an exception
- This service does have access for toasts!
Notifications -- many apps do them!
Local or Remote: local -> this device, remote -> remote server
Icon on the status bar, pull down drawer shows notifications
Flip gets rid of notification
Creating a Notification channel
```
notificationMgr = (NotificationManager)
    getSystemService(Context.NOTIFICATION_SERVICE);
```
NotificationChannel chnl = new NotificationChannel(id, name, importance)
  id -- something like edu.wwu.csci412.app
  name --
  channel attributes
    setDescription(String)
    setImportance(int)
      IMPORTANCE_UNSPECIFIED, _MIN, _DEFAULT, _NONE, _LOW, _HIGH
    UNSPECIFIED should not be used
    setLightColor
    setLockscreenVisibility
    setName
    setShowBadge
Notifications (page 2)

- enableVibration
- setVibrationPattern
- notificationMgr.createNotificationChannel(chnl)

Creating the Notification

- Action buttons or pushes can fire intents
- Create a PendingIntent
- Notification.Action can be created.
  - "Buttons" on the notification
  - Array of actions need to be created
    - new Notification.Builder(context, channelId)
      - .setContentTitle, .setContentText
      - .setSmallIcon
      - .setContentIntent (pendingIntent)
      - .setActions
      - .setNumber -- number of outstanding notifications
      - .build() -- final action
- notificationMgr.notify(notificationID, notification) -- ID -> which notification
- NotifyDemo app
  - Two activities
- Android 7.0 can bundle notifications into groups
  - .setGroup in building the notification
  - .setGroupSummary(true) - to just show a summary of the group
  - build several in a row, builderSummary.build() shows summary
  - (not shown in app)
- Android 7.0 added "Direct Reply"
  - Remote Input Object -- View in the notification
  - RemoteInput.Builder(resultkey)
    - setLabel
    - setAllowDataType -- default string
    - setChoices -- make choices available
- DirectReply app --
  - Intent sent to self
  - ri = RemoteInput.getResultsFromIntent(Intent)
  - ri.getCharSequence(resultkey)
  - Send "reply" back to notification
Chapter 53 - Multi-Window support

- Split-Screen, Freeform, Picture-in-Picture Modes (TV) (large devices)
  - long press on Overview button (square button)
  - draw window size, to zero to close, Overview double box .. long click
  - freeform arbitrary window sizes on device

getPackageManager().hasSystemFeature(PackageManager.FEATURE_FREEFORM_WINDOW_MANAGEMENT)

- Multi-Window -- apps can be aware and change UI based on Window
  - activity.isInMultiWindowMode()
  - onMultiWindowModeChanged(boolean,Configuration)

- Attributes
  - android:defaultWidth
  - android:defaultHeight
  - android:gravity - start, end, left, right, top ...
    - can be used like top|left
    - need to read more about gravity
Can create a new task and run an activity in the same app
   - `intent.FLAG_ACTIVITY_LAUNCH_ADJACENT, _MULTIPLE_TASK, _NEW_TASK`

Can set up freeform size and position

```java
Rect rect = new Rect(0, 0, 100, 100);

ActivityOptions options = ActivityOptions.makeBasic();
ActivityOptions bounds = options.setLaunchBounds(rect);
startActivity(i, bounds.toBundle())
```

Chapter 54 -- Multi Window demo app

review code
SharedPreferences

- Ref: https://developer.android.com/training/data-storage/shared-preferences
- Also Professional Android, 4ed, Meier & Lake, Chapter 8

- Saving data across invocations of the application
  - can be shared between all components of an app

- SharedPreferences is the class -- a key/value pair data storage
  - Supports: Boolean, Float, Int, Long, String, StringSet
  - May have multiple named preferences
    - getSharedPreferences(name, mode)
      - mode: MODE_PRIVATE, MODE_WORLD_READABLE, MODE_WORLD_WRITABLE, MODE_MULTI_PROCESS

- In a "Context" method
  - Context ctxt = getApplicationContext();
  - SharedPreferences prefs = ctxt.getSharedPreferences("Name", MODE_PRIVATE);

- Other Methods:
  - getDefaultSharedPreferences(context)
  - activity method: getPreference() -- gets Default
SharedPreferences (page 2)

- SharedPreferences methods
  - getAll() - returns a Map
  - getBoolean(key, defaultValue)
  - getFloat(), getInt(), getLong(), getString(), getStringSet()
  - registerOnSharedPreferenceChangeListener, unregisterOnSharedPreferenceChangeListener
  - edit() -- returns a SharedPreferences.Editor

- SharedPreferences.Editor
  - putBoolean(key, value)
  - putFloat(), putInt(), putLong(), putString(), putStringSet()
  - remove(key)
  - clear() - removes all keys
  - commit() or apply() - makes the changes

- Sample programs:
  - SavingStateDemo
  - Other Prefs programs -- not much direct code
What about more complex data?
Common use in Mobile applications/servers
Supported in several ways
org.json: JSONArray, JSONObject, JSONStringer, JSONTokenizer
Builds Objects you can use ...
android.util.JsonReader/JsonWriter/JsonTokenizer
Allows your code to more easily parse JSON.
Encode a larger "object" into a JSON string
Add that string to the preferences
Reading: convert JSON string to object
Example JSON

[
  {
    "id": 912345678901,
    "text": "How do I read JSON on Android?",
    "geo": null,
    "user": {
      "name": "android_newb",
      "followers_count": 41
    }
  },
  {
    "id": 912345678902,
    "text": "@android_newb just use android.util.JsonReader!"
  },
  {
    "id": 912345678903,
    "text": null
  }
]
Internet enabled applications ... 

- Need permissions in manifest
  - `<uses-permission android:name="android.permissionINTERNET"/>

- URLConnection -- primary class to talk to a web server
  - subclass: HttpURLConnection
  - indirect subclass: HttpsURLConnection

- URL class to prepare data for the URLConnection

```java
URL url = new URL("http://www.android.com/");
HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
try {
    InputStream in = new BufferedInputStream(urlConnection.getInputStream());
    readStream(in);
} finally {
    urlConnection.disconnect();
}
```
A different version

```java
try {
    URL url = newURL("http://facultyweb.cs.wwu.edu/~phil");
    URLConnection conn = url.openConnection();
    HttpURLConnection httpConn = (HttpURLConnection) conn;
    if (httpConn.getResponseCode() == HttpURLConnection.HTTP_OK) {
        InputStream in = httpConn.getInputStream();
        .... use in ...
    }
    httpConn.disconnect();
} catch (MalformedURLException e) {
    ...
} catch (IOException e) {
    ...
}
```
URLConnection can not run on the UI thread
  □ An exception will be thrown
  □ Use and AsyncTask or a content provider
  □ Https version is more complex as it deals with the SSL layer

Download Manager -- service for downloading files from internet
  □ << not complete >>