Handling Background Tasks

The UI Thread

- Activities run on UI thread
  - are responsible for handling user interaction
- Unless separate thread created anything an Activity starts runs on UI thread
  - e.g. Service, web-download, database access
- To retain responsive UI must move time consuming tasks to another thread
  - Two main ways
    - threads using a Handler
    - subclass AsyncTask

Handler

- Handles message queue for a thread
  - from which handler created
- Can send messages or post runnables on message queue
  - handler provides synchronization
- Runnables
  - post runnables to be executed on thread
    - e.g. using post = PlayAudioProgressProject
- Messages
  - send messages to handler to perform an action on thread
    - handler overrides handleMessage to receive message
- E.g. HandlerAppProject
  - handler running on UI thread
    - thread sends messages every second
    - Message object
AsyncTask

- Abstract helper class
  - creates an asynchronous task that communicates back to UI thread
  - creates thread for task and handles it
  - provides callbacks for synchronization and message passing

- Generic parameters
  - Params
  - Progress
  - Result

- Override
  - doInBackground
    - code for asynchronous task
    - gets Params
    - post back via publishProgress with Progress
    - returns Result
  - onProgressUpdate
    - callback to UI thread from task
    - receives Progress
    - onPreExecute
    - callback to UI when task complete
    - receives Result
    - also
      - onPreExecute, onCancelled

- E.g. AsyncAppProject
  - same example using AsyncTask

E.g. HTTPAppProject

- Download and display image from web
  - using HttpURLConnection
    - requires INTERNET permission

- Steps
  - create a URL
  - open connection
  - setup connection attributes
  - obtain InputStream
  - read data
  - close stream

- Time consuming so wrap in AsyncTask
  - download code executed in task
  - when done, load bitmap into UI
Services

- Processing that does not require a UI
  - resource intensive processing that does not require user interaction
    - service must create thread
  - processing that is done routinely or at regular intervals
    - e.g. downloading fresh content
  - processing that spans Activity lifecycle
  - processing to be supplied to other apps
- Lifecycle
  - created via startService or bindService
    - unless already running
  - stopped when stopService, stopSelf and/or no longer bound

Started service
- started by startService
- can pass info to service
- no further communication
- typically local service

Bound service
- started by bindService
- opens a long-standing connection
- can send/receive messages
- can be local or bound from other apps

Extend Service class and override

Started Service

- Started via startService(intent)
  - intent can carry data in a bundle
- Stopped externally via stopService(intent)
- Stopped internally via stopSelf()
- Callbacks
  - onCreate ()
    - when first created
  - onStartCommand(intent, flags, startID)
    - when started (can be multiple starts)
    - receives data via Intent Bundle
    - override to handle request
    - return value indicates operation mode
      - START_STICKY – run until stopped
      - START_NOT_STICKY – run until processed request
E.g. CreateServiceProject

- Input id, service processes id
- Service registered in manifest
- Started service
  - onBind returns null
  - override onStartCommand to process request
  - retrieves data from Bundle
- Activity starts service via startService
  - Intent
  - local service uses class name
  - external service uses package name
  - creates Bundle to pass id
- Runs until stopped (START_STICKY)

Bound Service

- Service with interaction
  - client can send/receive messages from service
  - service only runs while bound
  - Started via bindService(intent, connection, flags)
  - intent action gives package name of service
  - connection object receives IBinder when connection established
  - Send messages to service via Messenger
    - messenger attached to binder returned by service
    - Message contains data Bundle
  - Service creates Handler to handle messages received
    - override handleMessage
    - msg can have a messenger to replyTo
  - Service creates Messenger using handler
    - return binder from messenger in onBind

E.g. CreateBoundServiceProject

- Input id, bound service processes ids
- Service registered in manifest with intent filter
  - indicates which intent actions the service will respond to
  - Bound service
    - extends Handler to receive messages
    - overrides handleMessage
      - gets data bundle from message
    - creates a Messenger using handler
    - implements onBind
    - returns binder from the messenger
Activity
- creates a ServiceConnection to receive connection notifications
  * on connected, creates messenger on binder
- binds to service when started (onStart)
  * intent with ACTION since client
- unbinds when stopped (onStop)
- onClick
  * creates Message with data bundle
  * sends message to bound service

System Services
- Interaction with system-level services
- Services
  - alarm manager
  - power manager
  - notification manager
  - location manager
  - etc.
- Accessed via getSystemService

AlarmManager
- To schedule execution of activity/service/broadcast at specific time
  - set(type, time, intent)
- Alarm types
  - RTC_WAKEUP - wakes device from sleep to fire intent
  - RTC - fires intent but doesn’t wake up device
  - ELAPSED_REALTIME - fires at time since device booted
  - ELAPSED_REALTIME_WAKEUP - wakes up and fires
- One alarm per intent
- Cancel via cancel(intent)
- Can be repeating
- Time via Calendar object
  - setTimeMillis
  - add
E.g. AlarmProject

- E.g. reminder
- ReminderActivity
  - fired at a specified time
  - displays reminder message
- MainActivity
  - sets alarm on button click
  - PendingIntent
    - reference to an intent that is started at another time
    - methods to return different types of pending intents
      - e.g. Activity, Broadcast, etc.
      - getActivity(context, id, intent, flags)
  - getAlarmManager
  - set alarm (Calendar object)