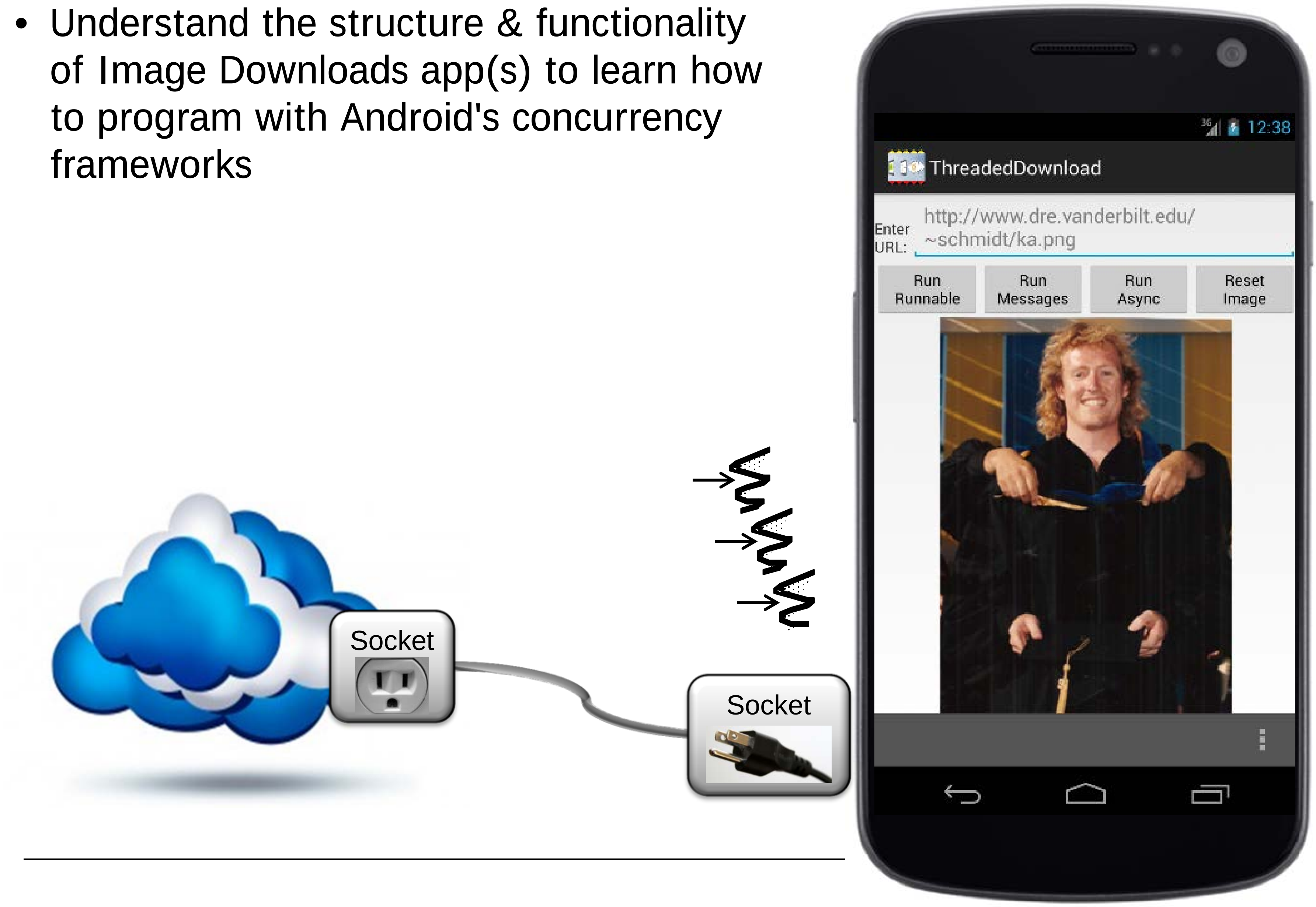
Android Concurrency: Overview of Image Downloads App(s)

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software
Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA

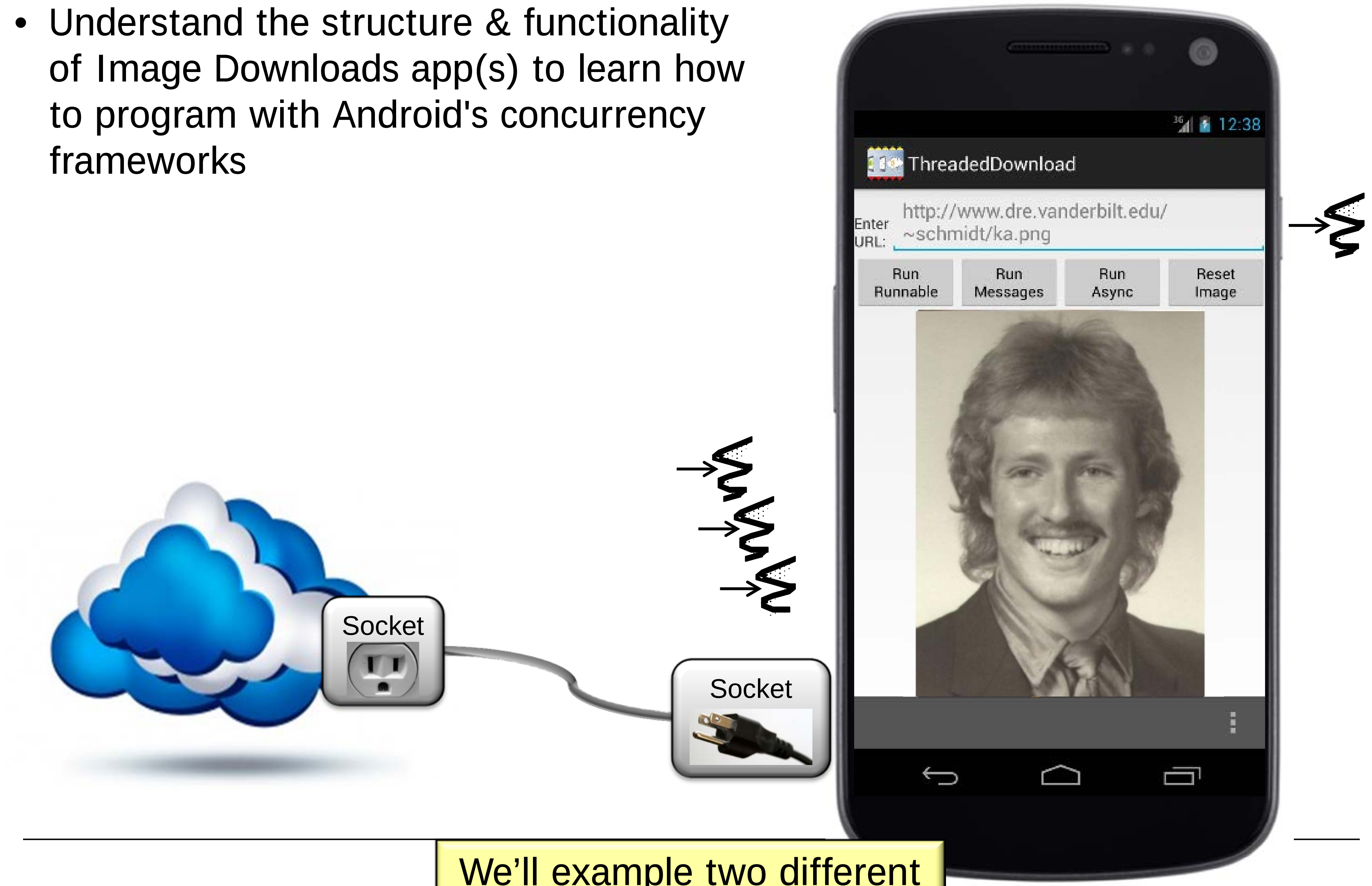
Learning Objectives in this Part of the Module

- Understand the structure & functionality of Image Downloads app(s) to learn how to program with Android's concurrency frameworks



Learning Objectives in this Part of the Module

- Understand the structure & functionality of Image Downloads app(s) to learn how to program with Android's concurrency frameworks

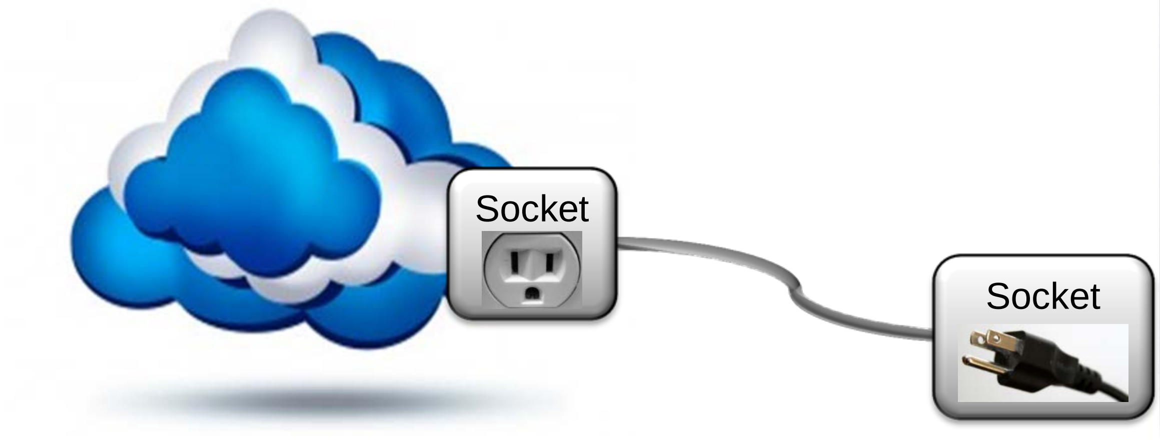


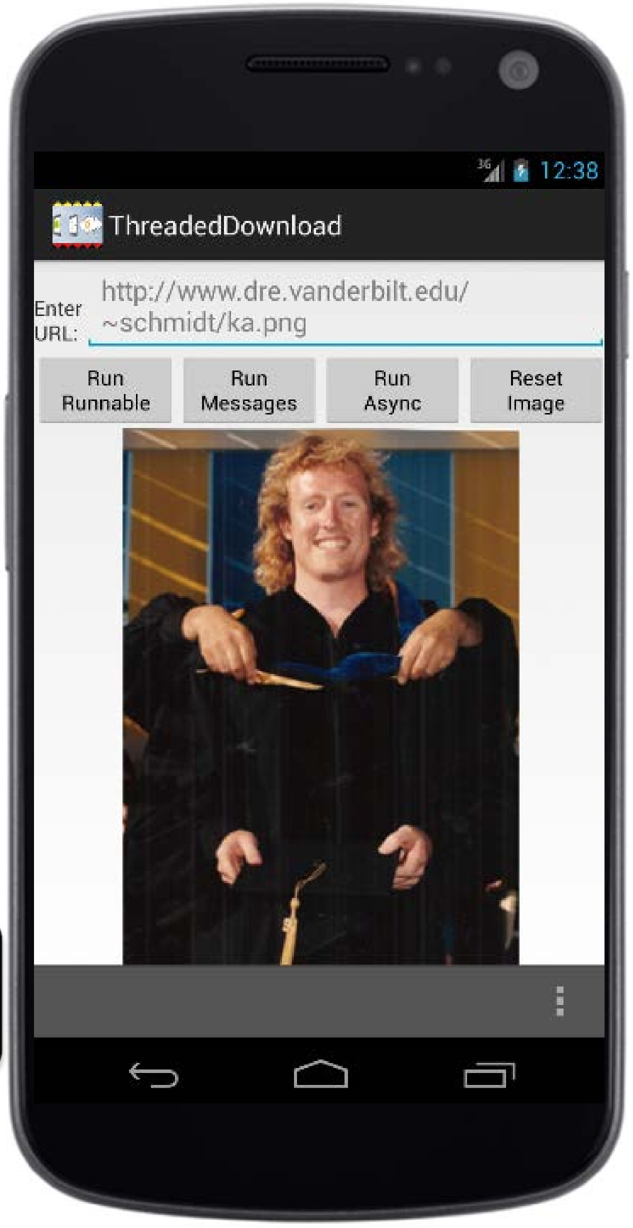
We'll example two different implementations of this app

Overview of the Image Downloads App

Overview of the Image Downloads App

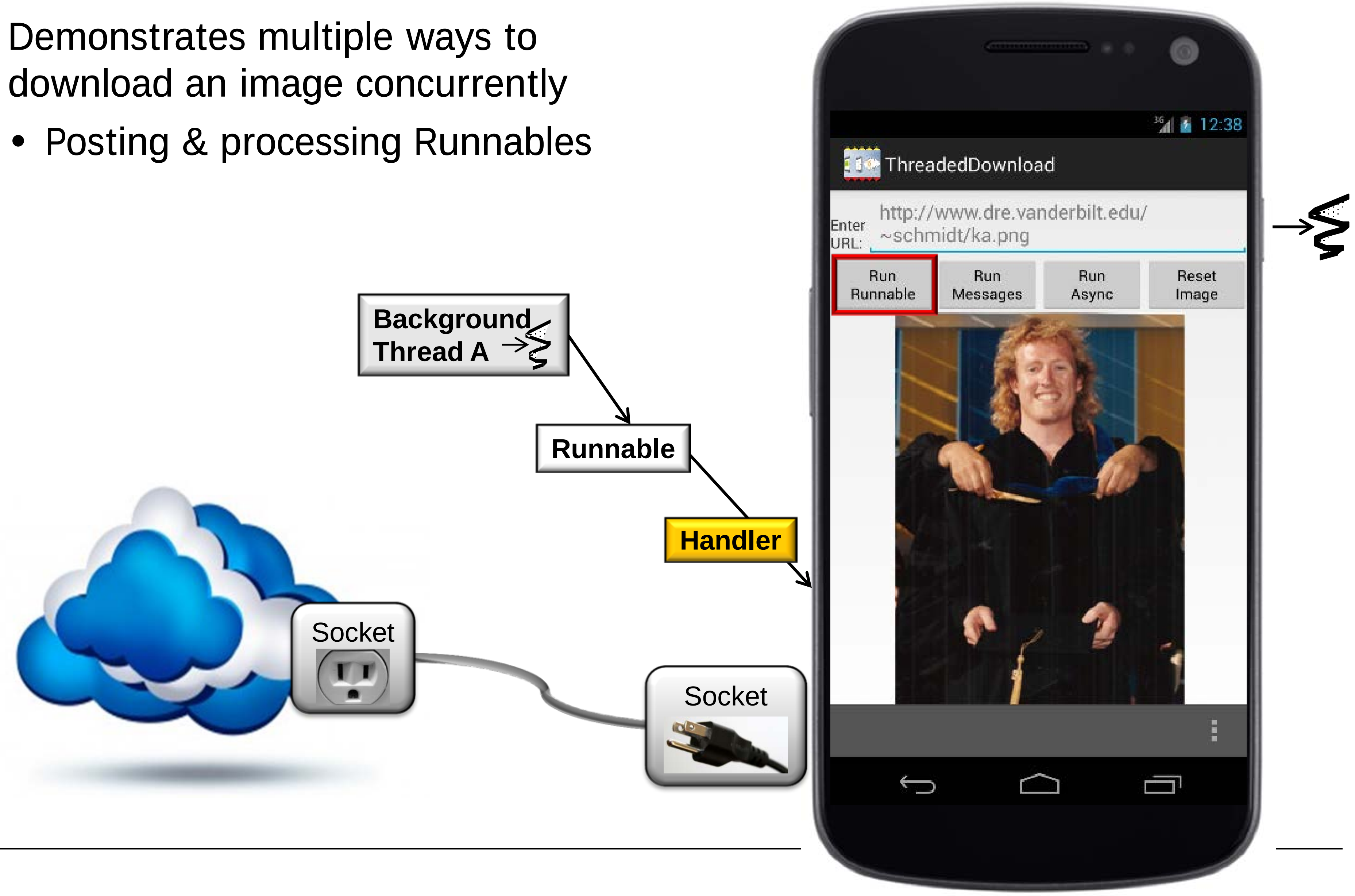
- Demonstrates multiple ways to download an image concurrently





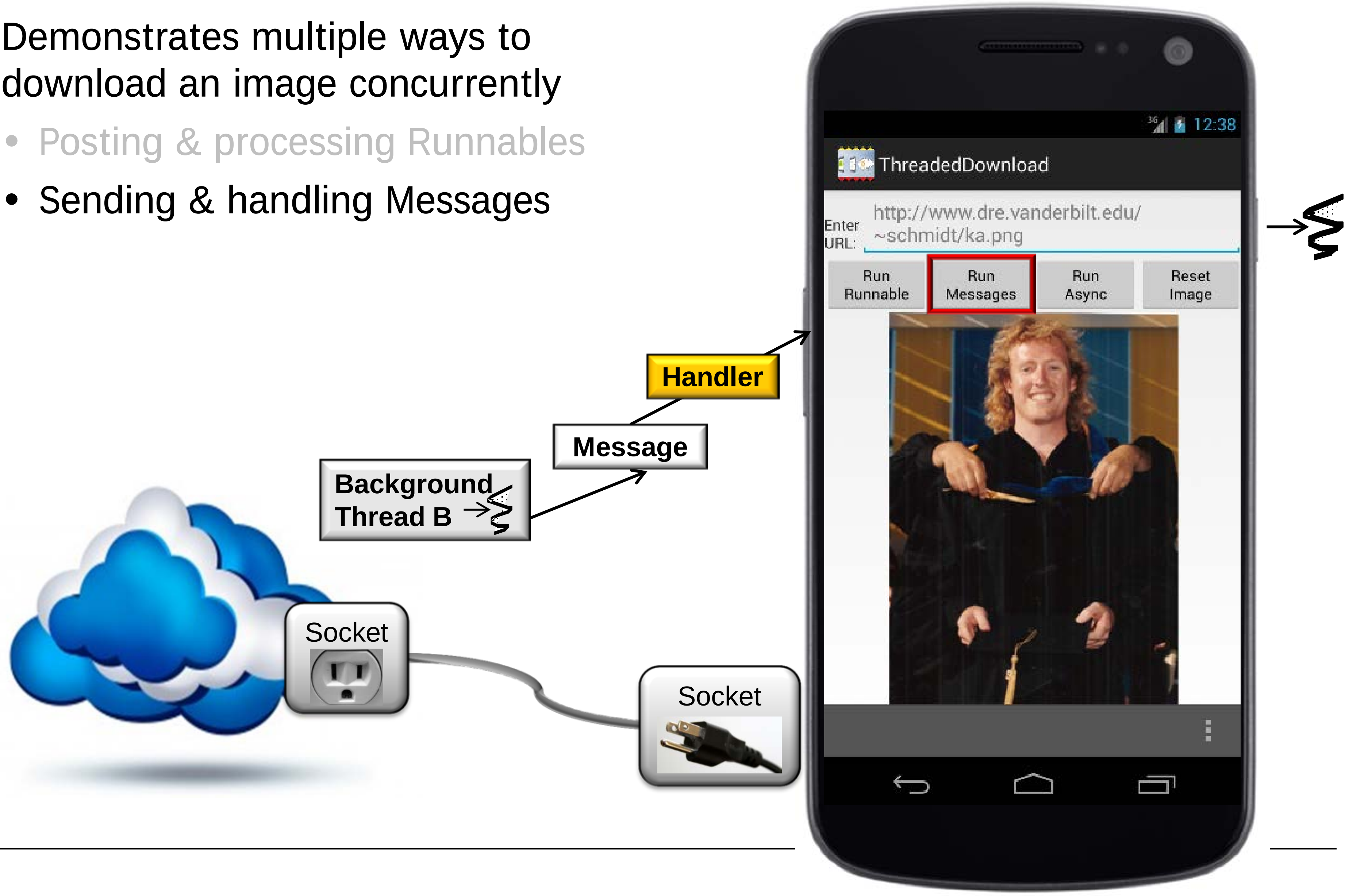
Overview of the Image Downloads App

- Demonstrates multiple ways to download an image concurrently
  - Posting & processing Runnables



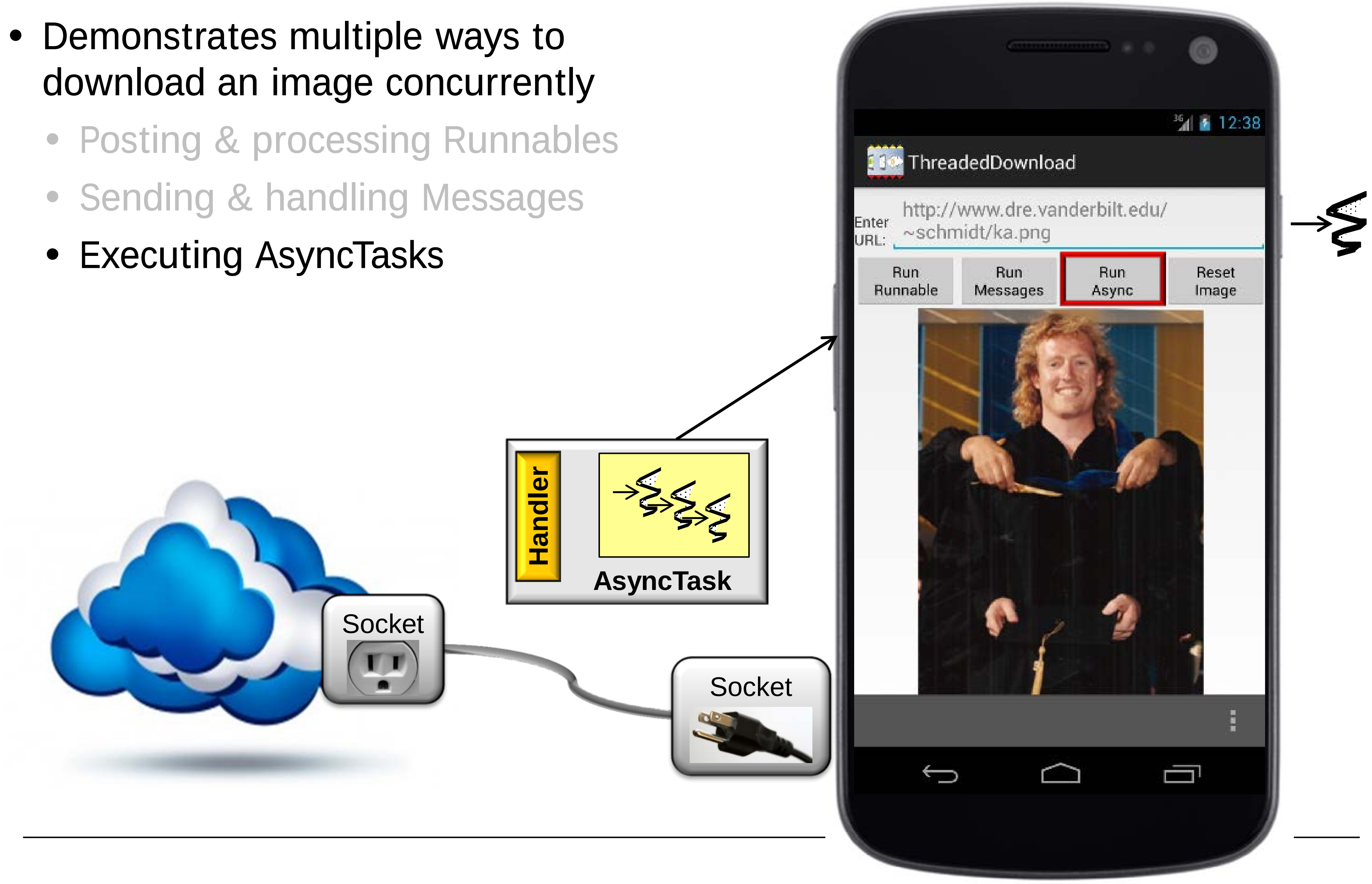
Overview of the Image Downloads App

- Demonstrates multiple ways to download an image concurrently
  - Posting & processing Runnables
  - Sending & handling Messages



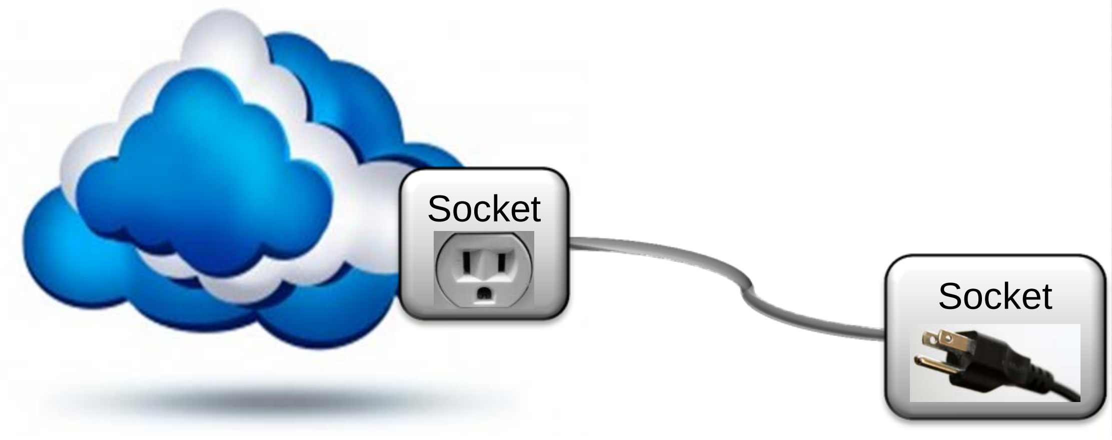
Overview of the Image Downloads App

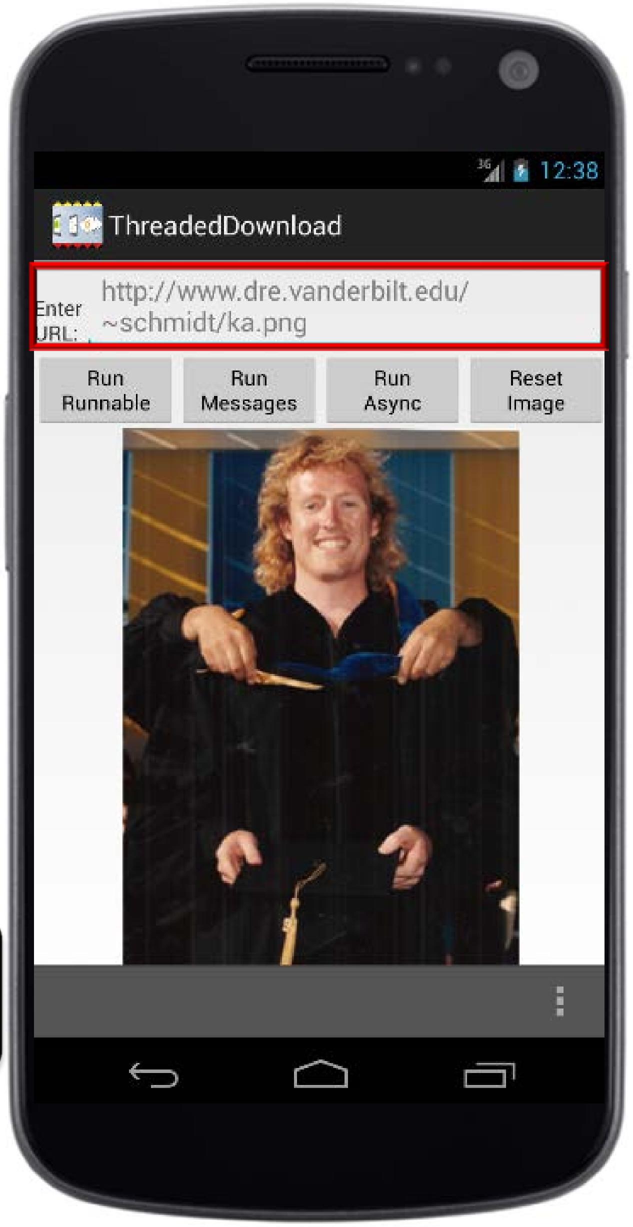
- Demonstrates multiple ways to download an image concurrently
  - Posting & processing Runnables
  - Sending & handling Messages
  - Executing AsyncTasks



Overview of the Image Downloads App

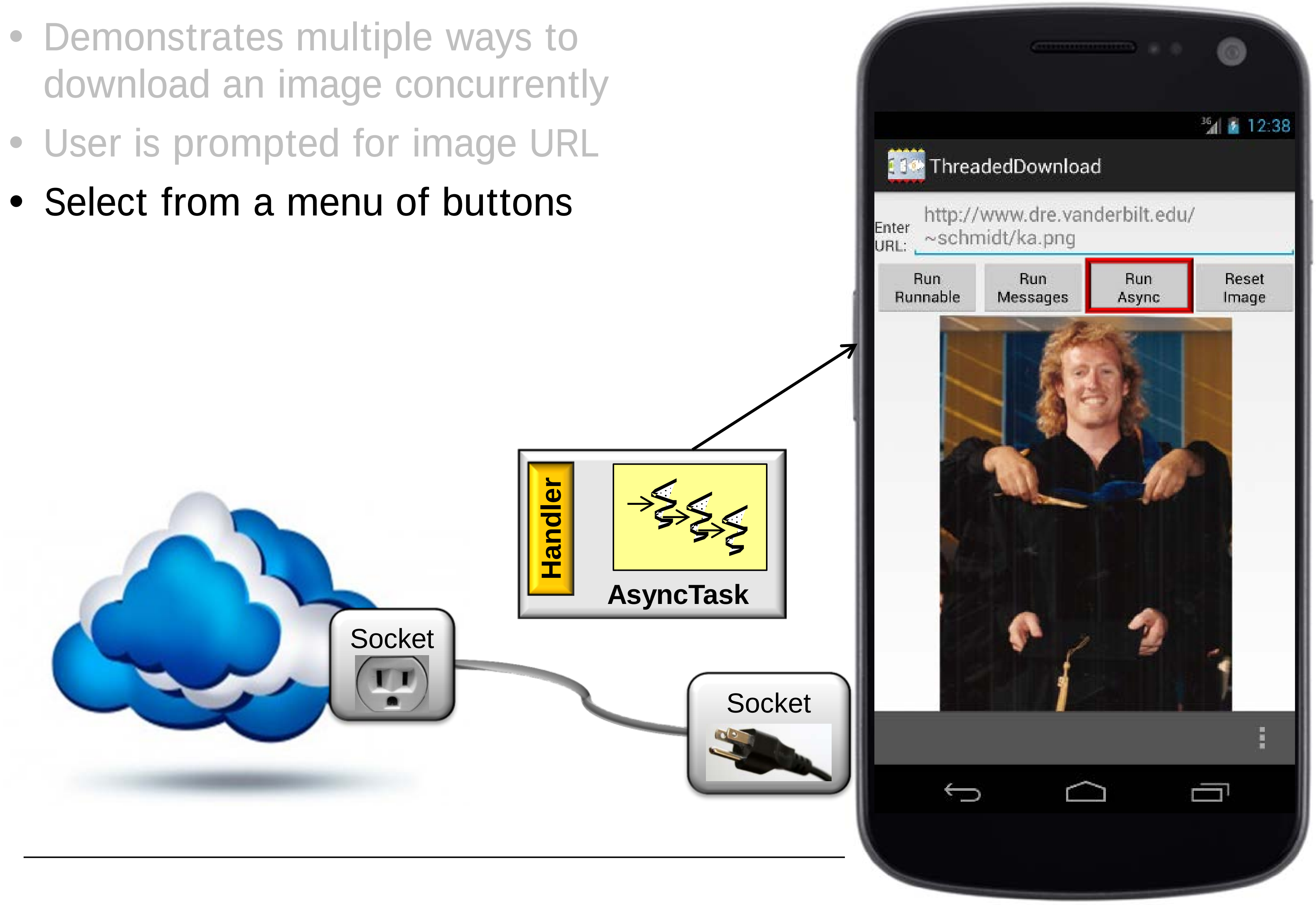
- Demonstrates multiple ways to download an image concurrently
- User is prompted for image URL





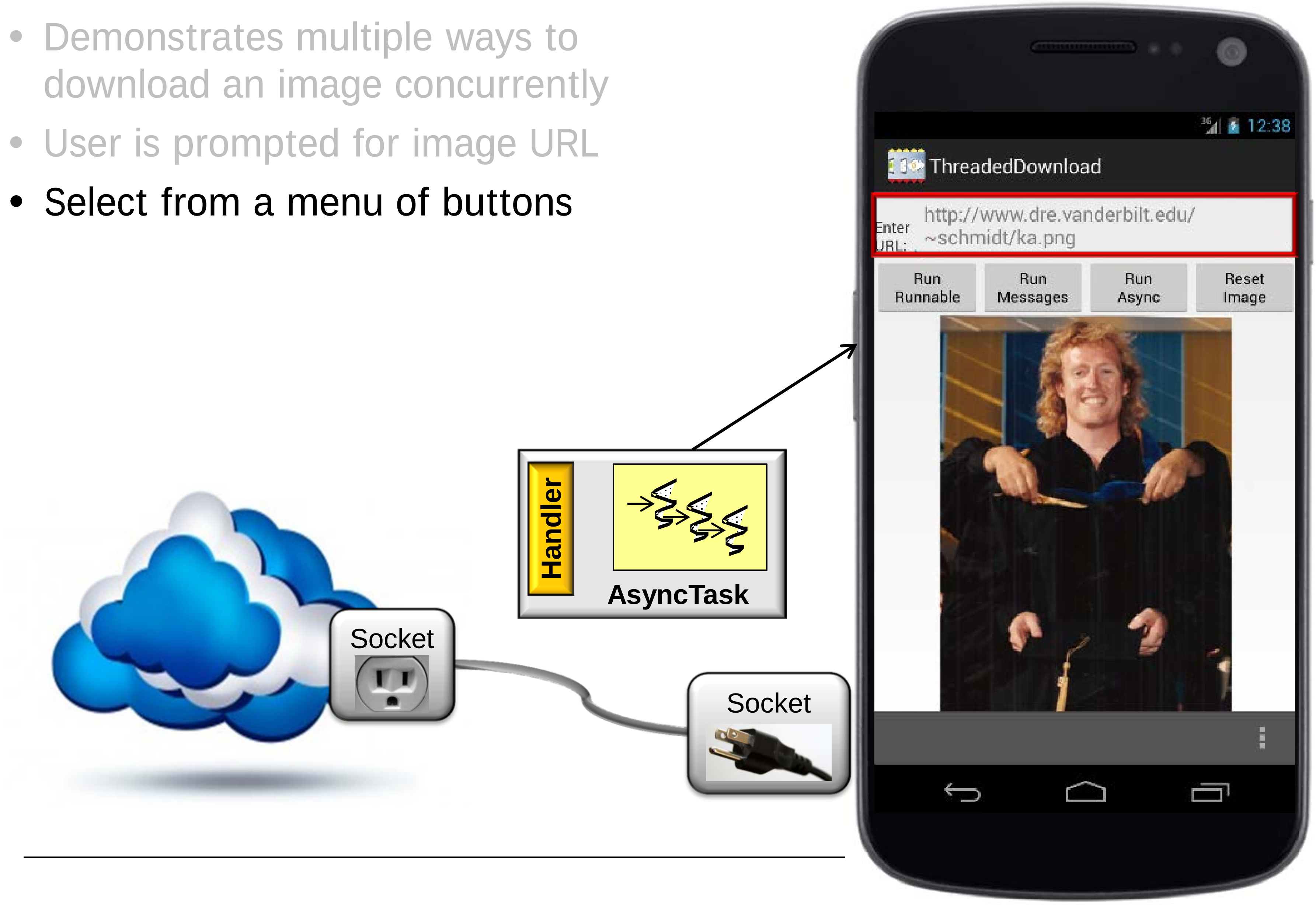
Overview of the Image Downloads App

- Demonstrates multiple ways to download an image concurrently
- User is prompted for image URL
- Select from a menu of buttons



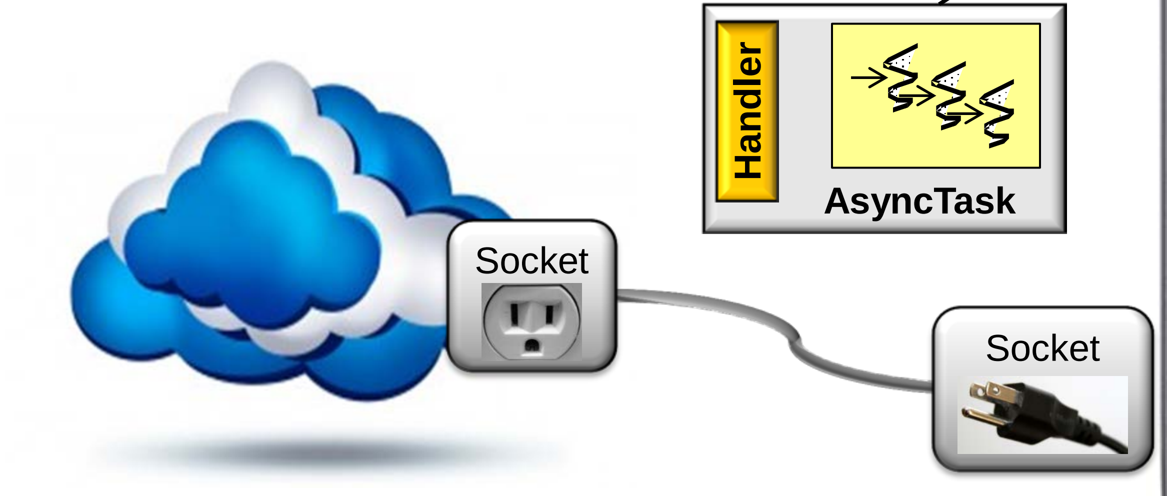
Overview of the Image Downloads App

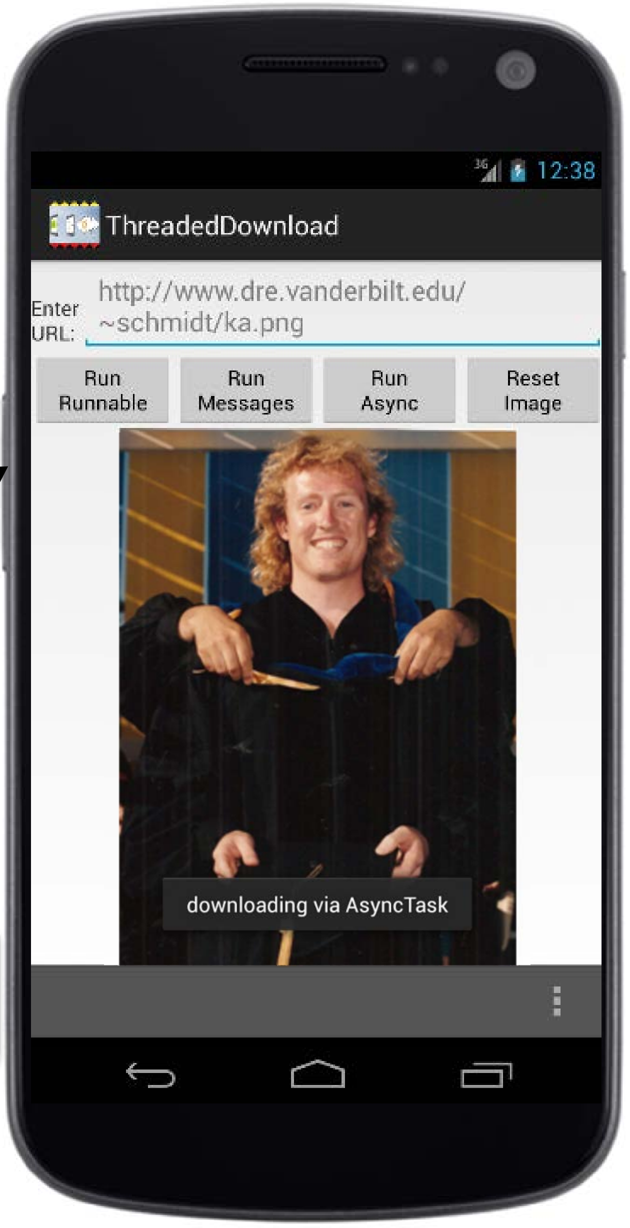
- Demonstrates multiple ways to download an image concurrently
- User is prompted for image URL
- Select from a menu of buttons



Overview of the Image Downloads App

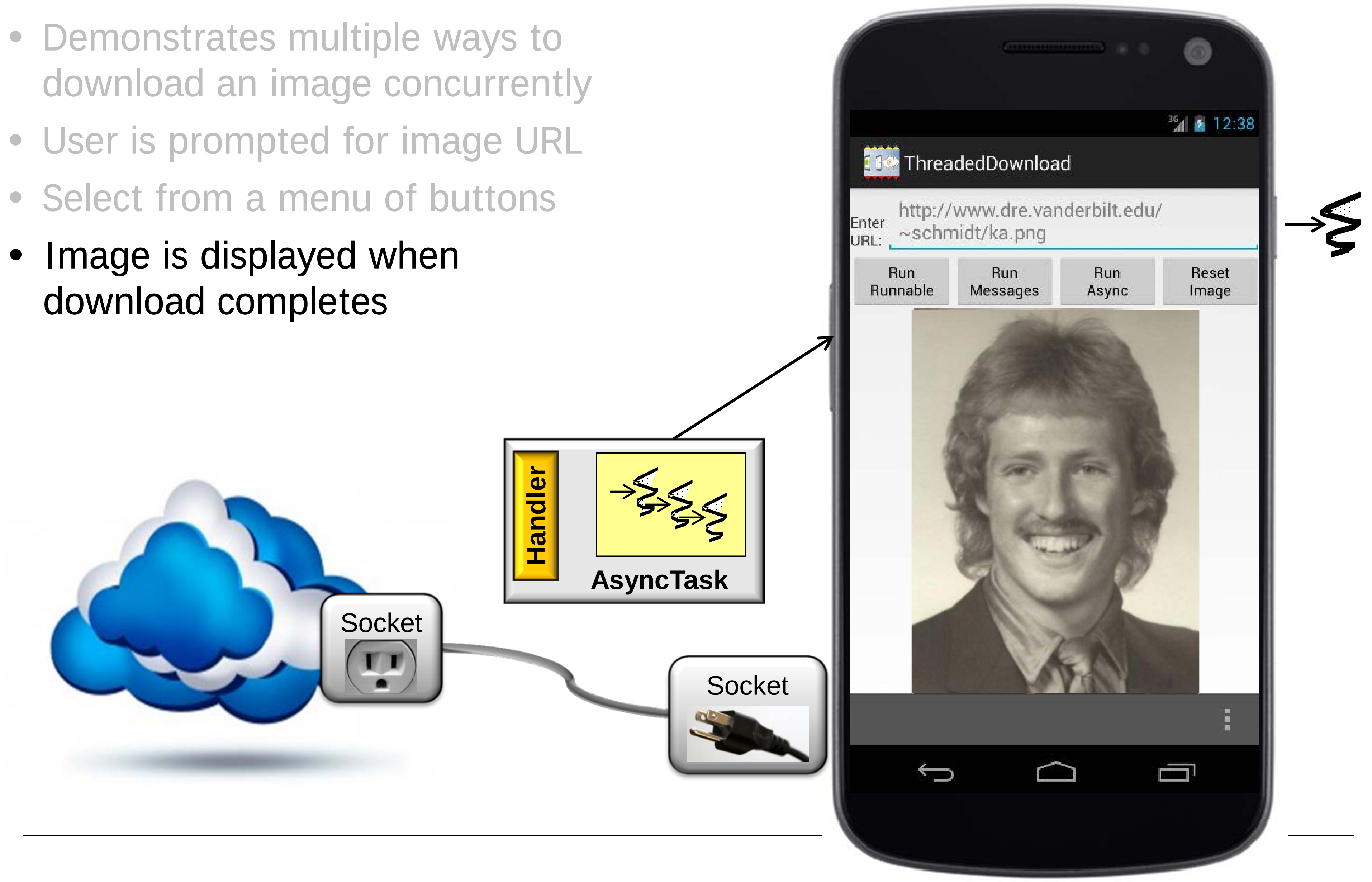
- Demonstrates multiple ways to download an image concurrently
- User is prompted for image URL
- Select from a menu of buttons
  - Toast is displayed when download begins





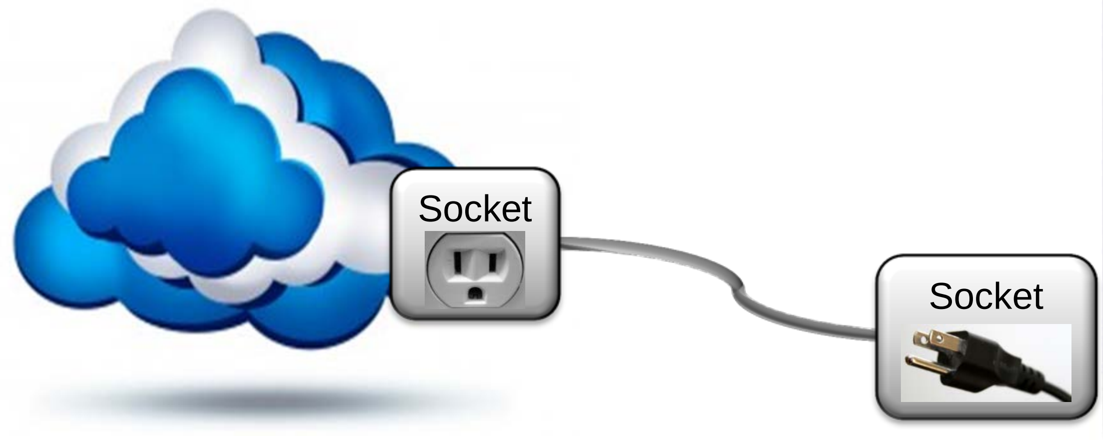
Overview of the Image Downloads App

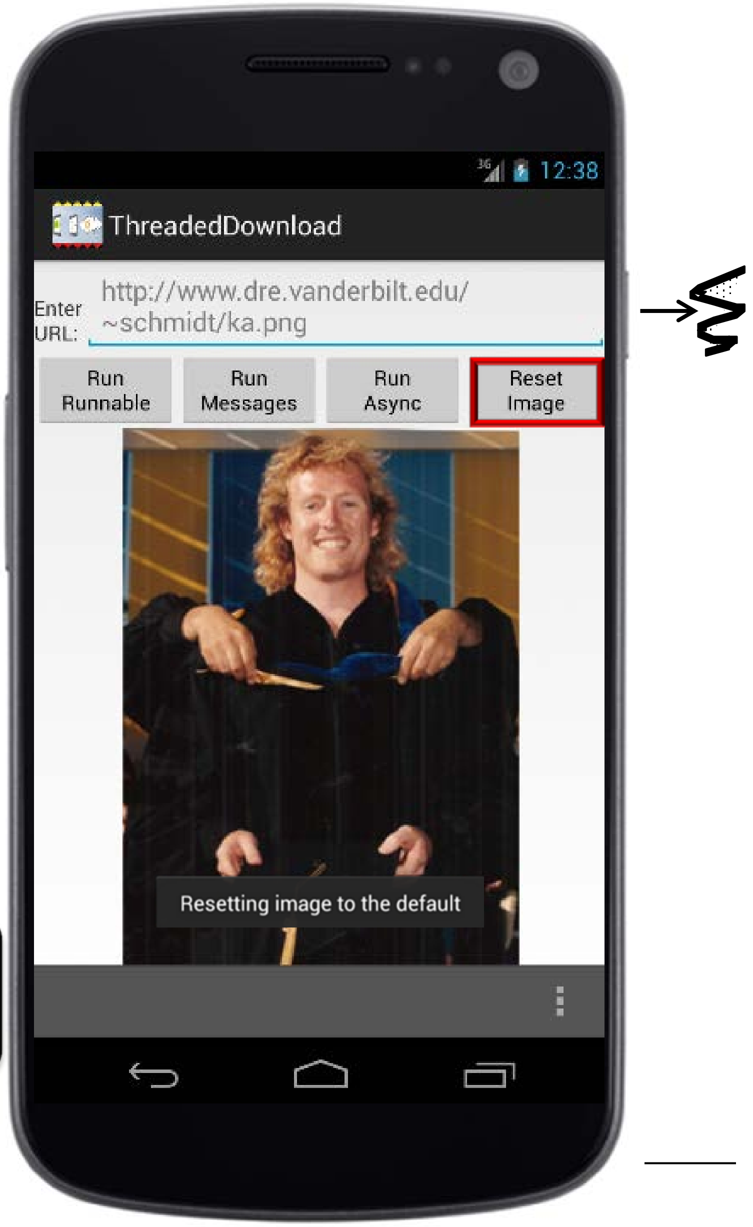
- Demonstrates multiple ways to download an image concurrently
- User is prompted for image URL
- Select from a menu of buttons
- Image is displayed when download completes



Overview of the Image Downloads App

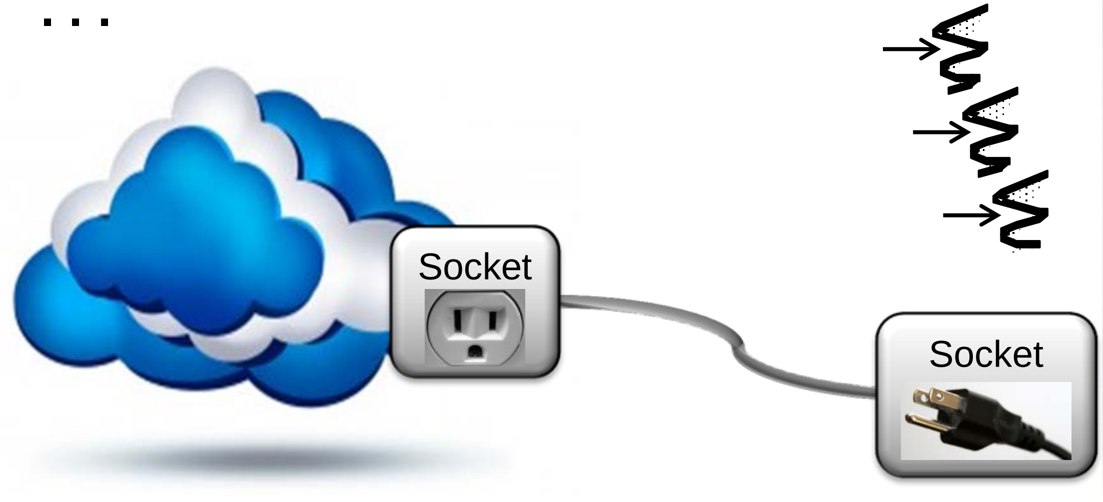
- Demonstrates multiple ways to download an image concurrently
- User is prompted for image URL
- Select from a menu of buttons
- Image is displayed when download completes
- Default image can be reset

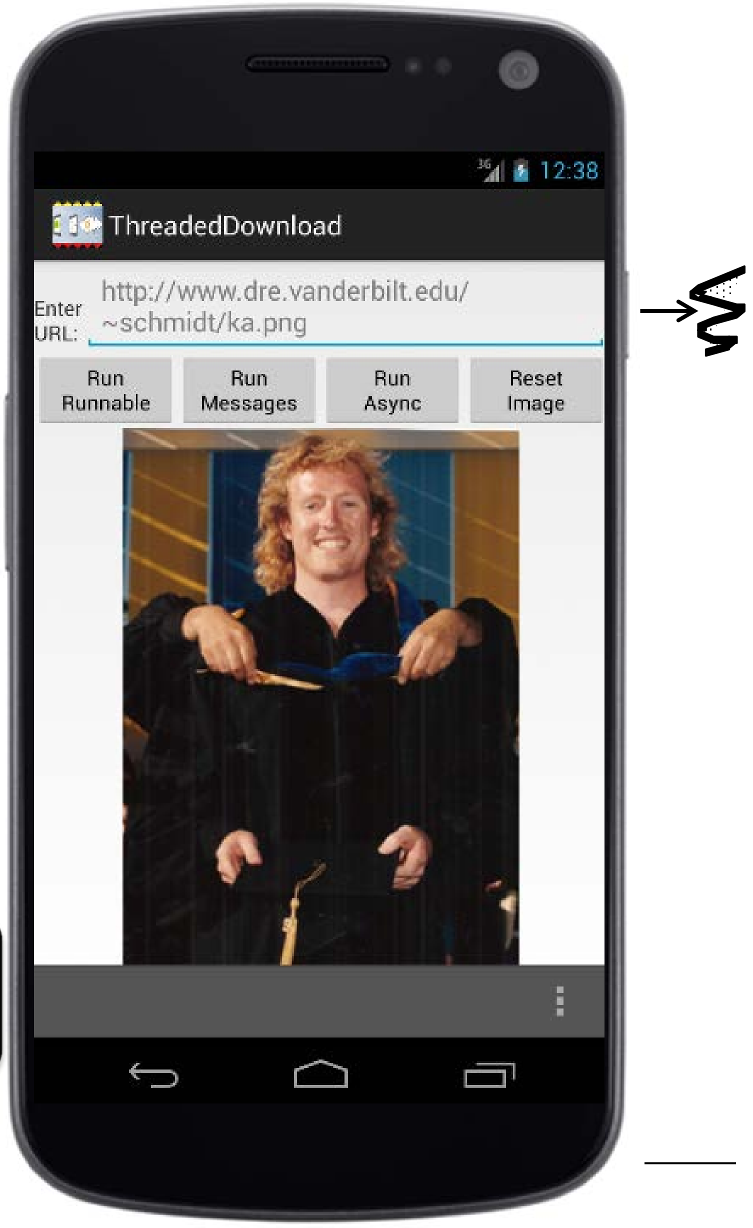



12:38
ThreadedDownload
Enter URL:
http://www.dre.vanderbilt.edu/~schmidt/ka.png
Run Runnable
Run Messages
Run Async
Reset Image
Resetting image to the default

Overview of the Image Downloads App

```
public class ImageDownloadsActivity
             extends Activity {
  ...
  public void runRunnable(View view) {
    /* App logic happens here */
  }

  public void runMessages(View view) {
    /* App logic happens here */
  }
  ...
```

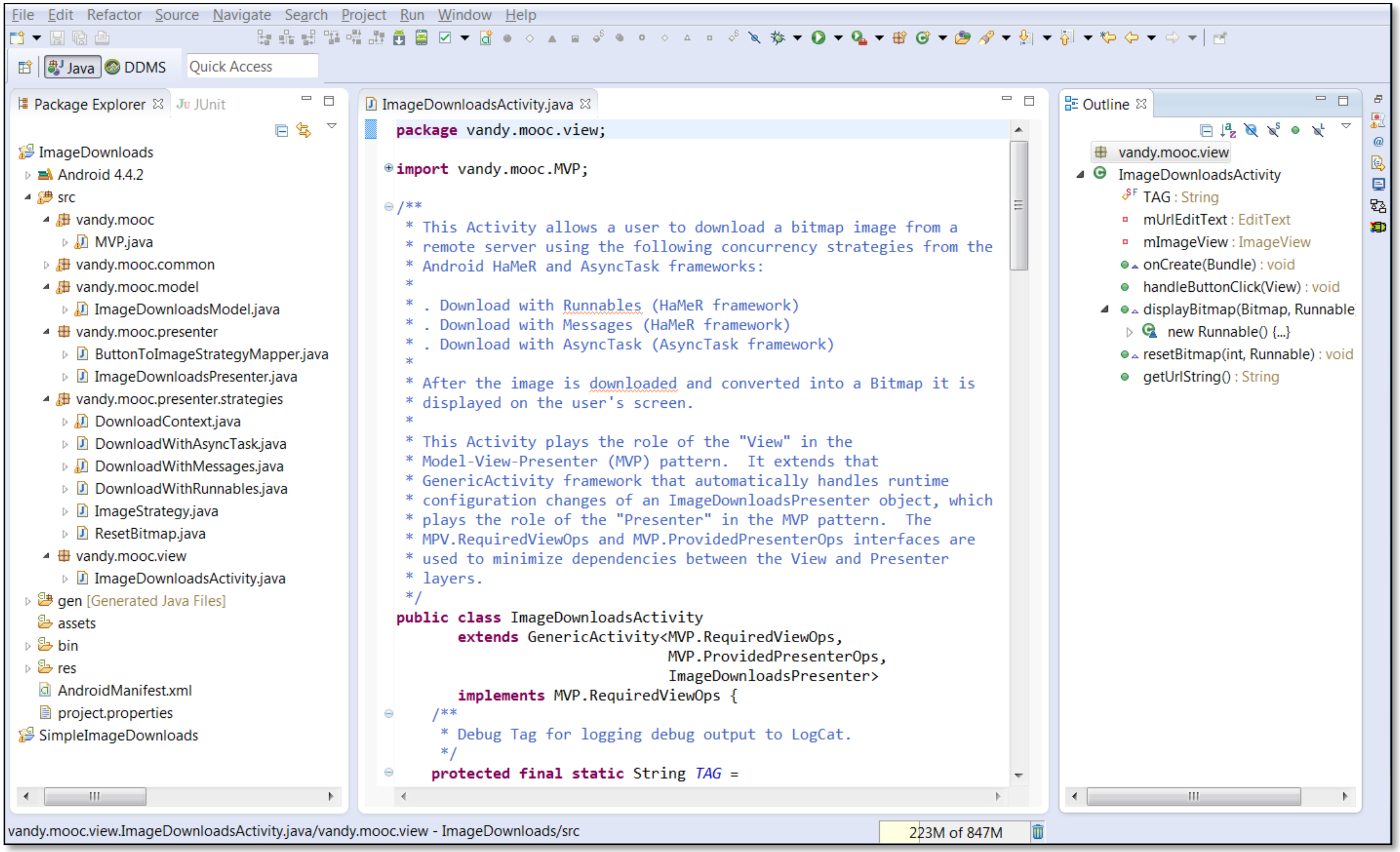




The Structure & Functionality of the Image Downloads Project

Image Downloads Structure & Functionality

- Image Downloads project


File Edit Refactor Source Navigate Search Project Run Window Help
Java DDMS Quick Access
Package Explorer JUnit
ImageDownloadsActivity.java
Outline

```
ImageDownloads
  Android 4.4.2
  src
    vandy.mooc
      MVP.java
    vandy.mooc.common
    vandy.mooc.model
      ImageDownloadsModel.java
    vandy.mooc.presenter
      ButtonToImageStrategyMapper.java
      ImageDownloadsPresenter.java
    vandy.mooc.presenter.strategies
      DownloadContext.java
      DownloadWithAsyncTask.java
      DownloadWithMessages.java
      DownloadWithRunnables.java
      ImageStrategy.java
      ResetBitmap.java
    vandy.mooc.view
      ImageDownloadsActivity.java
  gen [Generated Java Files]
  assets
  bin
  res
  AndroidManifest.xml
  project.properties
SimpleImageDownloads
```

```java
package vandy.mooc.view;

import vandy.mooc.MVP;

/**
 * This Activity allows a user to download a bitmap image from a
 * remote server using the following concurrency strategies from the
 * Android HaMeR and AsyncTask frameworks:
 *
 * . Download with Runnables (HaMeR framework)
 * . Download with Messages (HaMeR framework)
 * . Download with AsyncTask (AsyncTask framework)
 *
 * After the image is downloaded and converted into a Bitmap it is
 * displayed on the user's screen.
 *
 * This Activity plays the role of the "View" in the
 * Model-View-Presenter (MVP) pattern.  It extends that
 * GenericActivity framework that automatically handles runtime
 * configuration changes of an ImageDownloadsPresenter object, which
 * plays the role of the "Presenter" in the MVP pattern.  The
 * MPV.RequiredViewOps and MVP.ProvidedPresenterOps interfaces are
 * used to minimize dependencies between the View and Presenter
 * layers.
 */
public class ImageDownloadsActivity
       extends GenericActivity<MVP.RequiredViewOps,
                               MVP.ProvidedPresenterOps,
                               ImageDownloadsPresenter>
       implements MVP.RequiredViewOps {
    /**
     * Debug Tag for logging debug output to LogCat.
     */
    protected final static String TAG =
```

```
vandy.mooc.view
ImageDownloadsActivity
  TAG : String
  mUrlEditText : EditText
  mImageView : ImageView
  onCreate(Bundle) : void
  handleButtonClick(View) : void
  displayBitmap(Bitmap, Runnable
    new Runnable() {...}
  resetBitmap(int, Runnable) : void
  getUrlString() : String
```

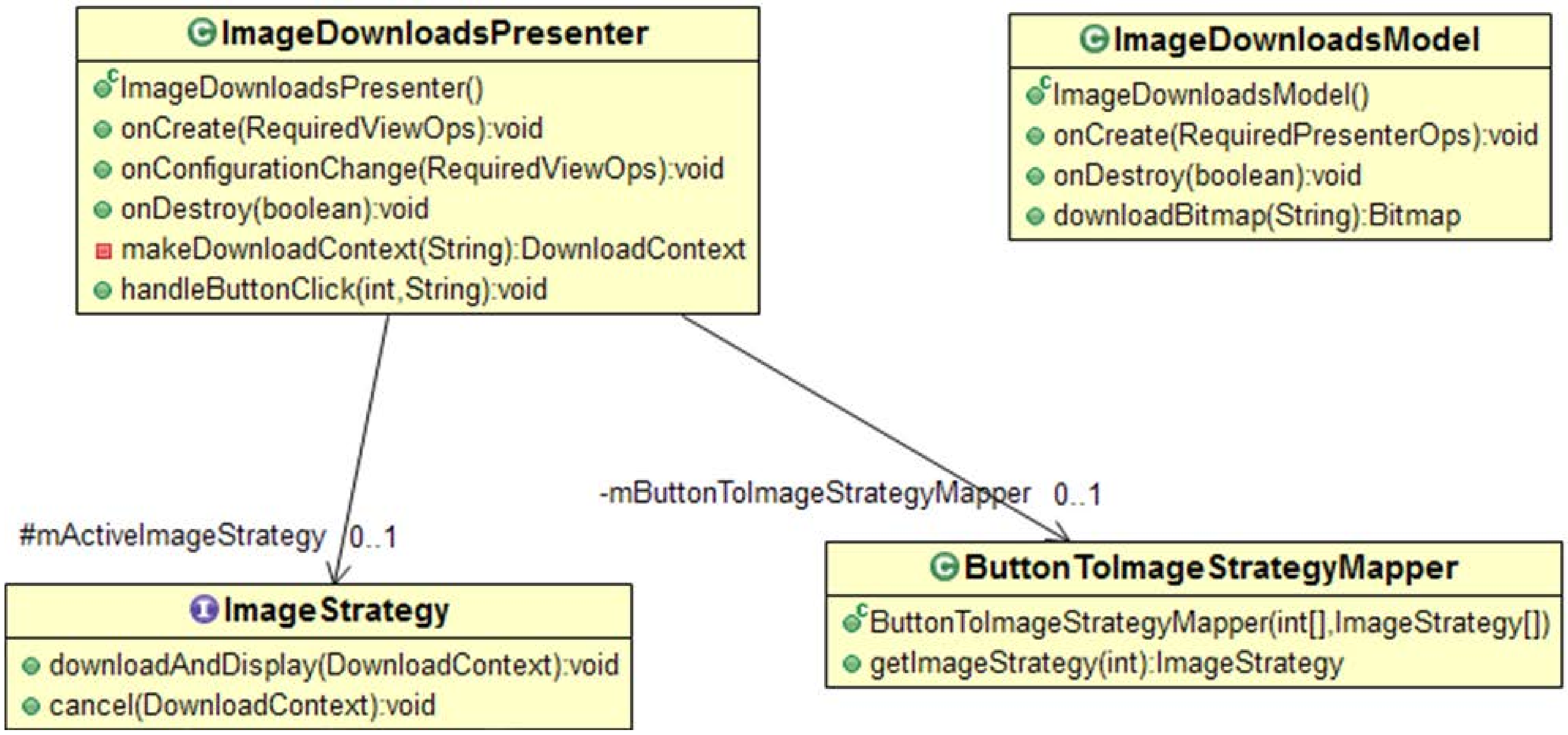
vandy.mooc.view.ImageDownloadsActivity.java/vandy.mooc.view - ImageDownloads/src   223M of 847M

Image Downloads Structure & Functionality

- Image Downloads project
- Three main elements:

Image Downloads Structure & Functionality

- Image Downloads project
- Three main elements:
  - Java source code


ImageDownloadsPresenter
ImageDownloadsPresenter()
onCreate(RequiredViewOps):void
onConfigurationChange(RequiredViewOps):void
onDestroy(boolean):void
makeDownloadContext(String):DownloadContext
handleButtonClick(int,String):void
ImageDownloadsModel
ImageDownloadsModel()
onCreate(RequiredPresenterOps):void
onDestroy(boolean):void
downloadBitmap(String):Bitmap
#mActiveImageStrategy 0..1
-mButtonToImageStrategyMapper 0..1
ImageStrategy
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
ButtonToImageStrategyMapper
ButtonToImageStrategyMapper(int[],ImageStrategy[])
getImageStrategy(int):ImageStrategy

The most creative & “free form” portion of an Android application

Image Downloads Structure & Functionality

- Image Downloads project
- Three main elements:
  - Java source code
  - Resources providing other files & static content used by Java code
    - e.g., bitmaps, UI layouts, internationalized strings, etc.

```
<LinearLayout
  android:layout_width=
    "fill_parent"
  android:layout_height=
    "wrap_content"
  android:orientation=
    "horizontal">
  <Button
    android:id="@+id/button1"
    android:layout_width=
      "wrap_content"
    android:layout_height=
      "wrap_content"
    android:onClick=
      “handleButtonClick"
    android:text=
      "@string/runRunnable" />
  ...
```

See developer.android.com/guide/topics/resources

Image Downloads Structure & Functionality

- Image Downloads project
- Three main elements:
  - Java source code
  - Resources providing other files & static content used by Java code
  - XML Manifest file containing info Android needs to execute the app

```
<manifest>  ...
  <application>
    <activity>
      <intent-filter>
       <action /> ... <data />
      </intent-filter> ...
    </activity>
    <service>
      <intent-filter> ...
      </intent-filter>
    </service>
    <receiver>
      <intent-filter> ...
      </intent-filter>
    </receiver>
    <provider>
      <grant-uri-permission />
    </provider> ...
```

See developer.android.com/guide/topics/manifest/manifest-intro.html

Image Downloads Structure & Functionality

- Image Downloads project
- Three main elements
- XML Manifest file for Image Downloads app contains essential info

```
<manifest ...
  package="vandy.mooc"
  ...
```

Image Downloads Structure & Functionality

- Image Downloads project
- Three main elements
- XML Manifest file for Image Downloads app contains essential info
  - Grants permission to use the Internet

```
<manifest ...
  package="vandy.mooc"
  <uses-permission
   android:name=
     "android.permission.INTERNET">
  </uses-permission>
  ...
```

Android M now gives Internet permission to all apps by default

Image Downloads Structure & Functionality

- Image Downloads project
- Three main elements
- XML Manifest file for Image Downloads app contains essential info
  - Grants permission to use the Internet
  - Indicates the main Activity & declares with Intents it handles

```
<manifest ...
  package="vandy.mooc"
  <application
    android:icon=
      "@drawable/ic_launcher"
    android:label=
      "@string/app_name"
    ...
    <activity android:name=
      "view.ImageDownloadsActivity"
      <intent-filter>
        <action android:name=
          "android.intent.
           action.MAIN" />
        ...
      </intent-filter>
    </activity>
  </application> ...
```

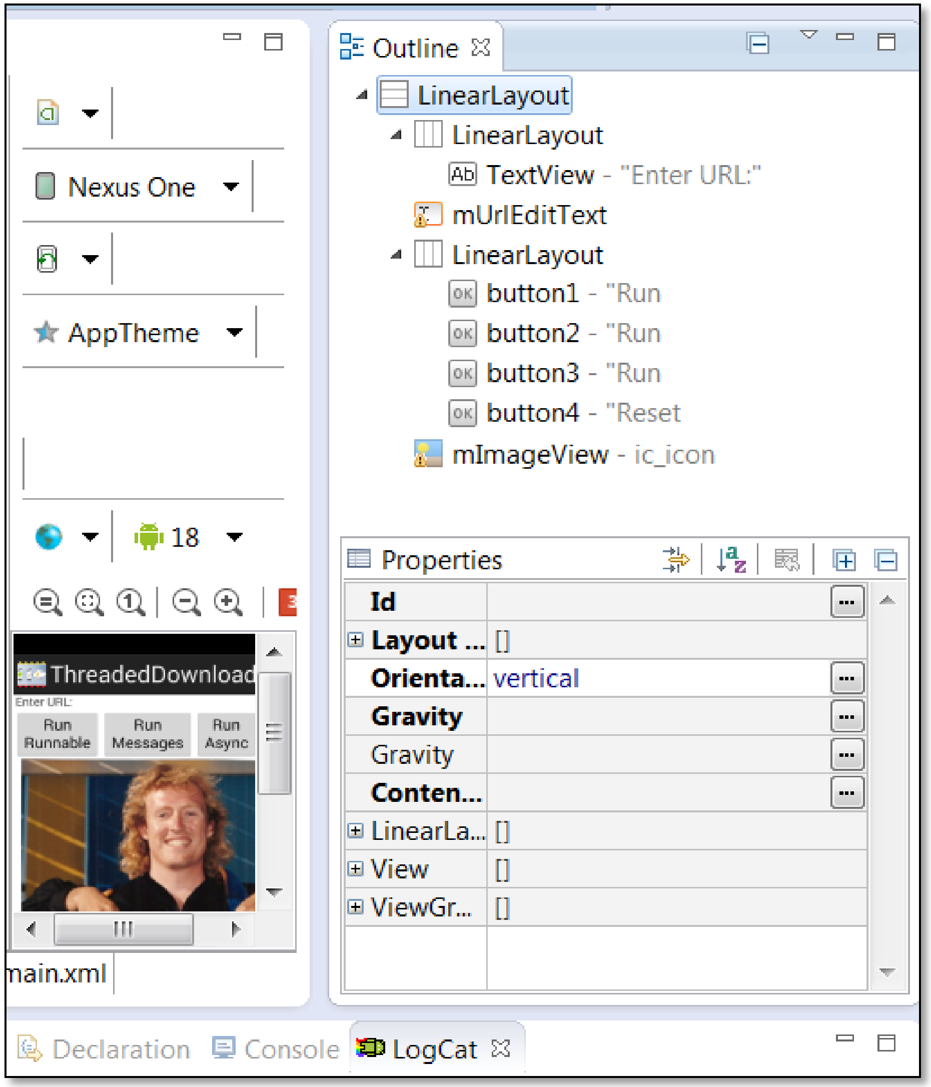
Image Downloads Structure & Functionality

- Image Downloads project
- Three main elements
- XML Manifest file for Image Downloads app contains essential info
  - Grants permission to use the Internet
  - Indicates the main Activity & declares with Intents it handles

```
<manifest ...
  package="vandy.mooc"
  <application
    android:icon=
      "@drawable/ic_launcher"
    android:label=
      "@string/app_name"
    ...
    <activity android:name=
      "view.ImageDownloadsActivity"
      <intent-filter>
        <action android:name=
          "android.intent.
           action.MAIN" />
        ...
      </intent-filter>
    </activity>
  </application> ...
```

Image Downloads Structure & Functionality

- Image Downloads project
- Three main elements
- XML Manifest file for Image Downloads app contains essential info
- The image_downloads_activity.xml resource file specifies app layout



```
LinearLayout
  LinearLayout
    TextView - "Enter URL:"
  mUrlEditText
  LinearLayout
    button1 - "Run
    button2 - "Run
    button3 - "Run
    button4 - "Reset
  mImageView - ic_icon
```

Id	
Layout ...	[]
Orienta...	vertical
Gravity	
Gravity	
Conten...	
LinearLa...	[]
View	[]
ViewGr...	[]

Image Downloads Structure & Functionality

- Image Downloads project
- Three main elements
- XML Manifest file for Image Downloads app contains essential info
- The image_downloads_activity.xml resource file specifies app layout
  - Dictates how text & buttons appears on the screen

```
<TextView
  android:layout_width=
   "wrap_content"
  android:layout_height=
   "wrap_content"
  android:text="@string/location“
   ... />

<EditText
  android:id="@+id/mUrlEditText"
  android:layout_height=
   "wrap_content"
  android:hint="@string/defaultURL“
  .../>

<Button
  android:id="@+id/button1“
  ...
```

Image Downloads Structure & Functionality

- Image Downloads project
- Three main elements
- XML Manifest file for Image Downloads app contains essential info
- The image_downloads_activity.xml resource file specifies app layout
  - Dictates how text & buttons appears on the screen
  - Maps methods to buttons

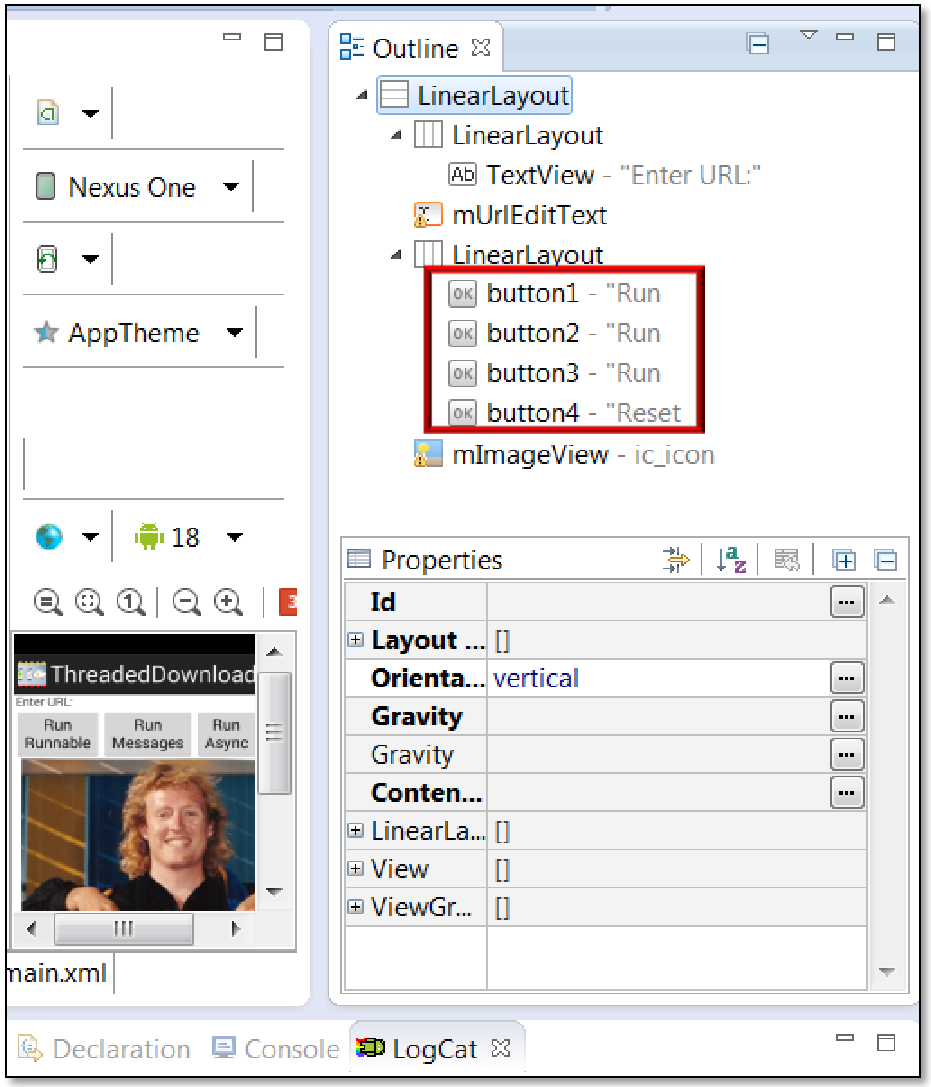


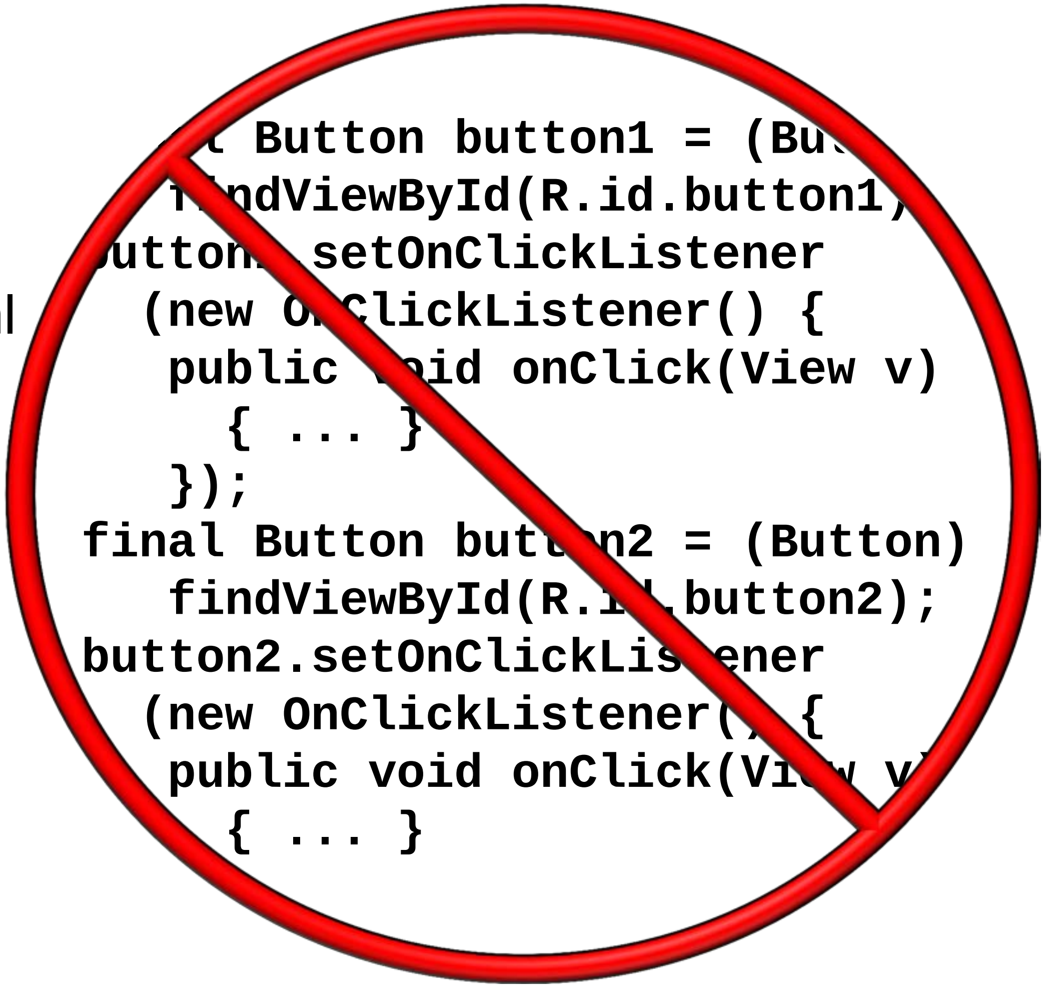
Image Downloads Structure & Functionality

- Image Downloads project
- Three main elements
- XML Manifest file for Image Downloads app contains essential info
- The image_downloads_activity.xml resource file specifies app layout
  - Dictates how text & buttons appears on the screen
  - Maps methods to buttons

```
<Button
  ...
  android:onClick="handleButtonClick"
  android:text="@string/runRunnable" />

<Button
  ...
  android:onClick="handleButtonClick"
  android:text="@string/runMessages" />

<Button
  ...
  android:onClick="handleButtonClick"
  android:text="@string/runAsyncTask" />

<Button
  ...
  android:onClick="handleButtonClick"
  android:text="@string/resetImage" />
```

Image Downloads Structure & Functionality

- Image Downloads project
- Three main elements
- XML Manifest file for Image Downloads app contains essential info
- The image_downloads_activity.xml resource file specifies app layout
  - Dictates how text & buttons appears on the screen
  - Maps methods to buttons
  - Avoids hard-coding UI components into class ImageDownloadsActivity



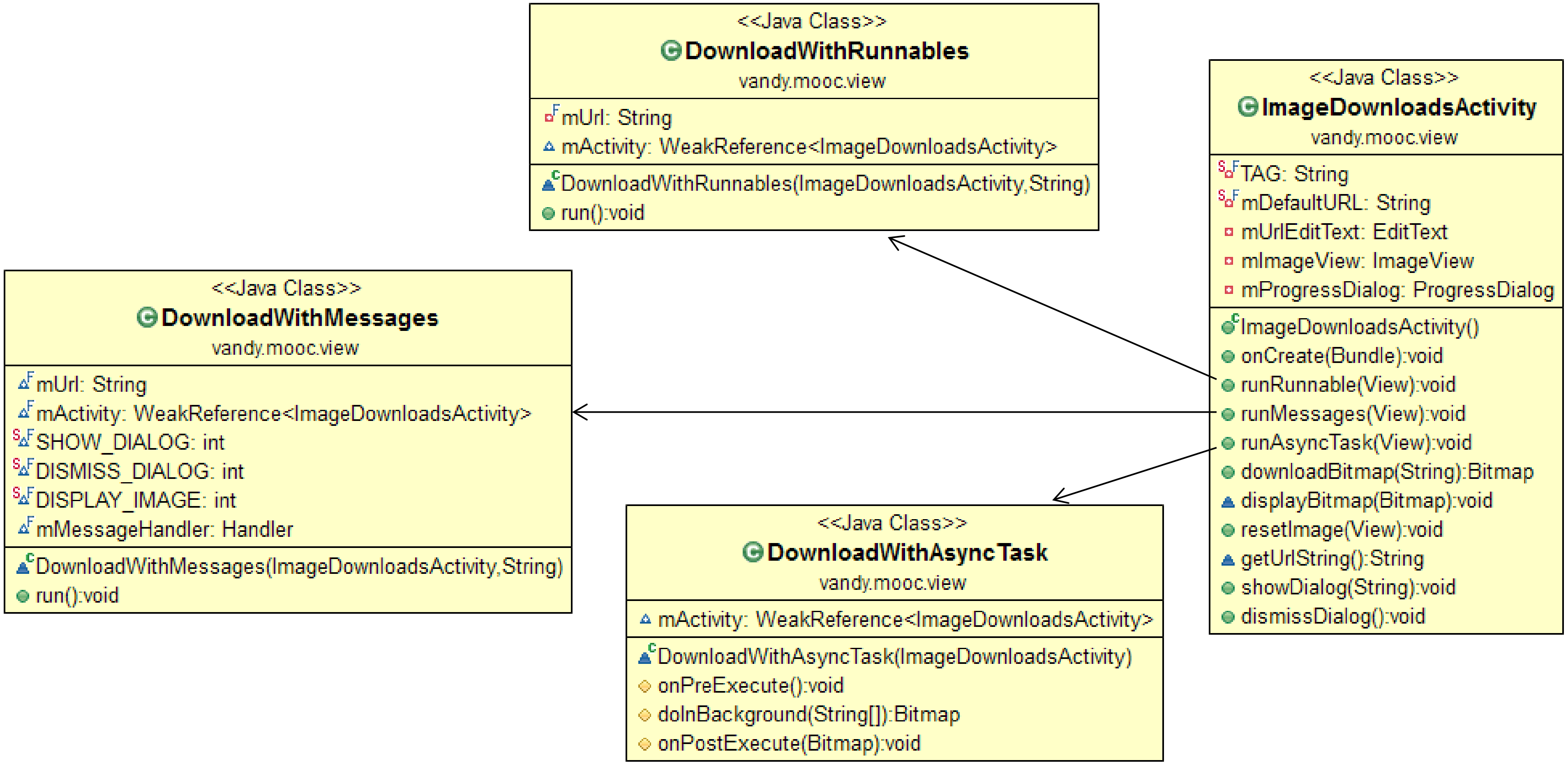
```
  t Button button1 = (Bu
    ndViewById(R.id.button1)
utton .setOnClickListener
  (new O ClickListener() {
    public  oid onClick(View v)
      { ... }
    });
final Button butt n2 = (Button)
    findViewById(R.i .button2);
button2.setOnClickLis ener
  (new OnClickListener( ) {
    public void onClick(Vi   v
      { ... }
```

There are drawbacks to this approach in more complex apps

Overview of Image Downloads Apps

Overview of SimpleImageDownloads

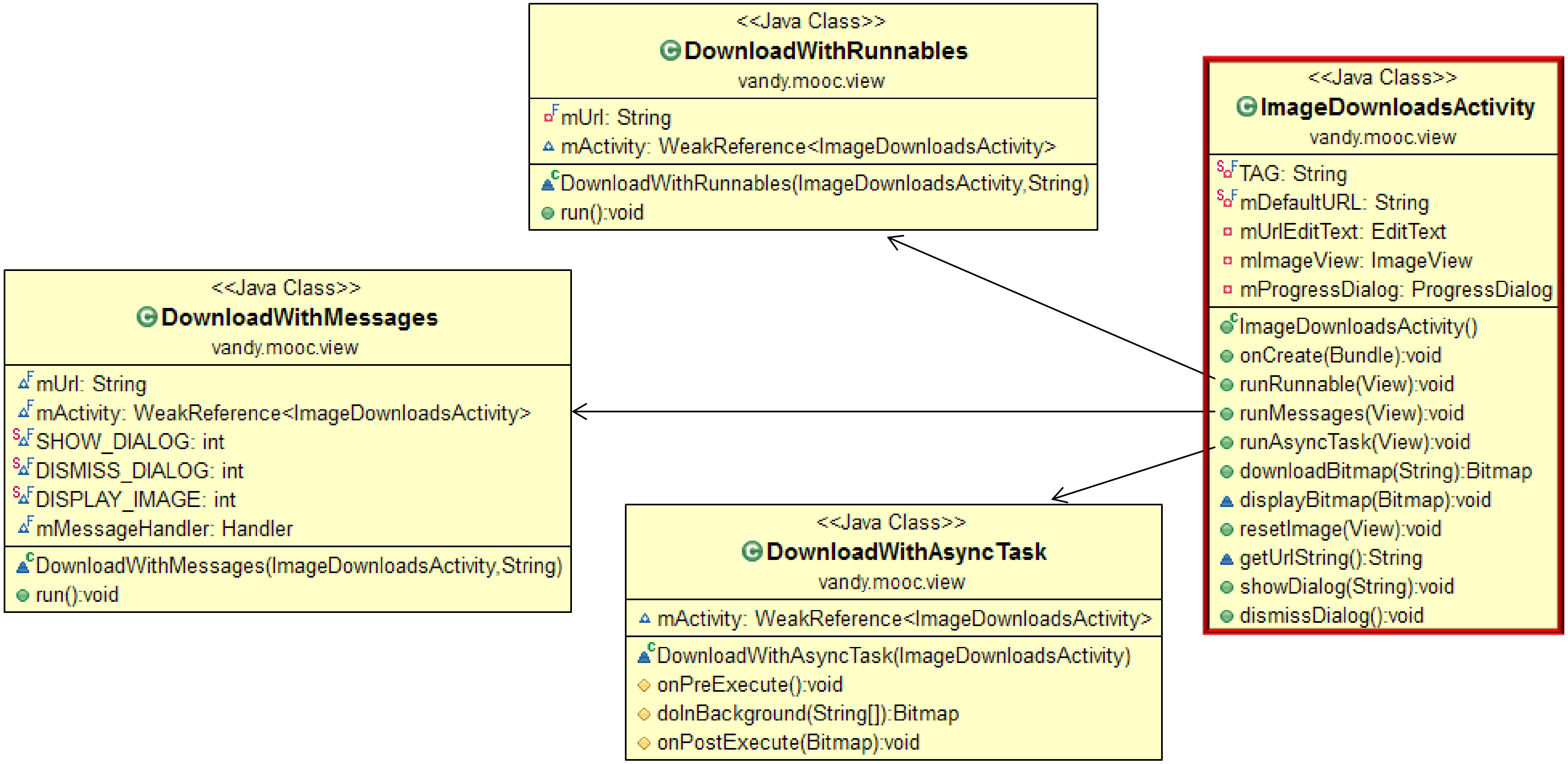
- Simple app showcasing Android HaMeR & AsyncTask concurrency frameworks



See github.com/douglascraigschmidt/POSA-15/tree/master/ex/SimpleImageDownloads

Overview of SimpleImageDownloads

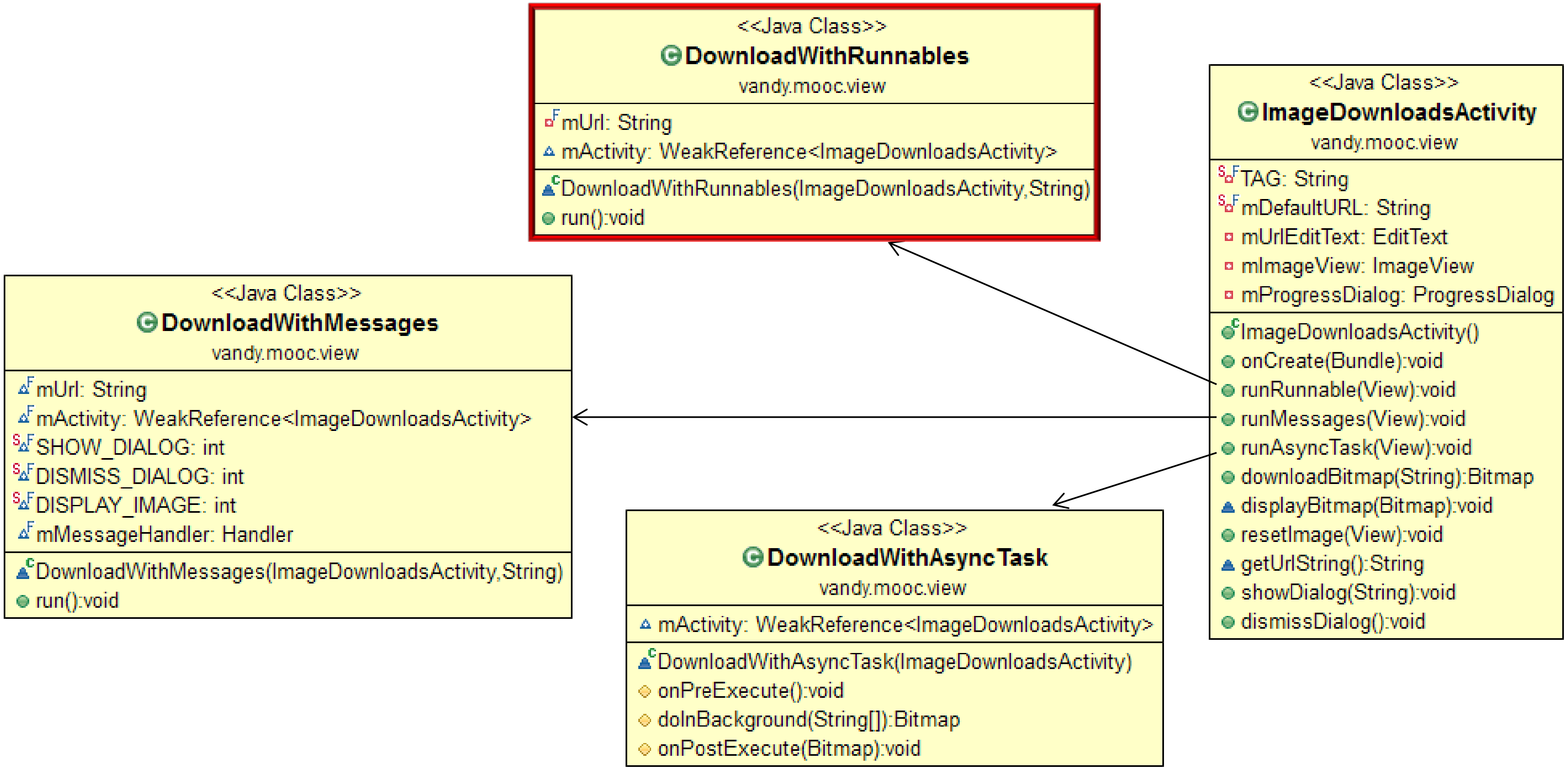
- Simple app showcasing Android HaMeR & AsyncTask concurrency frameworks



ImageDownloadsActivity is the only Activity defined in the app

Overview of SimpleImageDownloads

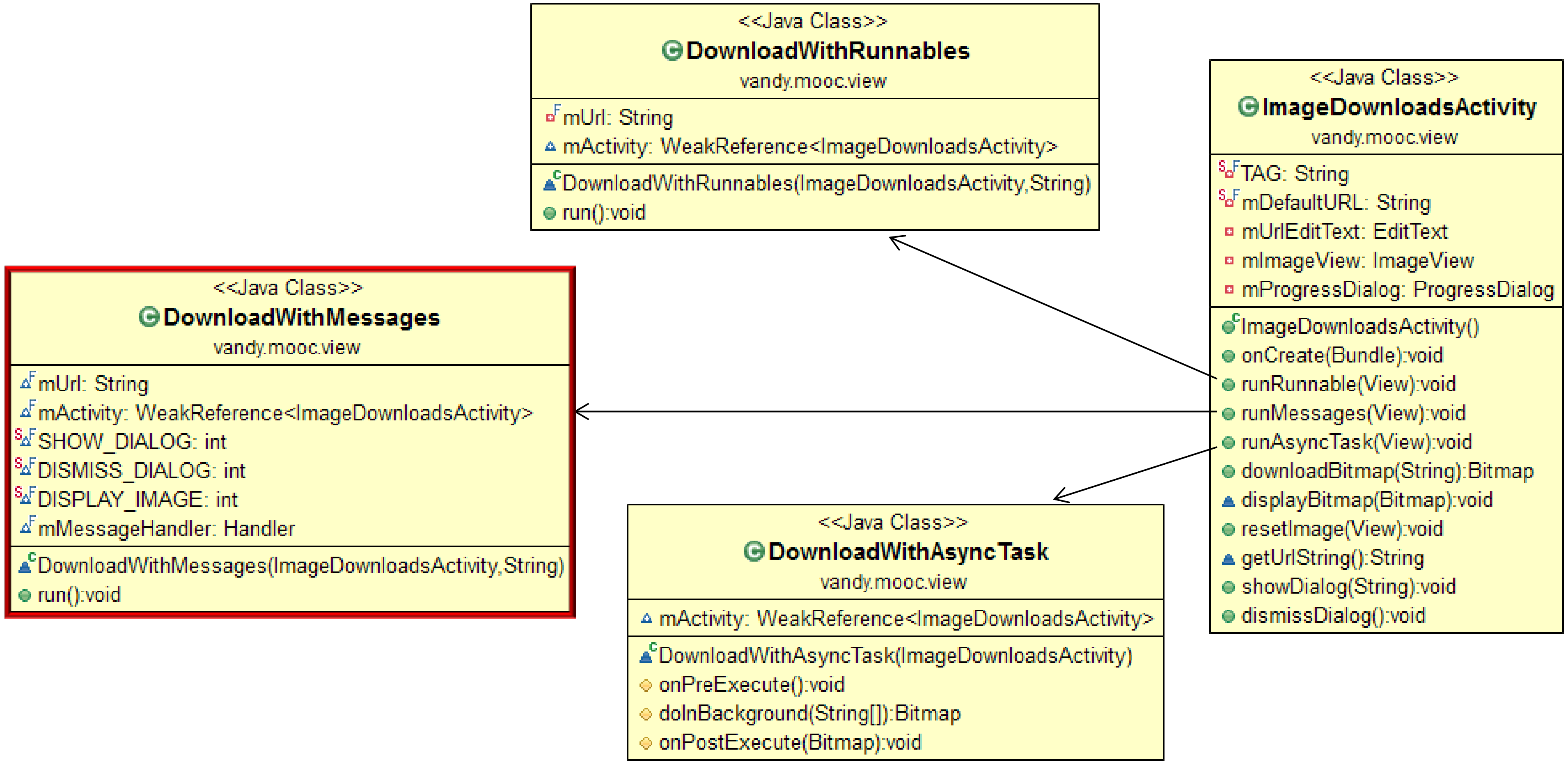
- Simple app showcasing Android HaMeR & AsyncTask concurrency frameworks



Downloads/Displays image via Handlers & Runnables

Overview of SimpleImageDownloads

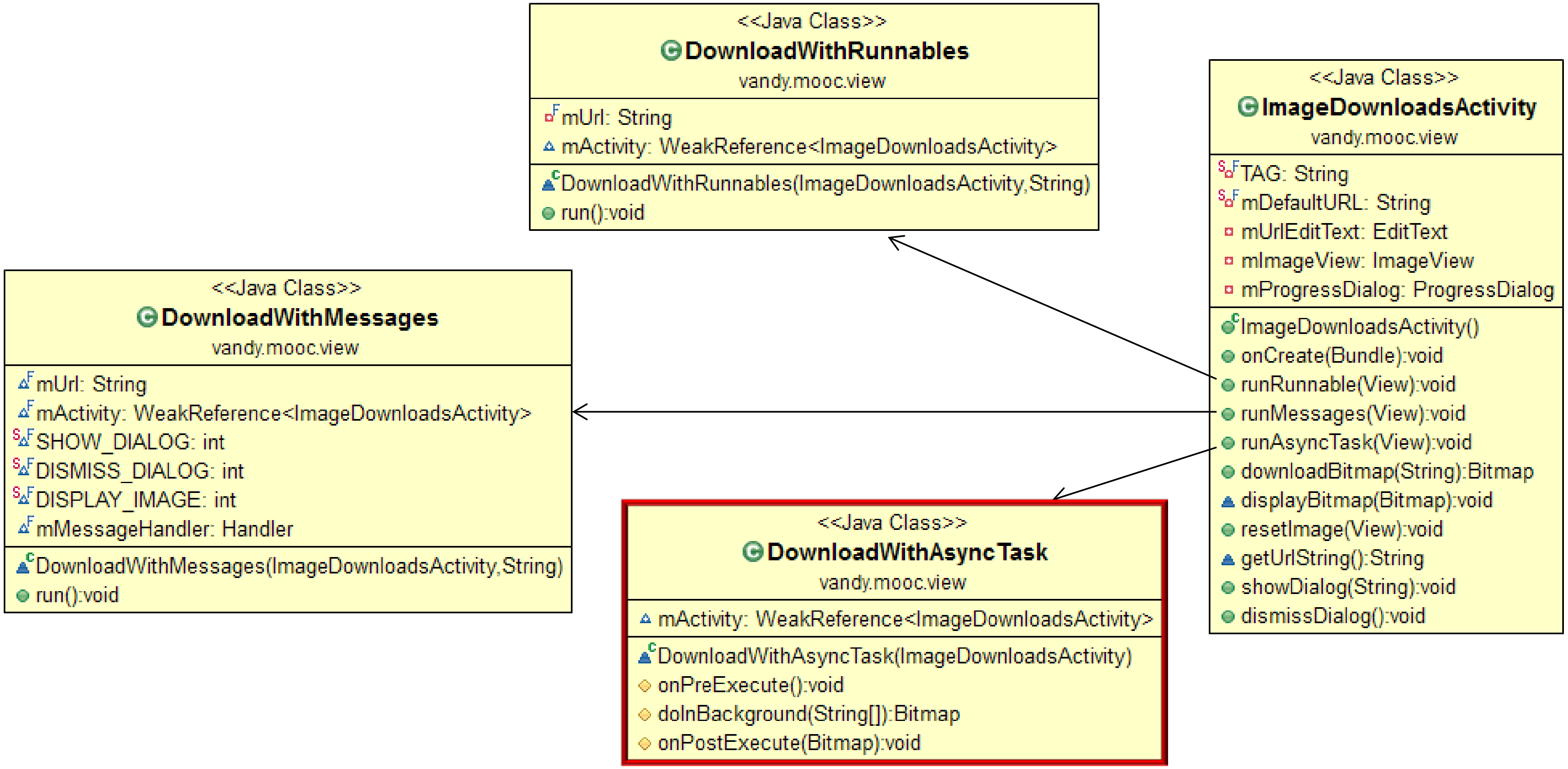
- Simple app showcasing Android HaMeR & AsyncTask concurrency frameworks



Downloads/Displays image via Handlers & Messages

Overview of SimpleImageDownloads

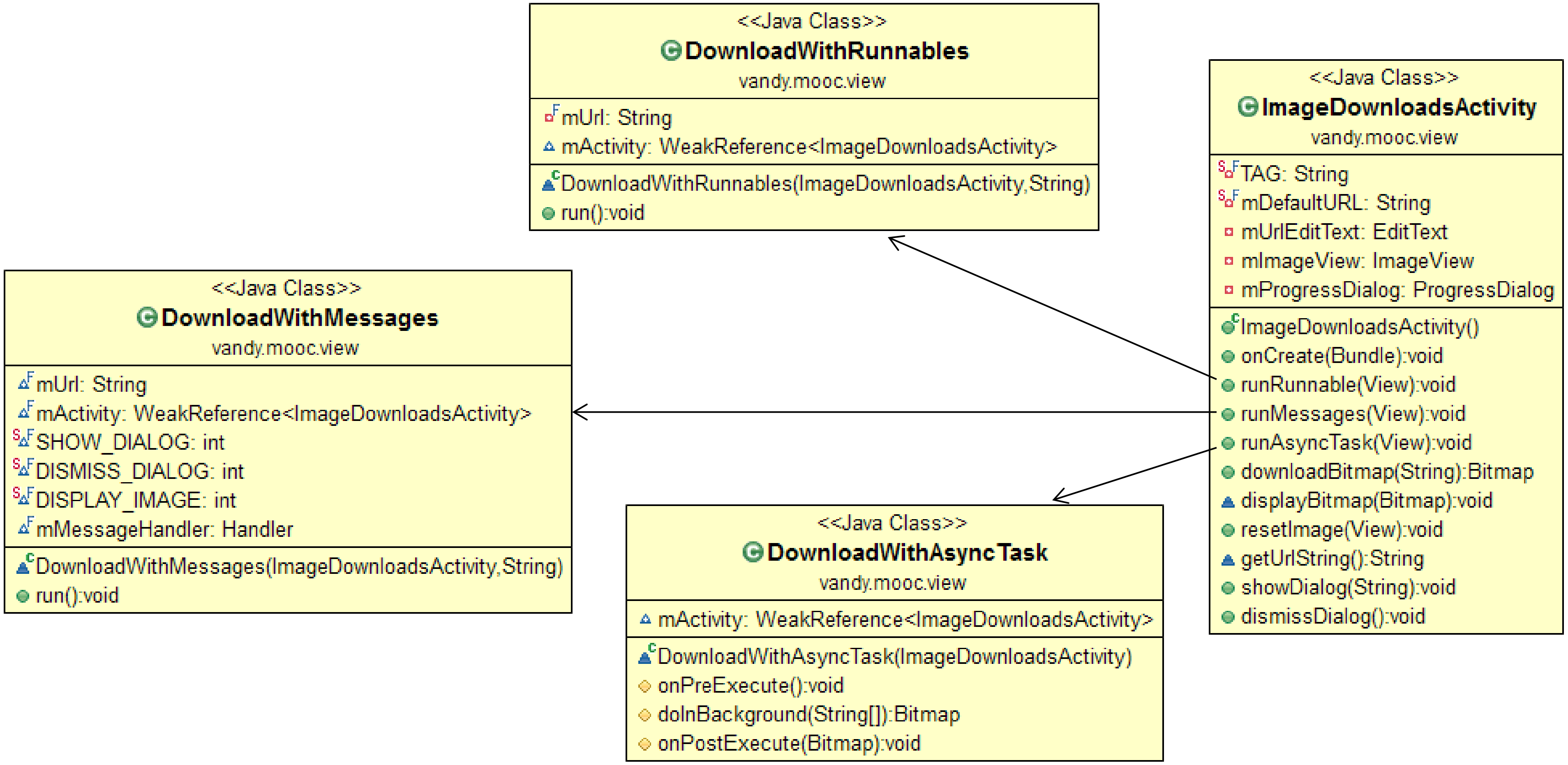
- Simple app showcasing Android HaMeR & AsyncTask concurrency frameworks



Downloads/Displays image via AsyncTask

Overview of SimpleImageDownloads

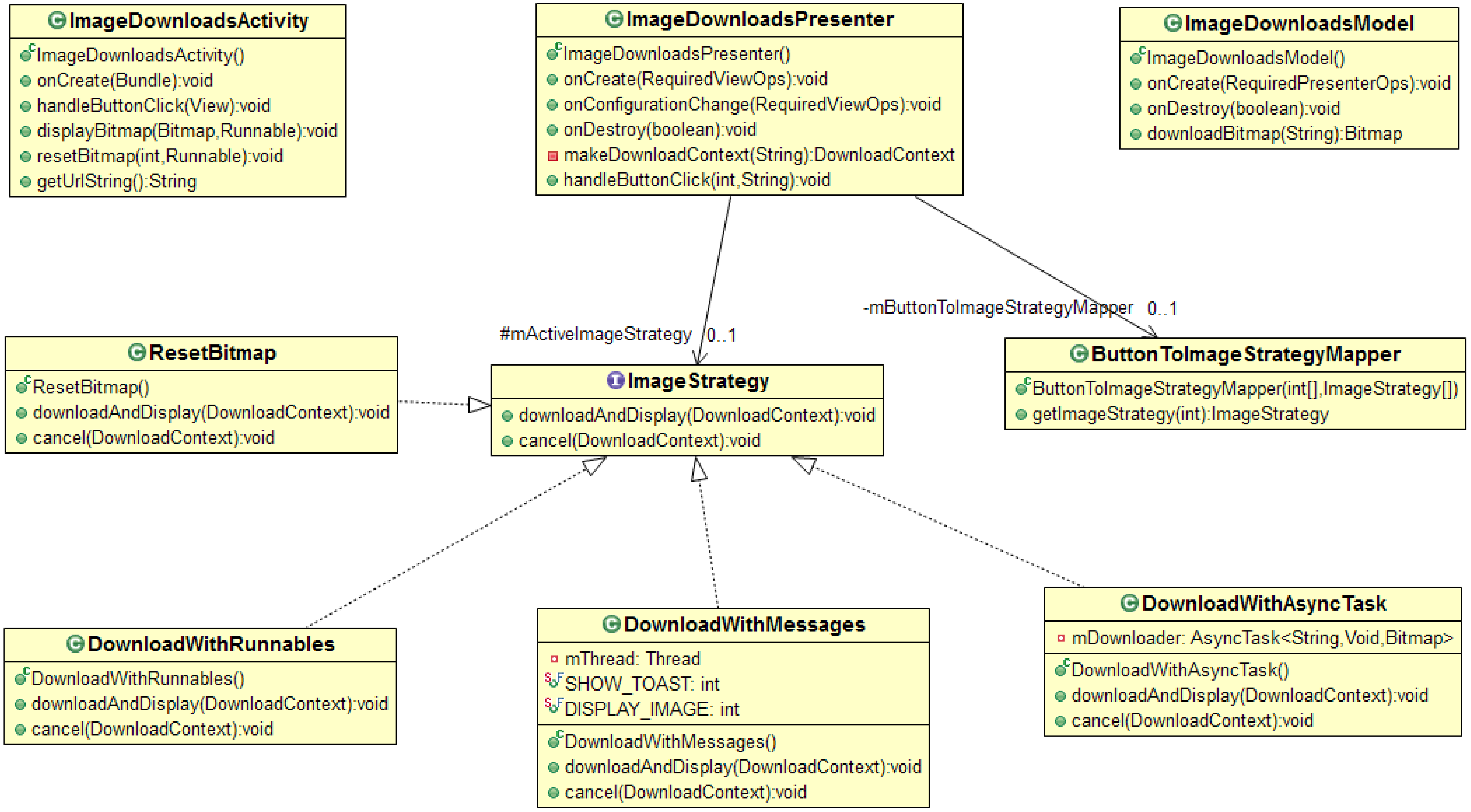
- Simple app showcasing Android HaMeR & AsyncTask concurrency frameworks



This solution doesn't handle runtime configuration changes

Overview of ImageDownloads

- This implementation is based on the *Model-View-Presenter* (MVP) pattern

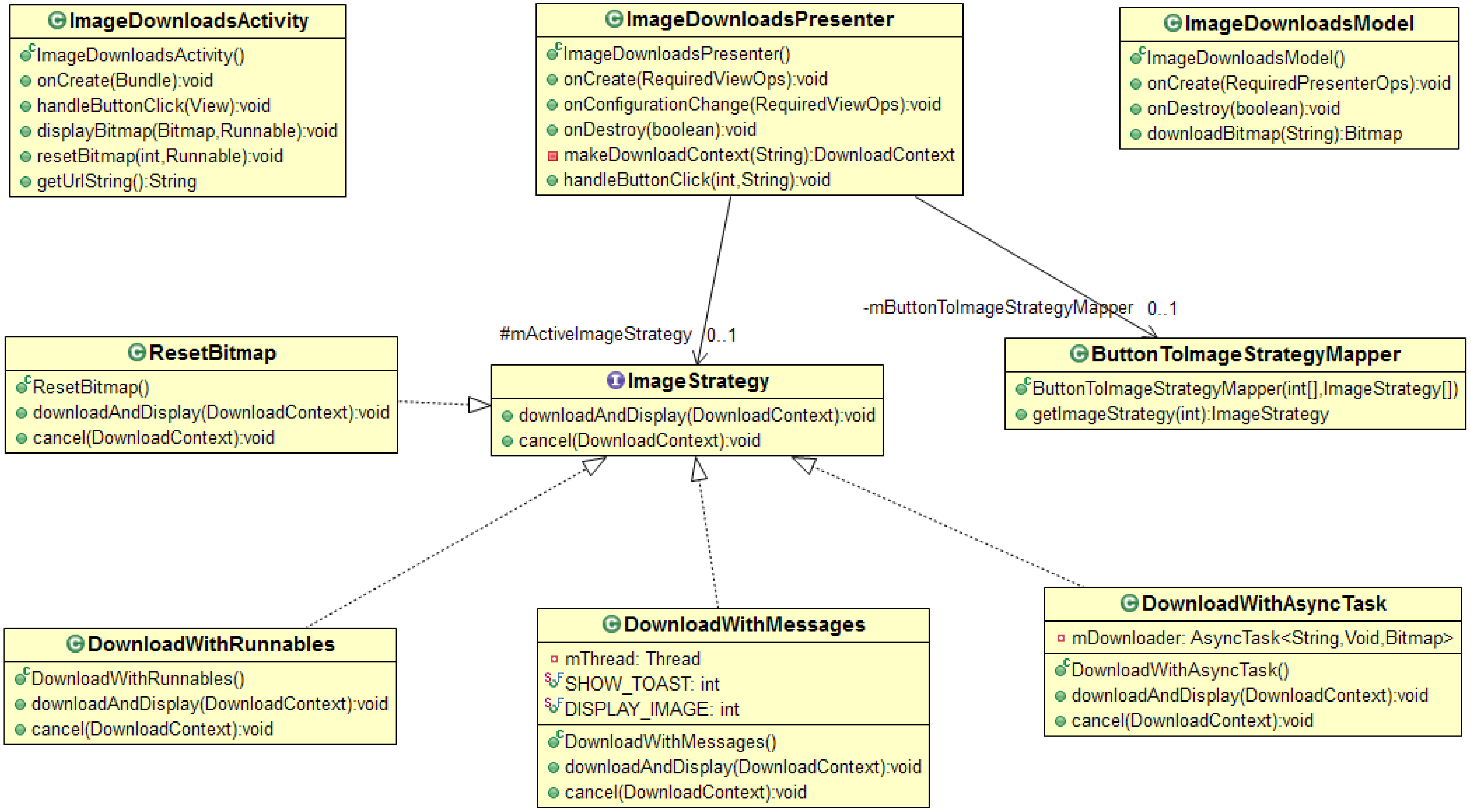


ImageDownloadsActivity
ImageDownloadsActivity()
onCreate(Bundle):void
handleButtonClick(View):void
displayBitmap(Bitmap,Runnable):void
resetBitmap(int,Runnable):void
getUrlString():String
ImageDownloadsPresenter
ImageDownloadsPresenter()
onCreate(RequiredViewOps):void
onConfigurationChange(RequiredViewOps):void
onDestroy(boolean):void
makeDownloadContext(String):DownloadContext
handleButtonClick(int,String):void
ImageDownloadsModel
ImageDownloadsModel()
onCreate(RequiredPresenterOps):void
onDestroy(boolean):void
downloadBitmap(String):Bitmap
#mActiveImageStrategy
0..1
-mButtonToImageStrategyMapper
0..1
ResetBitmap
ResetBitmap()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
ImageStrategy
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
ButtonToImageStrategyMapper
ButtonToImageStrategyMapper(int[],ImageStrategy[])
getImageStrategy(int):ImageStrategy
DownloadWithRunnables
DownloadWithRunnables()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
DownloadWithMessages
mThread: Thread
SHOW_TOAST: int
DISPLAY_IMAGE: int
DownloadWithMessages()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
DownloadWithAsyncTask
mDownloader: AsyncTask<String,Void,Bitmap>
DownloadWithAsyncTask()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void

See github.com/douglascraigschmidt/POSA-15/tree/master/ex/ImageDownloads

Overview of ImageDownloads

- This implementation is based on the *Model-View-Presenter* (MVP) pattern

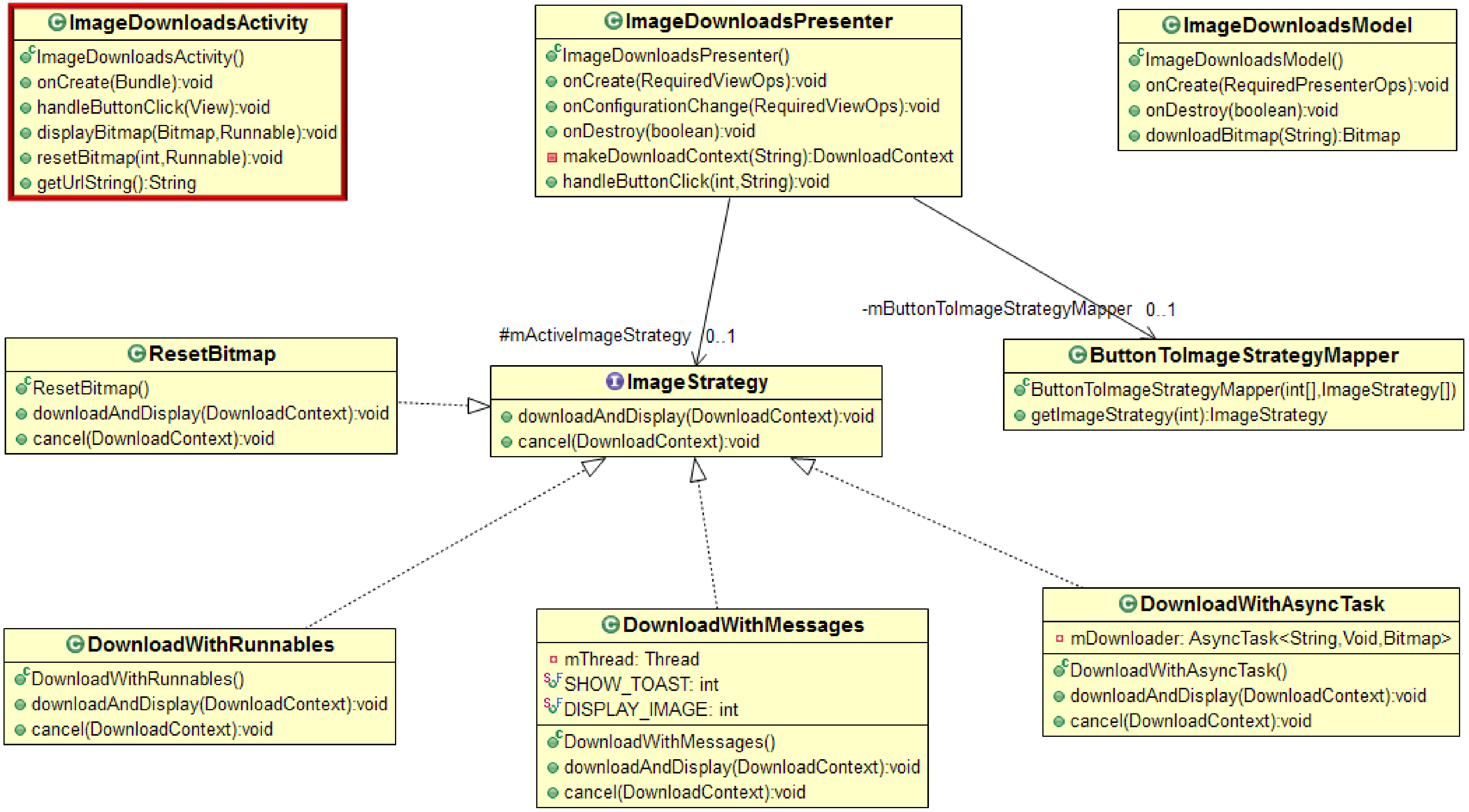


ImageDownloadsActivity
ImageDownloadsActivity()
onCreate(Bundle):void
handleButtonClick(View):void
displayBitmap(Bitmap,Runnable):void
resetBitmap(int,Runnable):void
getUrlString():String
ImageDownloadsPresenter
ImageDownloadsPresenter()
onCreate(RequiredViewOps):void
onConfigurationChange(RequiredViewOps):void
onDestroy(boolean):void
makeDownloadContext(String):DownloadContext
handleButtonClick(int,String):void
ImageDownloadsModel
ImageDownloadsModel()
onCreate(RequiredPresenterOps):void
onDestroy(boolean):void
downloadBitmap(String):Bitmap
#mActiveImageStrategy 0..1
-mButtonToImageStrategyMapper 0..1
ResetBitmap
ResetBitmap()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
ImageStrategy
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
ButtonToImageStrategyMapper
ButtonToImageStrategyMapper(int[],ImageStrategy[])
getImageStrategy(int):ImageStrategy
DownloadWithRunnables
DownloadWithRunnables()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
DownloadWithMessages
mThread: Thread
SHOW_TOAST: int
DISPLAY_IMAGE: int
DownloadWithMessages()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
DownloadWithAsyncTask
mDownloader: AsyncTask<String,Void,Bitmap>
DownloadWithAsyncTask()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void

This solution uses the MPV pattern to handle runtime configuration changes

Overview of ImageDownloads

- This implementation is based on the *Model-View-Presenter* (MVP) pattern

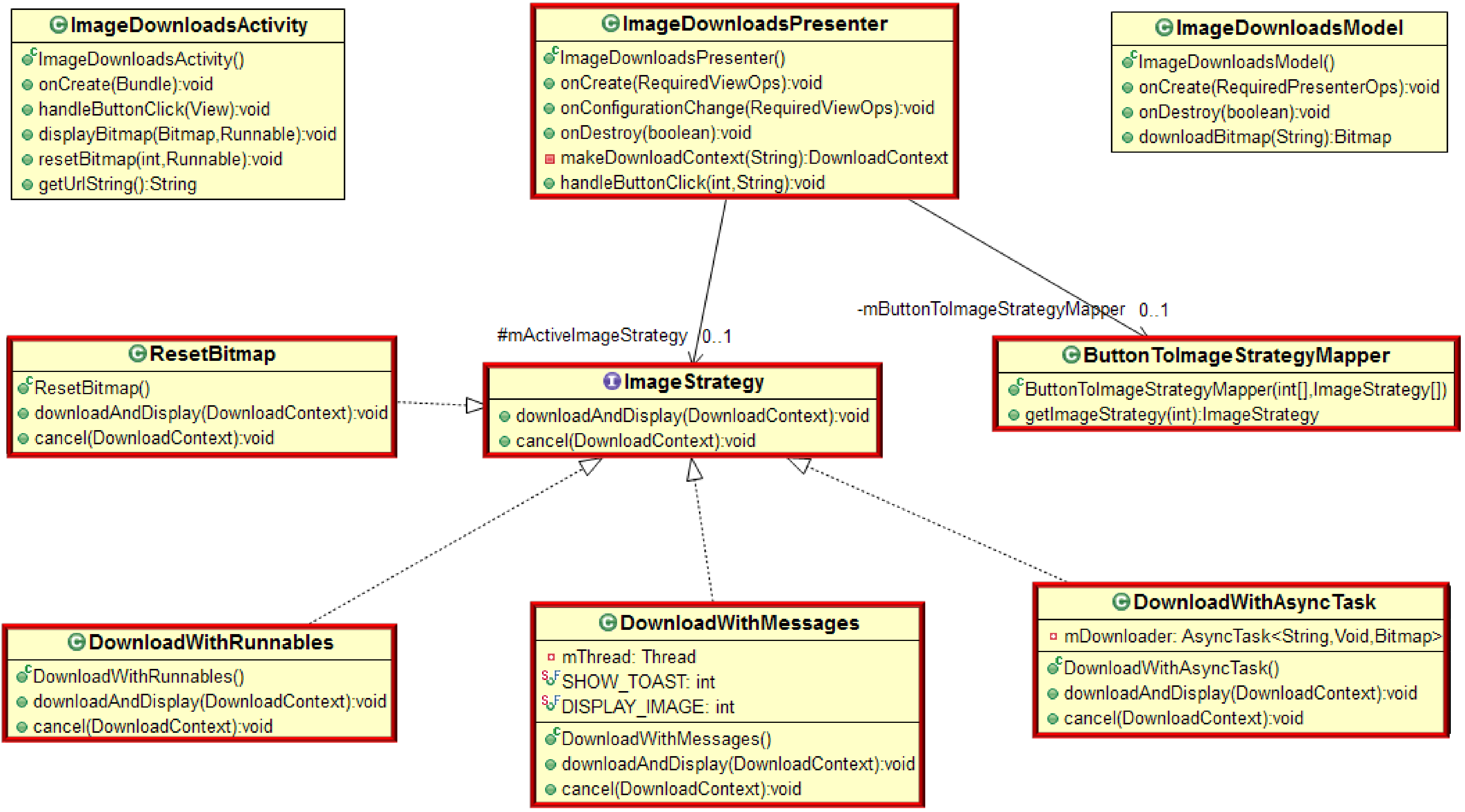


ImageDownloadsActivity
ImageDownloadsActivity()
onCreate(Bundle):void
handleButtonClick(View):void
displayBitmap(Bitmap,Runnable):void
resetBitmap(int,Runnable):void
getUrlString():String
ImageDownloadsPresenter
ImageDownloadsPresenter()
onCreate(RequiredViewOps):void
onConfigurationChange(RequiredViewOps):void
onDestroy(boolean):void
makeDownloadContext(String):DownloadContext
handleButtonClick(int,String):void
ImageDownloadsModel
ImageDownloadsModel()
onCreate(RequiredPresenterOps):void
onDestroy(boolean):void
downloadBitmap(String):Bitmap
-mButtonToImageStrategyMapper 0..1
#mActiveImageStrategy 0..1
ResetBitmap
ResetBitmap()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
ImageStrategy
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
ButtonToImageStrategyMapper
ButtonToImageStrategyMapper(int[],ImageStrategy[])
getImageStrategy(int):ImageStrategy
DownloadWithRunnables
DownloadWithRunnables()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
DownloadWithMessages
mThread: Thread
SHOW_TOAST: int
DISPLAY_IMAGE: int
DownloadWithMessages()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
DownloadWithAsyncTask
mDownloader: AsyncTask<String,Void,Bitmap>
DownloadWithAsyncTask()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void

The View layer handles interactions with the user

Overview of ImageDownloads

- This implementation is based on the *Model-View-Presenter* (MVP) pattern

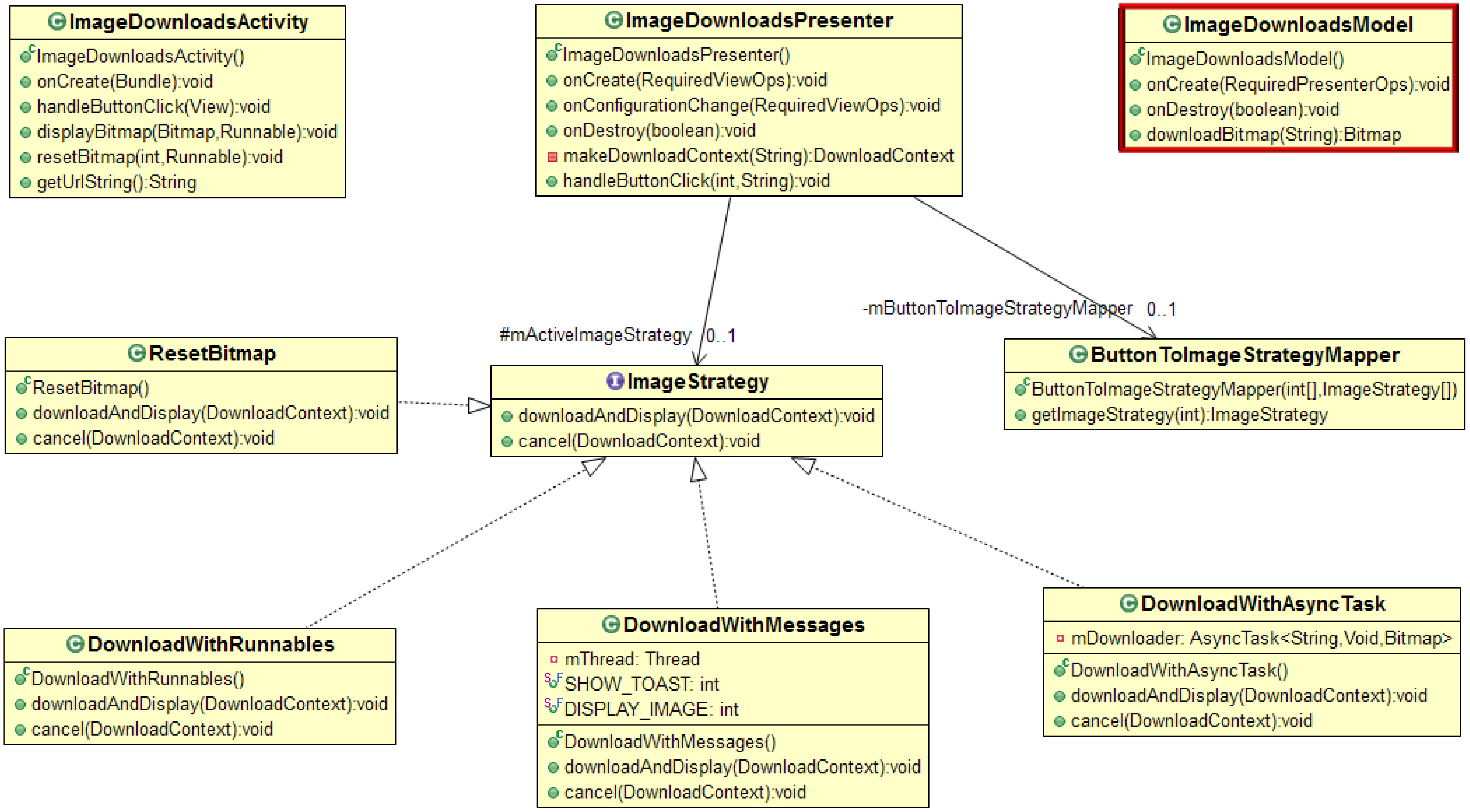


ImageDownloadsActivity
ImageDownloadsActivity()
onCreate(Bundle):void
handleButtonClick(View):void
displayBitmap(Bitmap,Runnable):void
resetBitmap(int,Runnable):void
getUrlString():String
ImageDownloadsPresenter
ImageDownloadsPresenter()
onCreate(RequiredViewOps):void
onConfigurationChange(RequiredViewOps):void
onDestroy(boolean):void
makeDownloadContext(String):DownloadContext
handleButtonClick(int,String):void
ImageDownloadsModel
ImageDownloadsModel()
onCreate(RequiredPresenterOps):void
onDestroy(boolean):void
downloadBitmap(String):Bitmap
-mButtonToImageStrategyMapper 0..1
#mActiveImageStrategy 0..1
ResetBitmap
ResetBitmap()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
ImageStrategy
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
ButtonToImageStrategyMapper
ButtonToImageStrategyMapper(int[],ImageStrategy[])
getImageStrategy(int):ImageStrategy
DownloadWithRunnables
DownloadWithRunnables()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
DownloadWithMessages
mThread: Thread
SHOW_TOAST: int
DISPLAY_IMAGE: int
DownloadWithMessages()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
DownloadWithAsyncTask
mDownloader: AsyncTask<String,Void,Bitmap>
DownloadWithAsyncTask()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void

The Presenter layer concurrently mediates interactions with the View & Model layers

Overview of ImageDownloads

- This implementation is based on the *Model-View-Presenter* (MVP) pattern


ImageDownloadsActivity
ImageDownloadsActivity()
onCreate(Bundle):void
handleButtonClick(View):void
displayBitmap(Bitmap,Runnable):void
resetBitmap(int,Runnable):void
getUrlString():String
ImageDownloadsPresenter
ImageDownloadsPresenter()
onCreate(RequiredViewOps):void
onConfigurationChange(RequiredViewOps):void
onDestroy(boolean):void
makeDownloadContext(String):DownloadContext
handleButtonClick(int,String):void
ImageDownloadsModel
ImageDownloadsModel()
onCreate(RequiredPresenterOps):void
onDestroy(boolean):void
downloadBitmap(String):Bitmap
-mButtonToImageStrategyMapper 0..1
#mActiveImageStrategy 0..1
ResetBitmap
ResetBitmap()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
ImageStrategy
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
ButtonToImageStrategyMapper
ButtonToImageStrategyMapper(int[],ImageStrategy[])
getImageStrategy(int):ImageStrategy
DownloadWithRunnables
DownloadWithRunnables()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
DownloadWithMessages
mThread: Thread
SHOW_TOAST: int
DISPLAY_IMAGE: int
DownloadWithMessages()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void
DownloadWithAsyncTask
mDownloader: AsyncTask<String,Void,Bitmap>
DownloadWithAsyncTask()
downloadAndDisplay(DownloadContext):void
cancel(DownloadContext):void

The Model layer interacts with the remote web server