Concurrency

Mobile Application Development

Using Handler Object

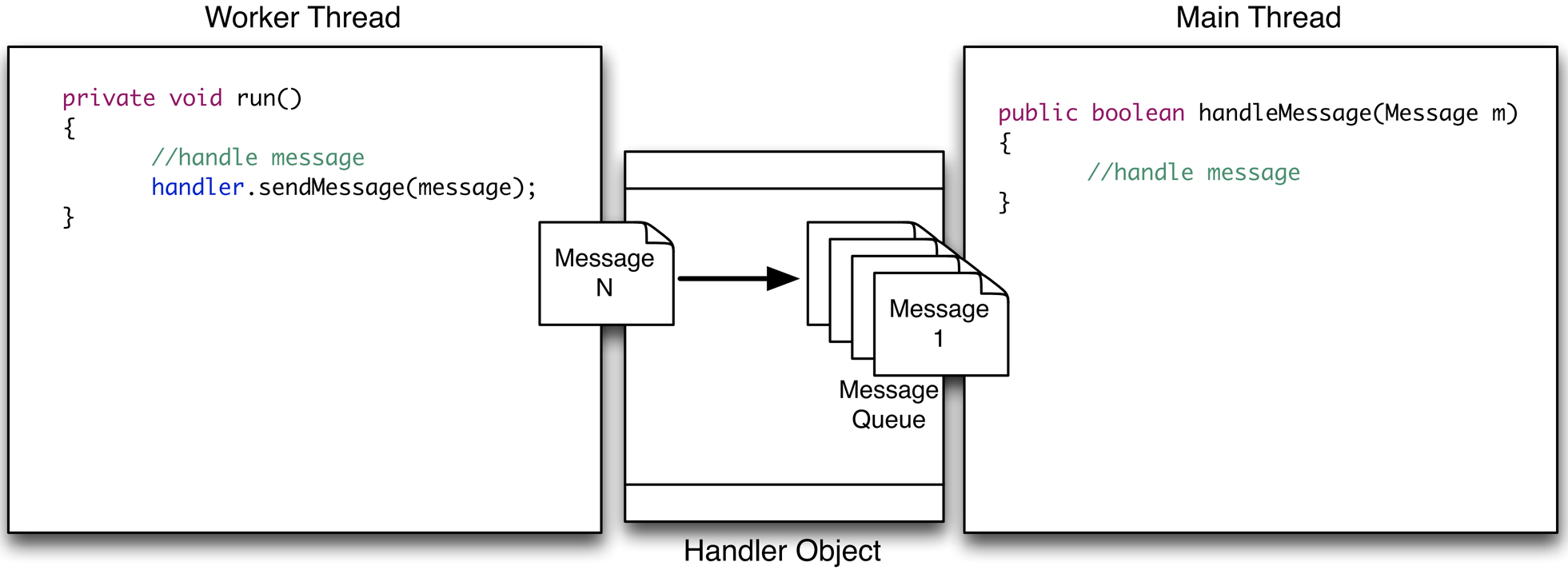
- **A Handler allows you to send and process Message and Runnable objects associated with a thread's MessageQueue.**

- **Each Handler instance is associated with a single thread and that thread's message queue.**
  - When you create a new Handler, it is bound to the thread / message queue of the thread that is creating it.

- **The handler will deliver messages and runnables to the message queue and execute them as they come out of the message queue.**

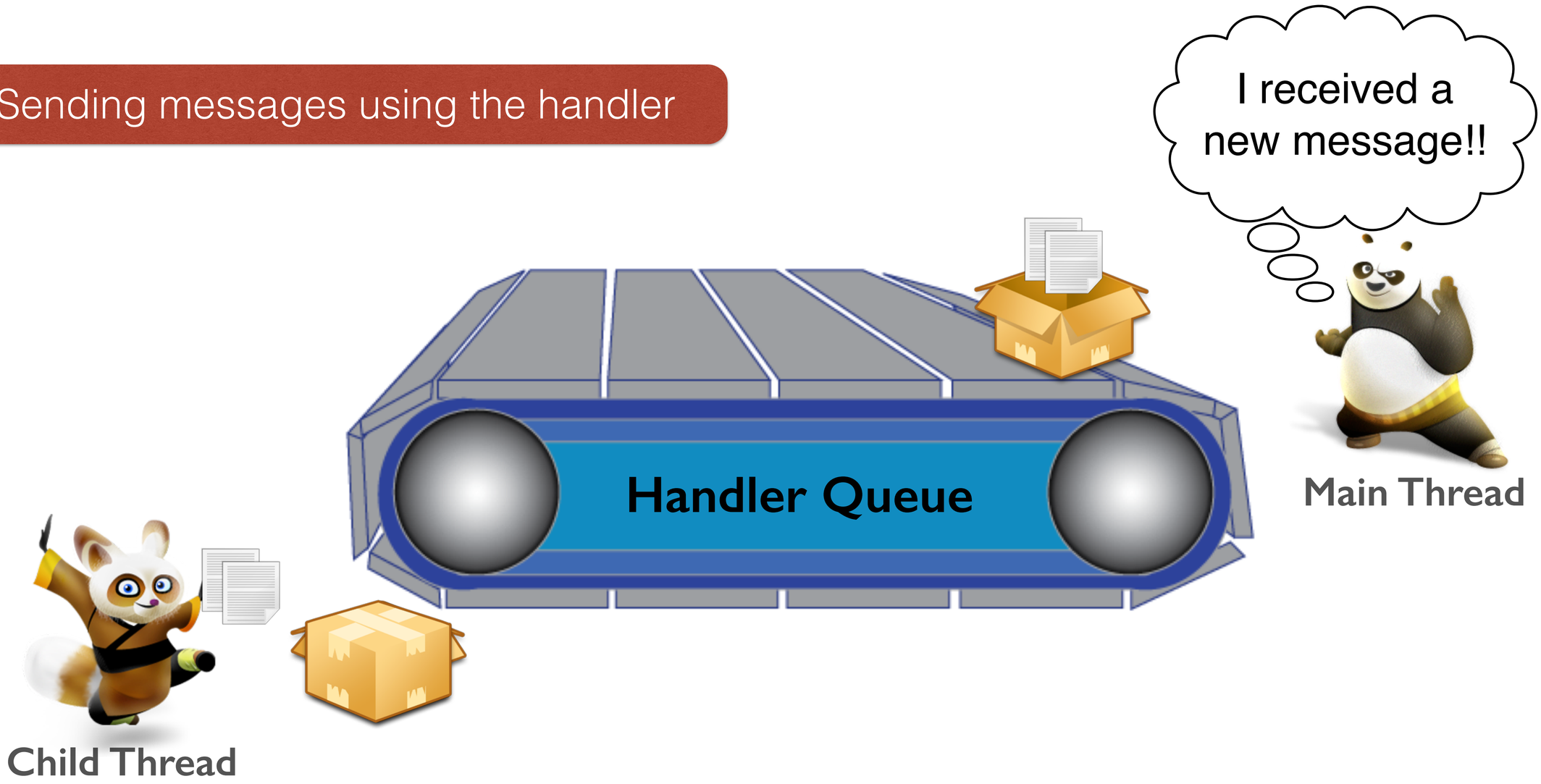
Thread Communication

- **Using the Handler Object a worker thread can send messages to the Main thread.**
  - For example, messages can be used to share progress information between the worker thread and the main thread.



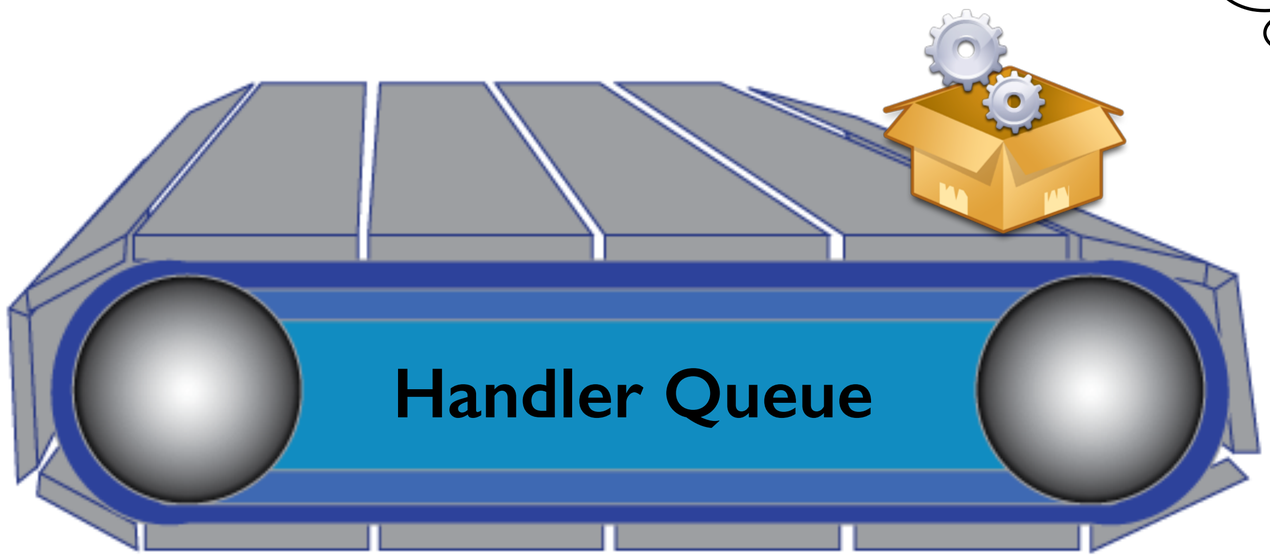
Handler Based Communication

Sending messages using the handler



Handler Based Communication

Sending Runnable using the handler

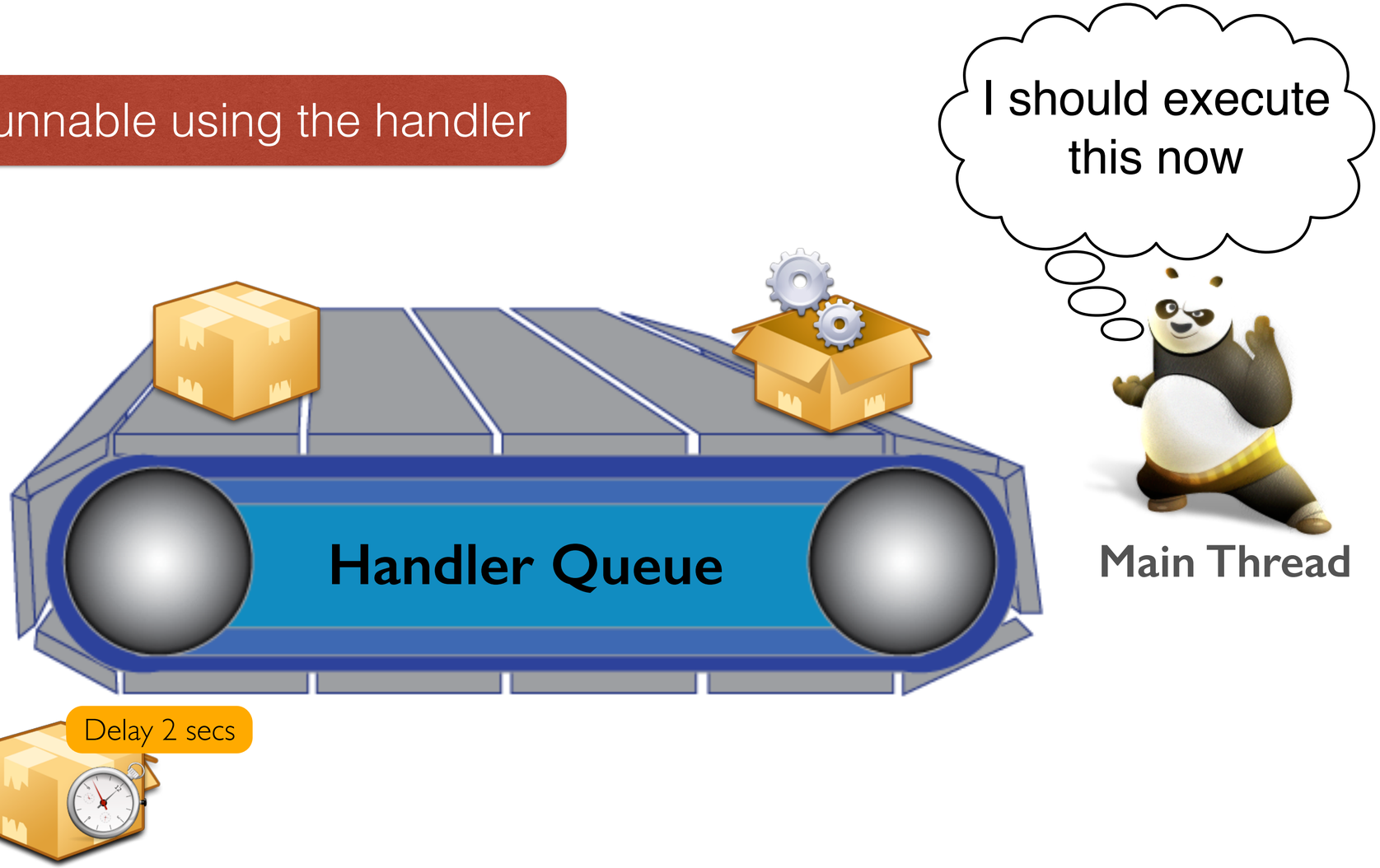


Main Thread

Child Thread

Handler Based Communication

Sending Delayed Runnable using the handler



Handler Object (Messages)

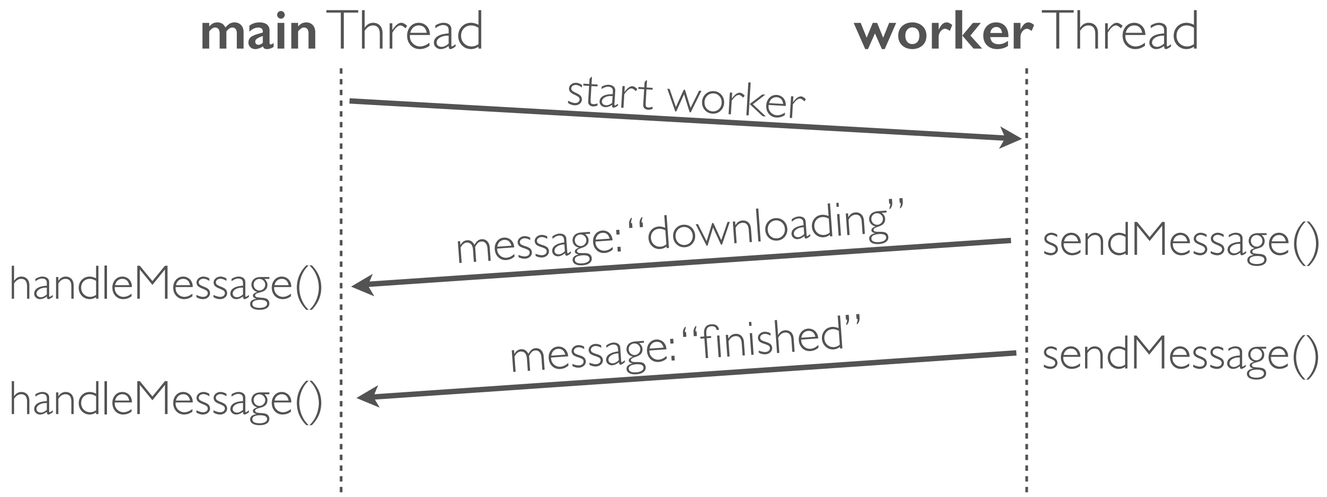
- **The worker thread can use the handler object to send messages to the UI thread by calling the handler's method `sendMessage(Message m)`.**
- **No synchronization is needed for the message queue as it is handled by the `Handler` object.**
- **The `Message` object contains a `Bundle` object that holds the data that will be processed by the Handler's callback function.**
- **The receiving thread should implement the callback method `handleMessage(Message)` defined by the interface "Handler.Callback".**

Steps using the Handler Object

- **A simple approach to using a worker thread and handlers:**
  - Step 1: Create a handler in the main thread.
  - Step 2: Create a worker thread that is going to do the actual work, make sure to pass the handler to the worker thread.
  - Step 3: The worker thread can now do work that can take longer than 5 seconds and will not hold the main thread.
  - Step 4: The worker thread will use the handler to send status messages and results to the handler.
  - Step 5: The received messages will be processed by the main thread which owns the handler.

Handler (Messages)

- **In this example we have a main thread and a worker thread.**
  - The worker thread downloads an image from a remote website.
  - The worker will report the status of the image download.



Handler (Messages)

```
public class HandlerExamplesActivity extends Activity {
   private Handler handler;
   private Runnable imageDownload = new Runnable() {
      private void sendMsg(String msgText){
         Bundle bundle = new Bundle();
         bundle.putString("status", msgText);
         Message message = new Message();
         message.setData(bundle);
         handler.sendMessage(message);
      }
      public void run() {
         sendMsg("Starting Thread");
         try {
            URL url = new URL("http://www.uncc.edu/sites/default/files/spotlight/give-blood-no-text.jpg");
            Bitmap image = BitmapFactory.decodeStream(url.openStream());
            if(image != null){
               sendMsg("File Retrieved");
            } else{
               sendMsg("File Error");
            }
         } catch (Exception e) {
            sendMsg("Failed Downloading");
            e.printStackTrace();
         }
      }
   };
```

The Bundle is similar to the Java Map, stores a pair (key/value)

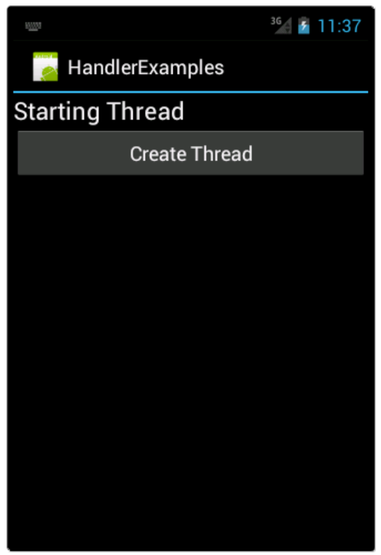
handler used to send the message to the message queue

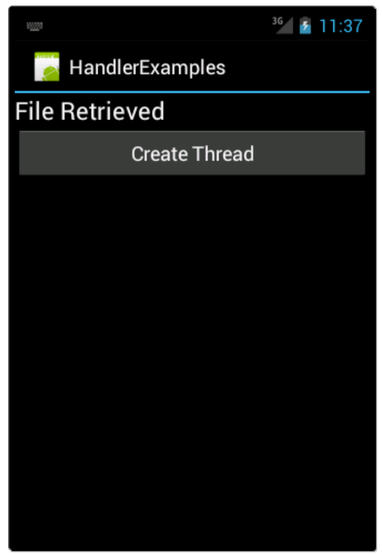
Download an image from the web.

Handler (Messages)

```
  public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
     handler = new Handler(new Handler.Callback() {
        public boolean handleMessage(Message msg) {
           if(msg.getData().containsKey("status")){
              String text = msg.getData().getString("status");
              ((TextView) findViewById(R.id.textView1)).setText(text);
           }
           return true;
        }
     });
     Button b = (Button) findViewById(R.id.button1);
     b.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
           new Thread(imageDownload, "Download Thread").start();
        }
     });
  }
}
```

Handler Callback function to receive the messages





Handler (Messages)

- **The message passing mechanism is not limited to exchanging Strings, more complex types can be passed.**

- **One example is the `Parcelable` Interface and the `Bitmap` is parcelable. So the `Bitmap` created can easily be passed via the message passing `Handler`.**

- **The worker thread can use the handler object to send a `Runnable` to the UI thread. The runnable will be run on the thread to which this handler is attached (UI thread).**
  - This can be done by using `post(Runnable r)` method.
  - The post versions allow you to enqueue `Runnable` objects to be called by the message queue when they are received.

Handler

- **Scheduling messages is accomplished by using the methods:**
  - `sendEmptyMessage(int)`
  - `sendMessage(Message)`
  - `sendMessageAtTime(Message, long)`
  - `sendMessageDelayed(Message, long)`

- **To send Runnable:**
  - `post(Runnable)`
  - `postAtTime(Runnable, long)`
  - `postDelayed(Runnable, long)`

Handler (postDelayed)

```
public class ServiceDemoActivity extends Activity {
   private Handler h;
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      h = new Handler();
      Button b = (Button) findViewById(R.id.button3);
      b.setOnClickListener(new View.OnClickListener() {
         public void onClick(View v) {
            h.postDelayed(new Runnable() {
               @Override
               public void run() {
                  showDialog("Done");
               }
            }, 5000);
         }
      });
   }

   public void showDialog(String msg){
      new AlertDialog.Builder(this).setMessage(msg).setPositiveButton("Ok", new DialogInterface.OnClickListener() {
         public void onClick(DialogInterface dialog, int which) {}
      }).create().show();
   }
}
```

Using Handler to submit a runnable to be executed after a 5000ms delay.