Small talk with the UI thread

Landon Cox

March 9, 2017

Android threads

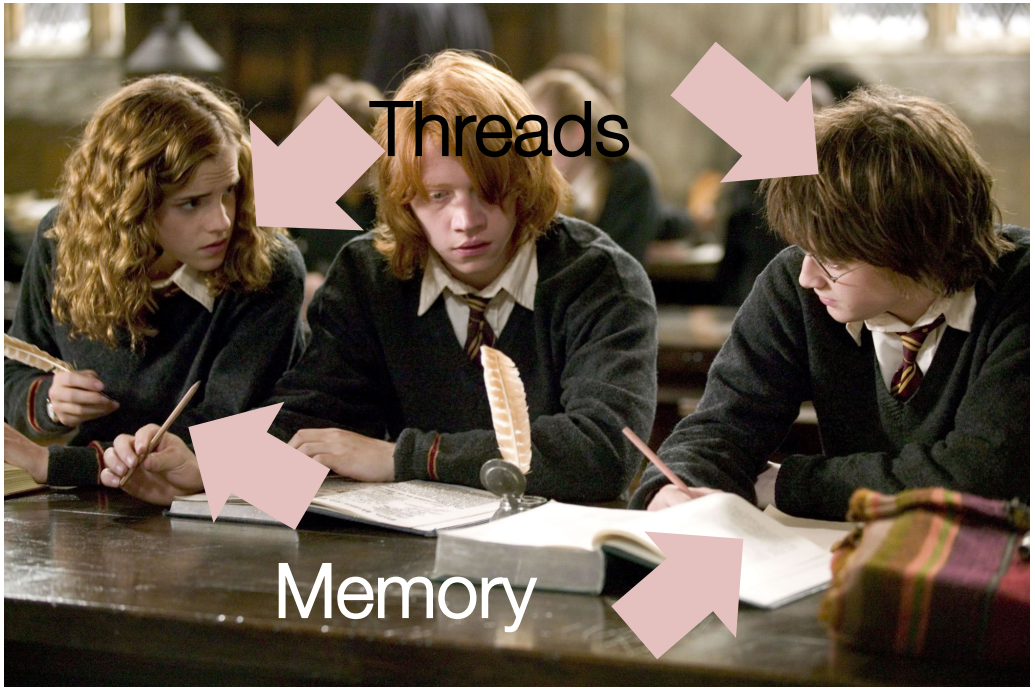
- Thread: “a sequence of executing instructions”
- Every app has a main thread
  - Processes UI events (taps, swipes, etc.)
  - Updates the UI
- Apps will probably want other threads too
  - Use threads to communicate over network
  - Read/write to file system or database
  - (anything slow)

Play analogy

- Process is like a play performance
- Program is like the play's script

What are the threads?

What is the memory?



Multiple threads in memory

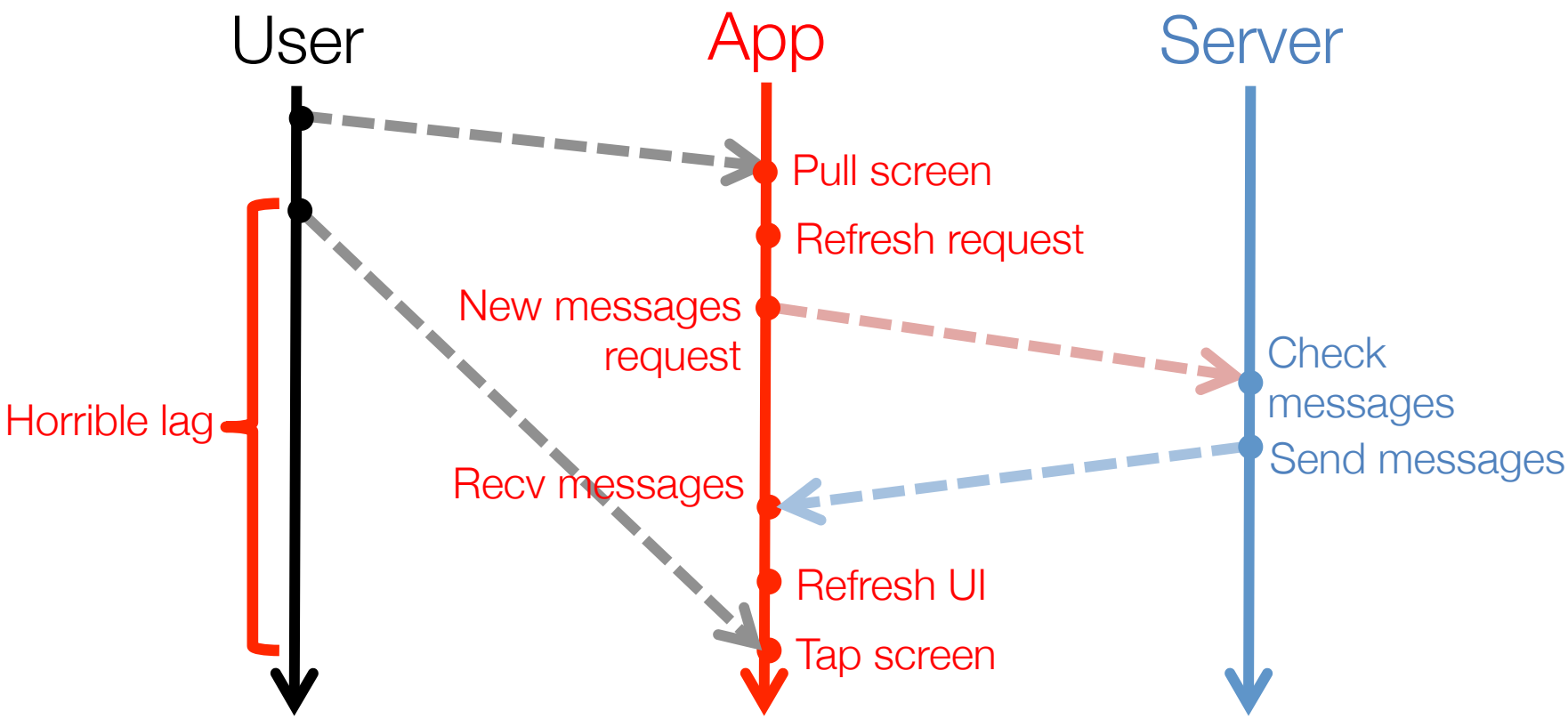
- Several actors on a single set
  - Sometimes they interact (speak, dance)
  - Sometimes they are apart (different scenes)

Private vs global thread state

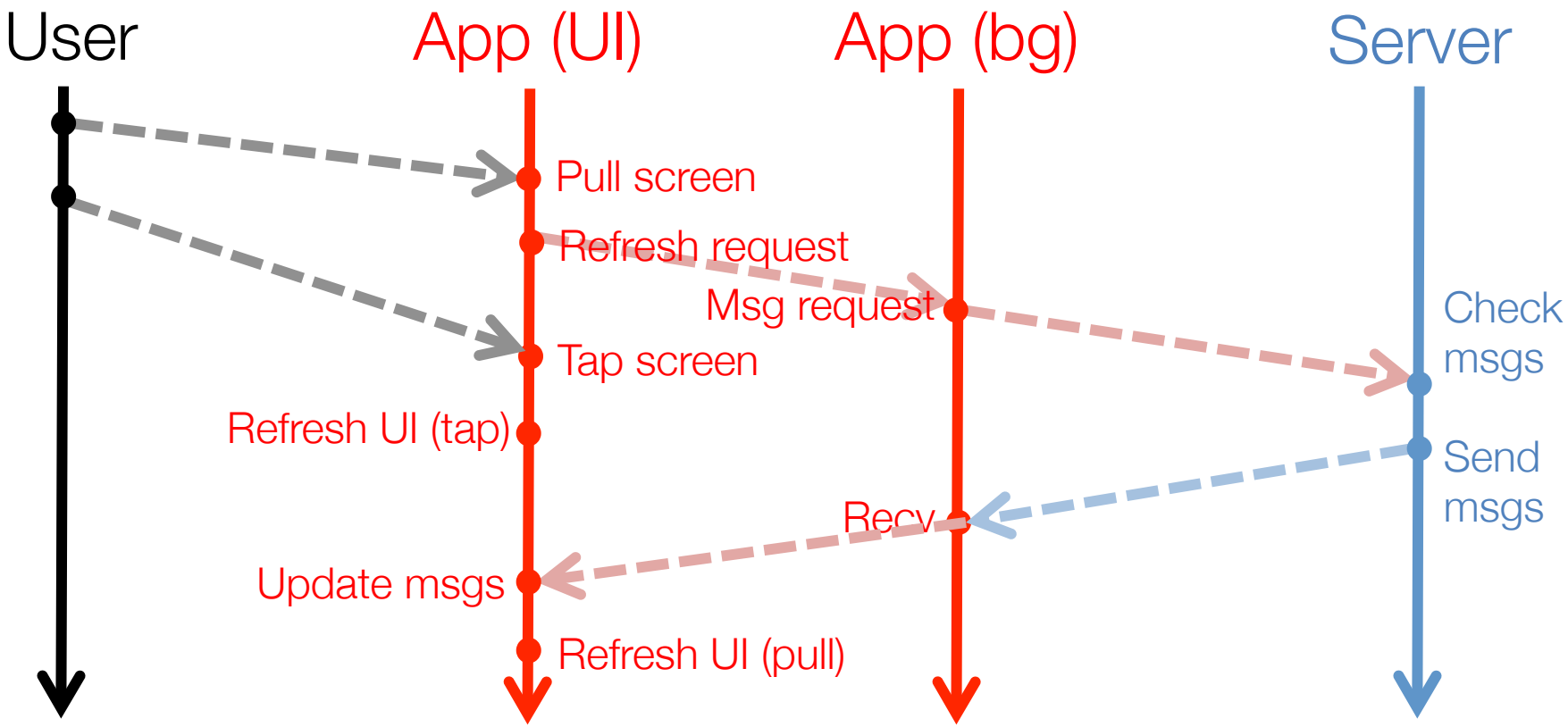
- What is private to each thread?
  - Current instruction (where actor is in his/her script)
  - Local variables, stack (actor's mindset)

- What is shared?
  - Global variables, heap
  - (props on set)
  - Code (like the script)

Single-threaded app

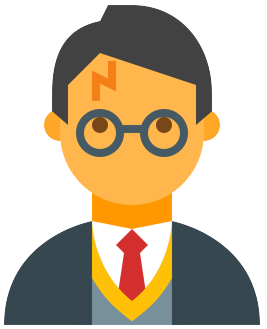


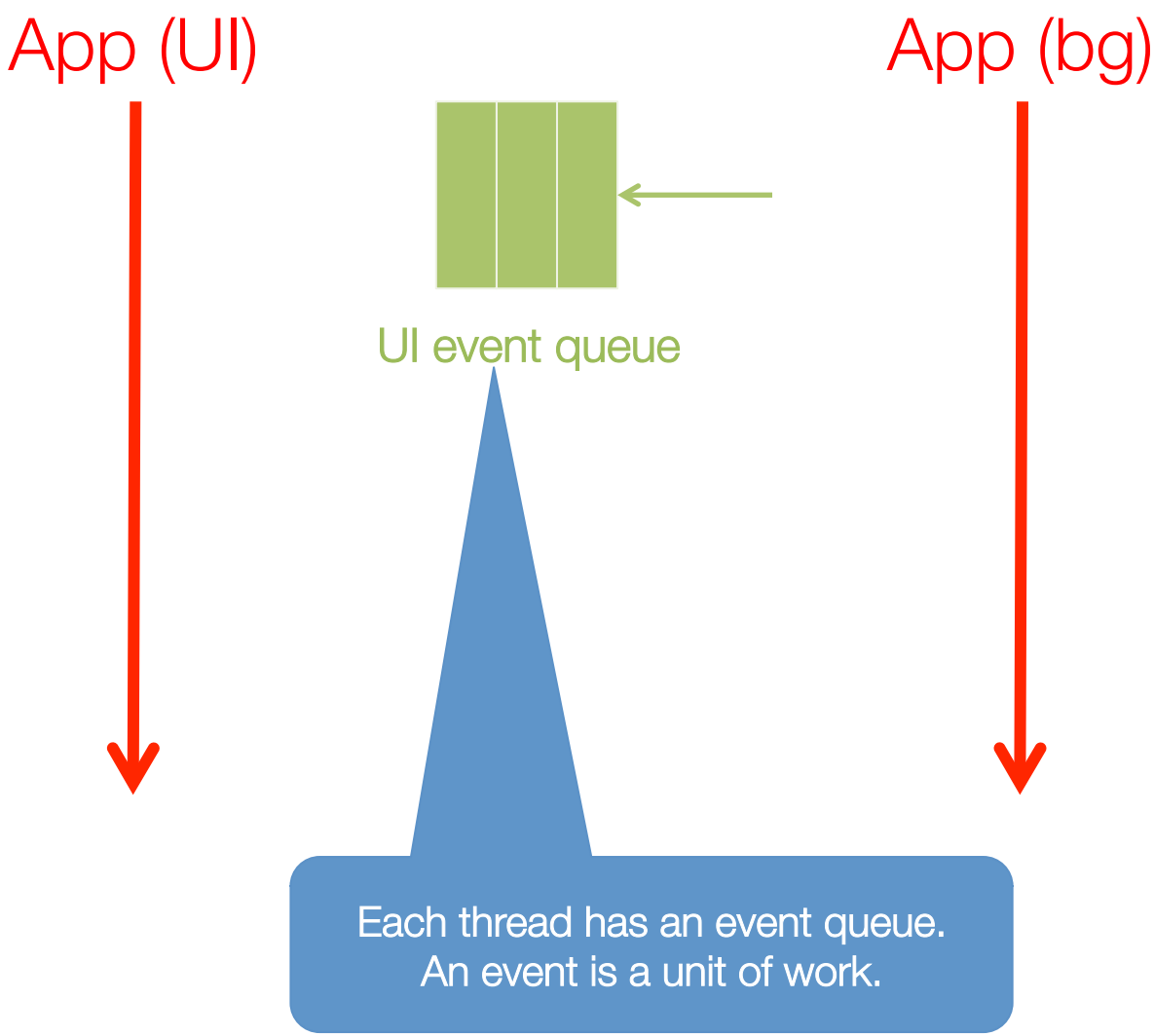
Multi-threaded app

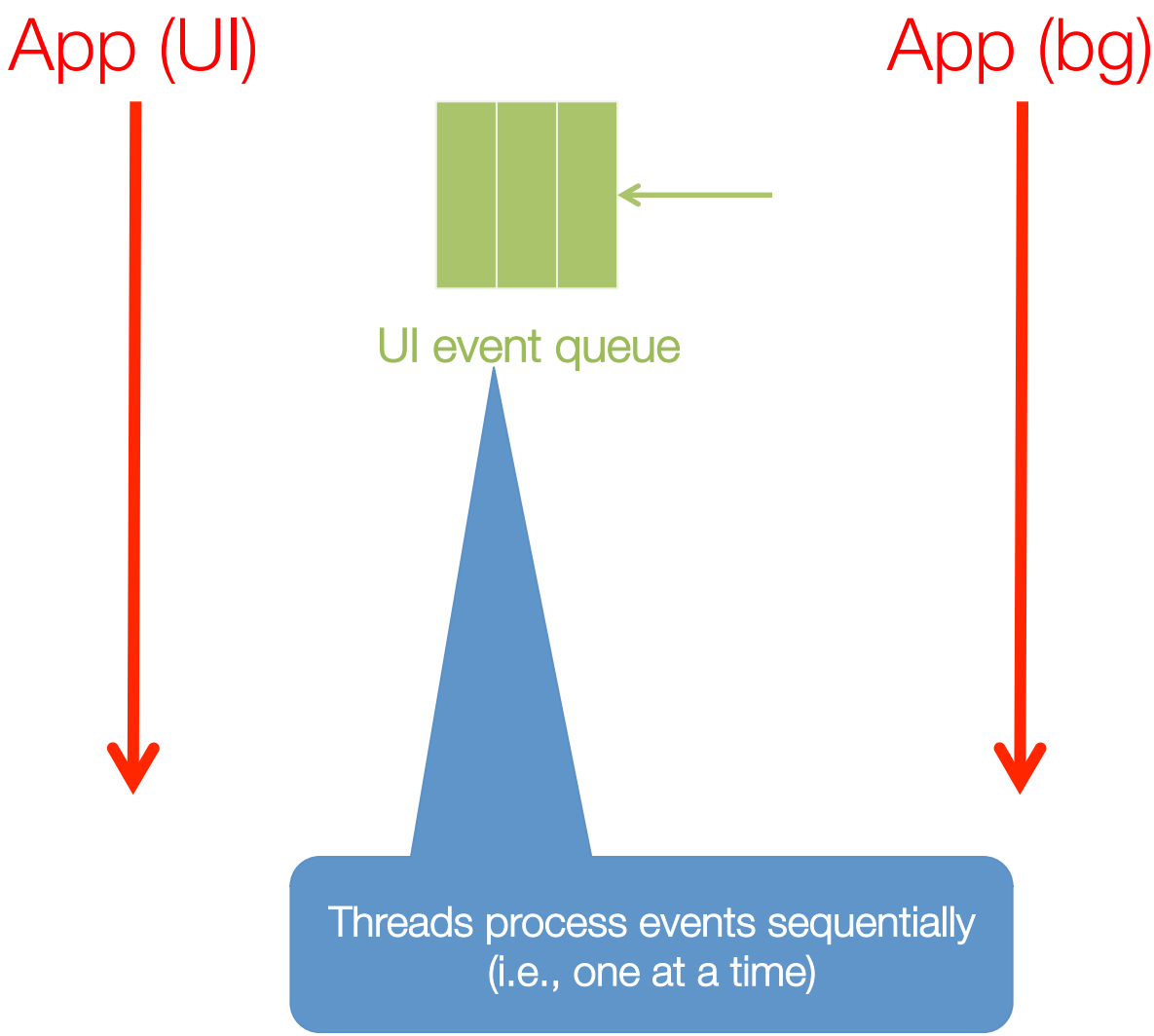


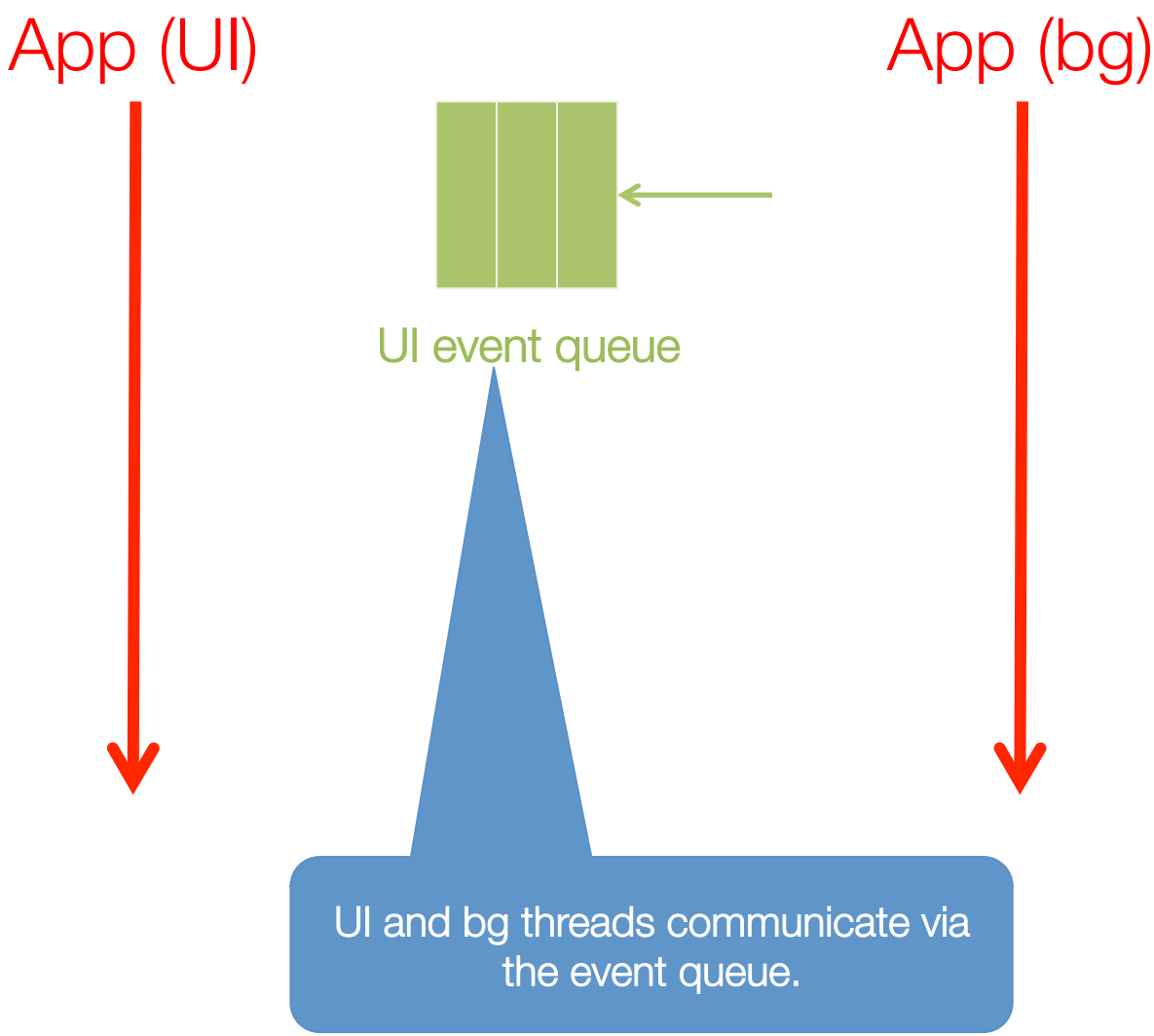
Threads in Android

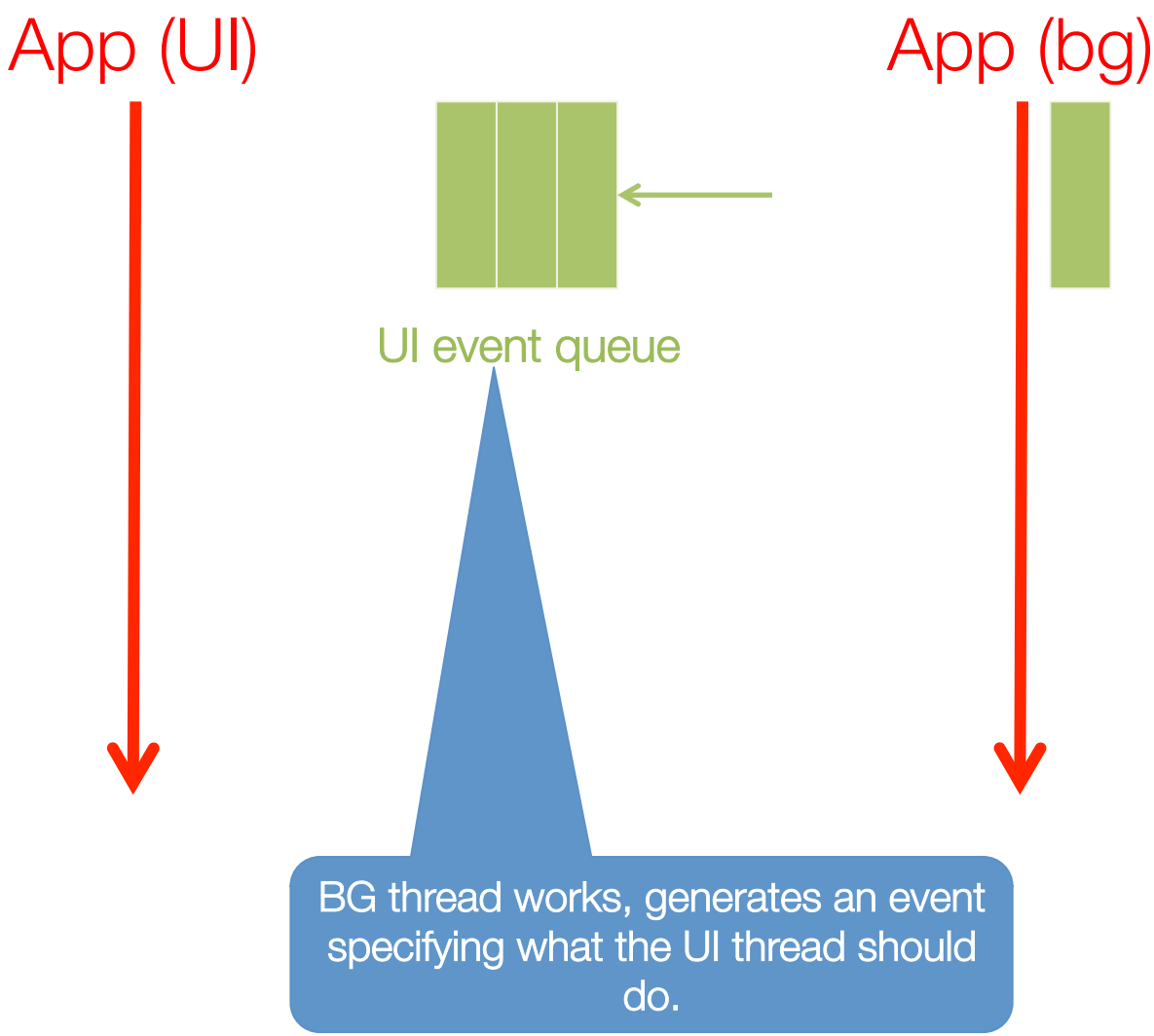
- Main/UI thread
  - Networking code cannot run on it
  - File/database code probably shouldn't run on it
  - Only thread on which can View code can run
  - Play analogy? something Harry cannot do …
- Background threads
  - Place to run slow code (e.g., networking/db)
  - Cannot run View code
  - Play analogy? something only Harry can do …
- How do we update the UI w/ background data?

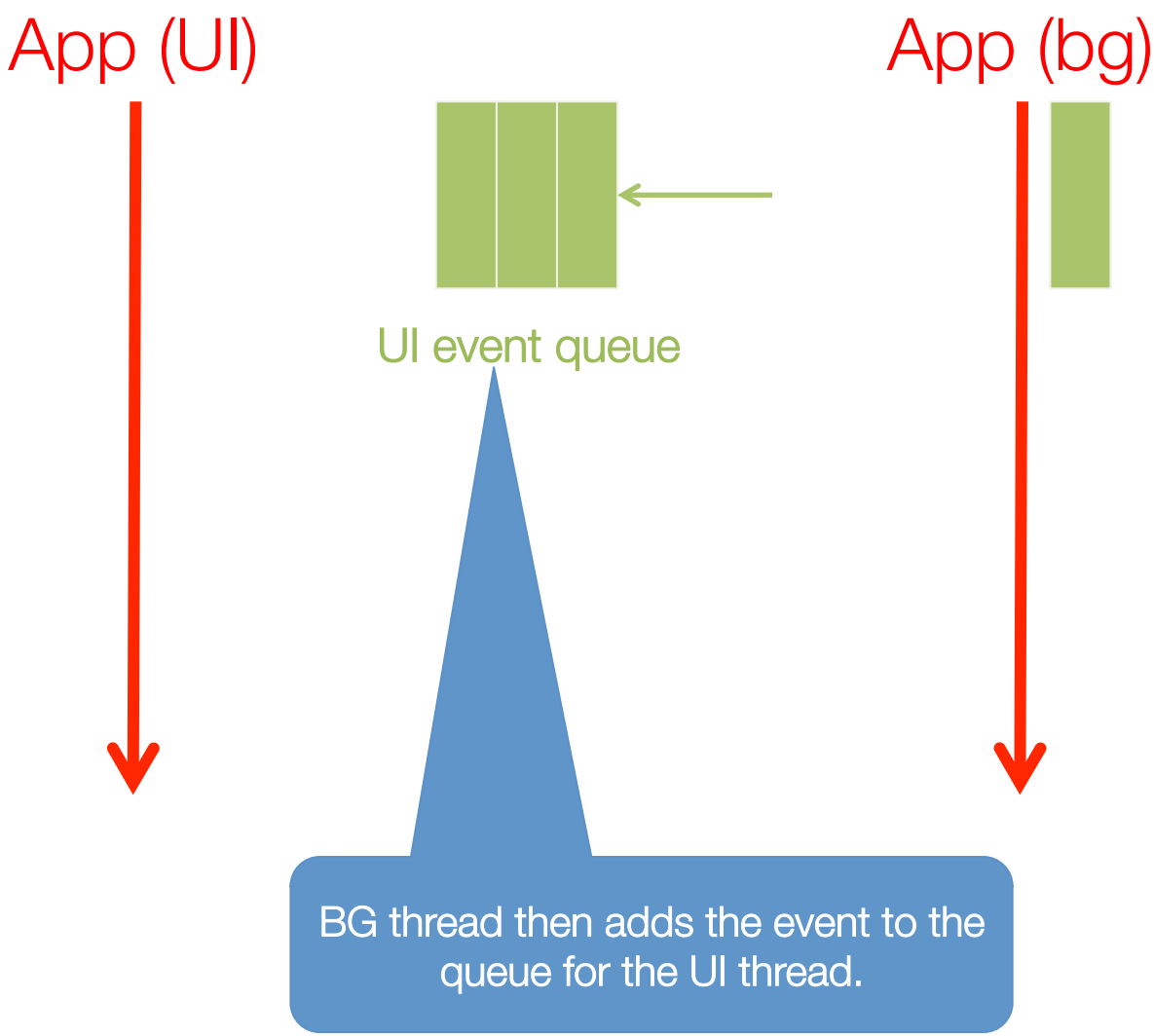


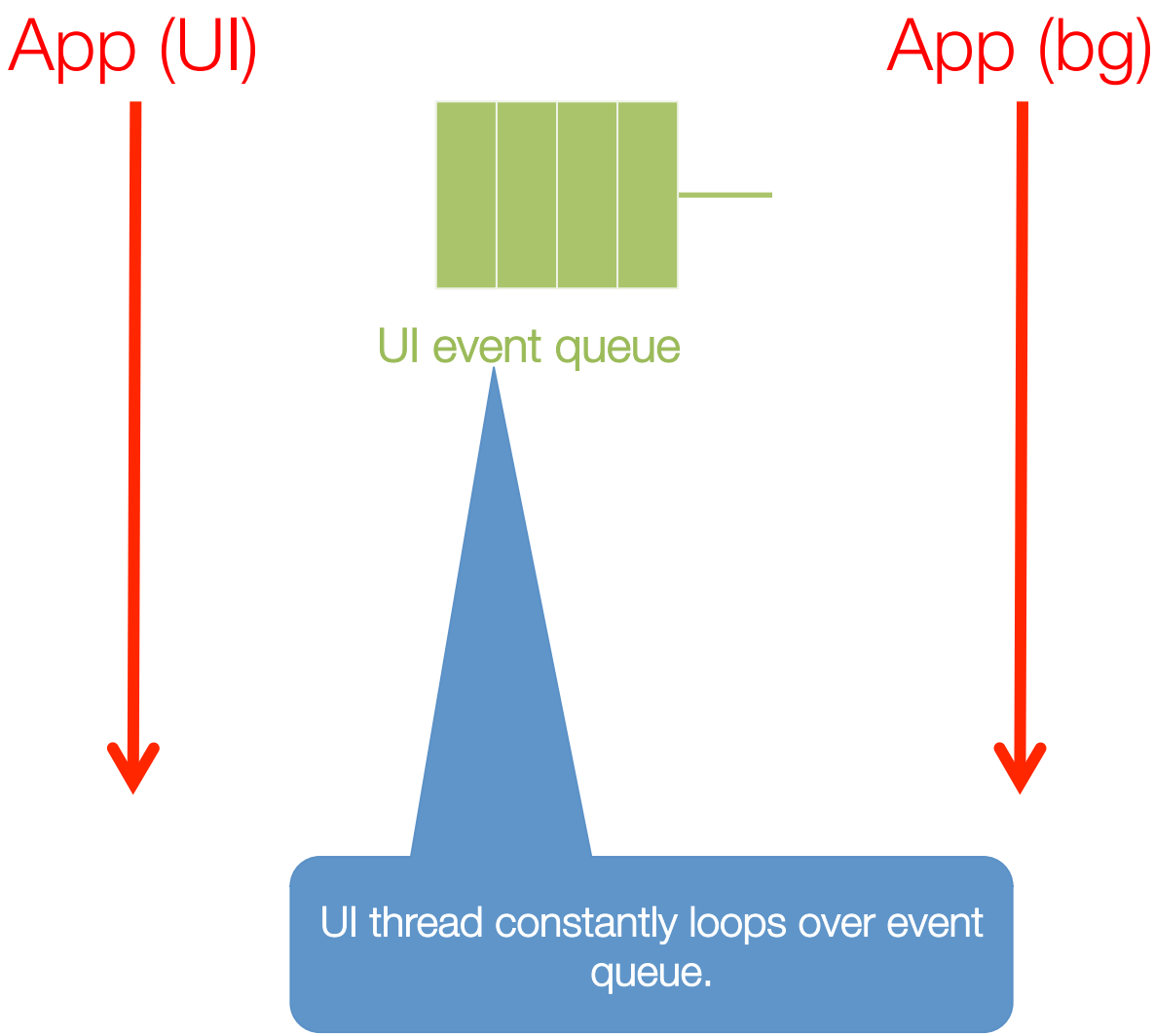

App (UI)
App (bg)
UI event queue
Each thread has an event queue.
An event is a unit of work.

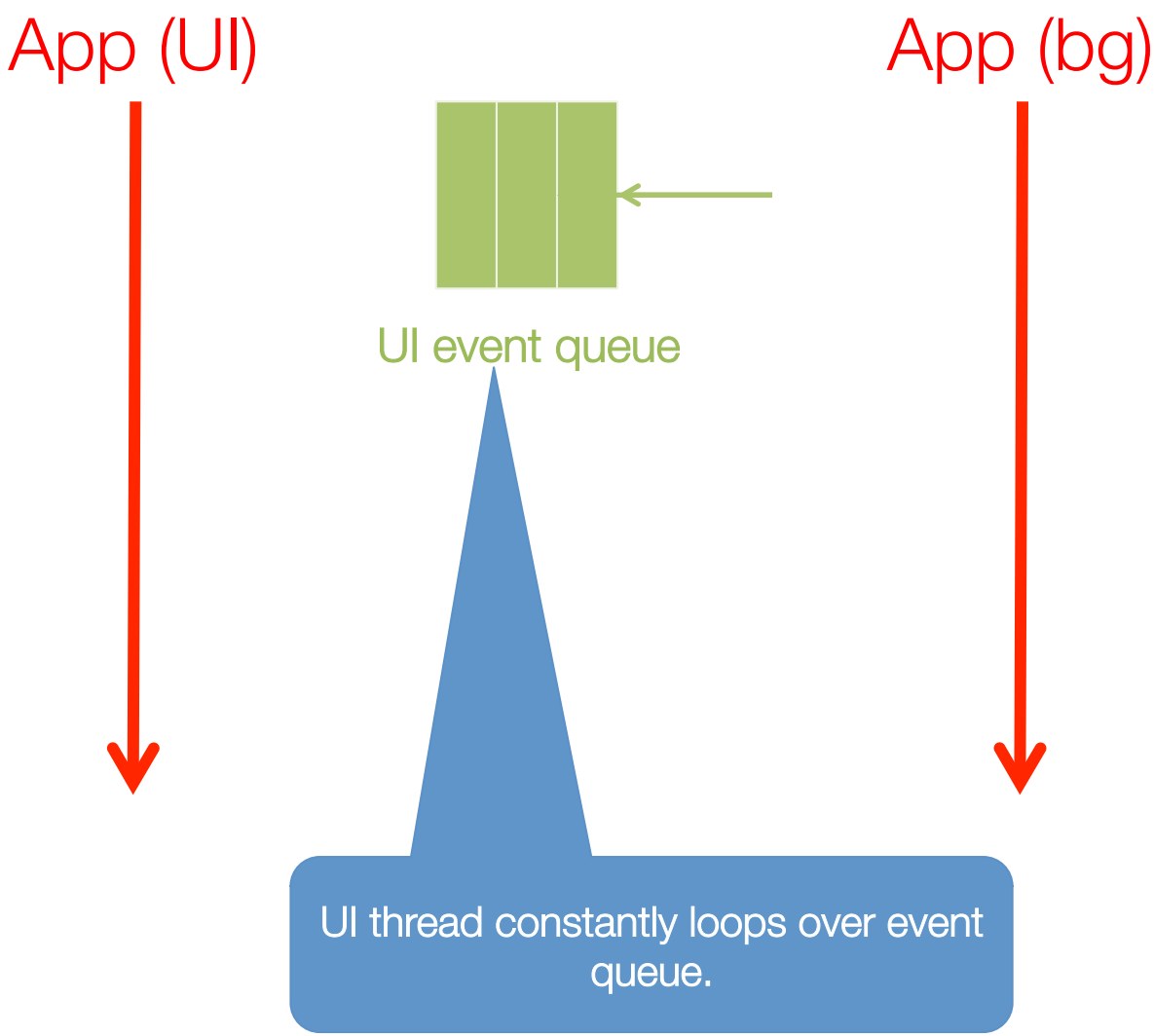

App (UI)
App (bg)
UI event queue
Threads process events sequentially
(i.e., one at a time)


App (UI)
App (bg)
UI event queue
UI and bg threads communicate via the event queue.


App (UI)
App (bg)
UI event queue
BG thread works, generates an event specifying what the UI thread should do.


App (UI)
App (bg)
UI event queue
BG thread then adds the event to the queue for the UI thread.


App (UI)
App (bg)
UI event queue
UI thread constantly loops over event queue.


App (UI)
App (bg)
UI event queue
UI thread constantly loops over event queue.

Updating the UI

- UI, non-UI threads talk via Handler/Looper

```
class LooperThread extends Thread {

  public Handler mHandler;

  public void run() {
    Looper.prepare();

    mHandler = new Handler() {
      public void handleMessage(Message msg) {
        // process incoming messages here
      }
    };

    Looper.loop();
  }
}
```

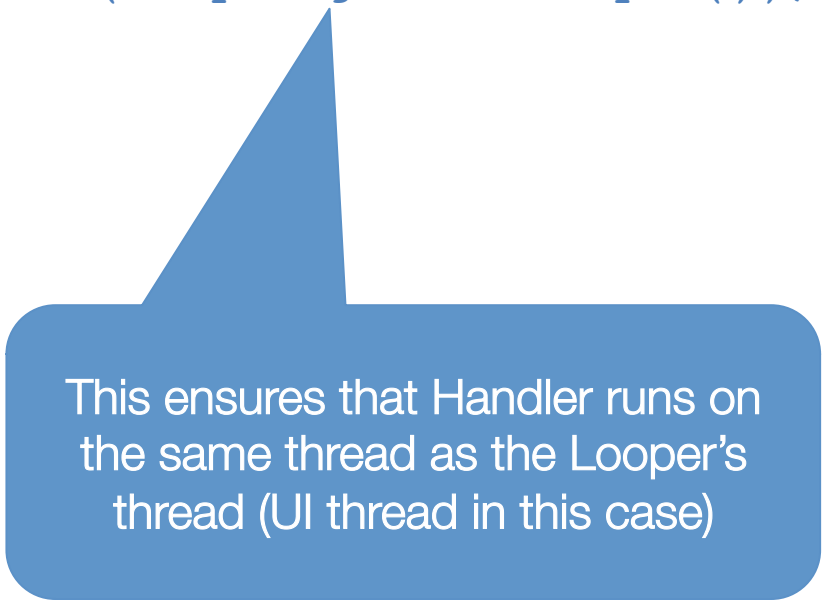
UI thread already setup with a Looper

Use a Handler to pass data to the UI thread

Updating the UI

- To talk to the UI thread, create a Handler

```
handler = new Handler(Looper.getMainLooper());
```



Updating the UI

- To talk to the UI thread, create a Handler

```
handler = new Handler(Looper.getMainLooper());
```

Updating the UI

- To talk to the UI thread, create a Handler

```
handler = new Handler(Looper.getMainLooper()) {
  public void handleMessage(Message msg){
    // do something with the msg
  }
};
```

On what thread will handleMessage run?

On the UI thread

Updating the UI

- To talk to the UI thread, create a Handler

```
handler = new Handler(Looper.getMainLooper()) {
  public void handleMessage(Message msg){
    // do something with the msg
  }
};
```

What data does the method need to update the UI?

Reference to UI element to update, update content (e.g., image data)

Updating the UI

- To talk to the UI thread, create a Handler

```
handler = new Handler(Looper.getMainLooper()) {
  public void handleMessage(Message msg){
    // do something with the msg
  }
};
```

Where will the method get the data it needs?

From the passed-in Message!

Updating the UI

- Let's take a look at Message

```
class Message {
  public int arg1;
  public int arg2;
  public Object obj;
  public Messenger replyTo;
  public int what;

  void sendToTarget();
  static Message obtain (Handler h, …);
}
```

Use this to create create a new Message

Updating the UI

- Let's take a look at Message

```
class Message {
  public int arg1;
  public int arg2;
  public Object obj;
  public Messenger replyTo;
  public int what;

  void sendToTarget();
  static Message obtain (Handler h, …);
}
```

Why this instead of calling the constructor?

Makes it easier to re-use Messages

Updating the UI

- Let's take a look at Message

```
class Message {
  public int arg1;
  public int arg2;
  public Object obj;
  public Messenger replyTo;
  public int what;

  void sendToTarget();
  static Message obtain (Handler h, …);
}
```

Use this to send the Message to another thread

Updating the UI

- Let's take a look at Message

```
class Message {
  public int arg1;
  public int arg2;
  public Object obj;
  public Messenger replyTo;
  public int what;

  void sendToTarget();
  static Message obtain (Handler h, …);
}
```

How does the Message know where to send itself?

Handler passed in to obtain()

Updating the UI

- Let's take a look at Message

```
class Message {
  public int arg1;
  public int arg2;
  public Object obj;
  public Messenger replyTo;
  public int what;

  void sendToTarget();
  static Message obtain (Handler h, …);
}
```

What are these fields for?

This is how a Message's data is stored/retrieved

Updating the UI

- To talk to the UI thread, create a Handler

```
handler = new Handler(Looper.getMainLooper()) {
  public void handleMessage(Message msg){
    MyMsg mymsg = (MyMsg) msg.obj;
    View v = mymsg.getView (); // UI View to
update
    String s = mymsg.getString (); // String data
  }
};
```

MyMsg is a class that we defined

We can define any functions we want, like getView and getString

Putting it all together

```
mView = activity.findViewById(R.id.list);
fillList (mList);

handler = new Handler(Looper.getMainLooper()) {
  public void handleMessage(Message msg){
    MyMsg mymsg = (MyMsg) msg.obj;
    TextView v = mymsg.getView ();
    List list = mymsg.getList (); // List of Strings
     for (String s : list) {
       v.append(s);
    }
  }
};

MyMsg mymsg = new MyMsg(mList, mView);
Message m = Message.obtain(handler, mymsg);
m.sendToTarget();
```

AsyncTasks

- Android will do all of this for you via …

- AsyncTask
  - Creates a background thread
  - Syncs it with the main thread
- To use AsyncTask, subclass it

```
// subclass an AsyncTask
class MyTask extends AsyncTask<URL, Integer, Long> {
 …
};

MyTask t = new MyTask;
t.execute (new String[] {“www.cs.duke.edu”});
```

AsyncTask

- AsyncTask: simplest way to do background work

```
// subclass an AsyncTask
class MyTask extends AsyncTask<URL, Integer, Long> {
  Long doInBackground(URL … urls);
  void publishProgress(Integer … values);
  void onProgressUpdate(Integer … progress);
  void onPostExecute(Long result);
}

// to use an AsyncTask
new MyTask.execute(url1, url2, url3);
```

AsyncTask

- AsyncTask: simplest way to do background work

```
// subclass an AsyncTask
class MyTask extends AsyncTask<URL, Integer, Long> {
  Long doInBackground(URL … urls);
  void publishProgress(Integer … values);
  void onProgressUpdate(Integer … progress);
  void onPostExecute(Long result);
}

// to use an AsyncTask
new MyTask.execute(url1, url2, url3);
```

AsyncTask

- AsyncTask: simplest way to do background work

```
// subclass an AsyncTask
class MyTask extends AsyncTask<URL, Integer, Long> {
  Long doInBackground(URL … urls);
  void publishProgress(Integer … values);
  void onProgressUpdate(Integer … progress);
  void onPostExecute(Long result);
}

// to use an AsyncTask
new MyTask.execute(url1, url2, url3);
```

AsyncTask

- AsyncTask: simplest way to do background work

```
// subclass an AsyncTask
class MyTask extends AsyncTask<URL, Integer, Long> {
  Long doInBackground(URL … urls);
  void publishProgress(Integer … values);
  void onProgressUpdate(Integer … progress);
  void onPostExecute(Long result);
}

// to use an AsyncTask
new MyTask.execute(url1, url2, url3);
```

AsyncTask

- AsyncTask: simplest way to do background work

```
// subclass an AsyncTask
class MyTask extends AsyncTask<URL, Integer, Long> {
  Long doInBackground(URL … urls);
  void publishProgress(Integer … values);
  void onProgressUpdate(Integer … progress);
  void onPostExecute(Long re
}

// to use an AsyncTask
new MyTask.execute(url1, url2, url3);
```

This is where most of the work is done

AsyncTask

- AsyncTask: simplest way to do background work

```
class MyTask extends AsyncTask<URL, Integer, Long> {
  Long doInBackground(URL … urls) {
    int count=0;
    for (URL url: urls) {
      Downloader.downloadFile(url);
      publishProgress((int)((
                        count++*(float)/urls.length)*100);
    }
  }
  void publishProgress(Integer … values);
  void onProgressUpdate(Integer … progress) {
    // display our progress on the UI thread
  }
}
```

What thread does this run on?

What thread does this run on?

What thread does this run on?

AsyncTask

- AsyncTask: simplest way to do background work

```
// subclass an AsyncTask
class MyTask extends AsyncTask<URL, Integer, Long> {
  Long doInBackground(URL … urls);
  void publishProgress(Integer … values);
  void onProgressUpdate(Integer … progress);
  void onPostExecute(Long result);
}

// to use an AsyncTask
new MyTask.execute(url1, url2, url3);
```

What thread does this run on?

AsyncTask

- AsyncTask: simplest way to do background work

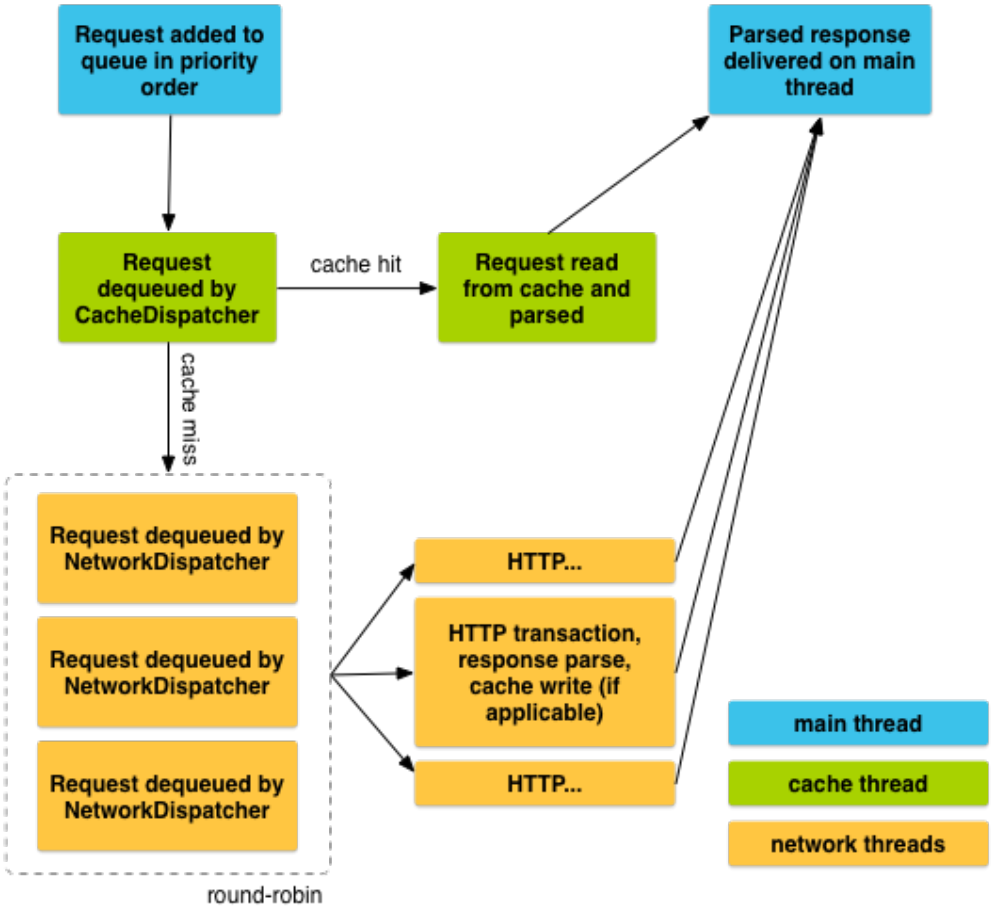
```
// subclass an AsyncTask
class MyTask extends AsyncTask<URL, Integer, Long> {
  Long doInBackground(URL … urls);
  void publishProgress(Integer … values);
  void onProgressUpdate(Integer … progress);
  void onPostExecute(Long result);
}

// to use an AsyncTask
new MyTask.execute(url1, url2,      );
```

AsyncTask hides almost all of the Handler stuff

Tools at your disposal

- Loopers/Handlers
  - Gives you the most control
  - Can run arbitrary code on a thread
- AsyncTasks
  - Simple, easy way to run code on a thread
  - May need more control than it offers
- Volley
  - Simpler, easier way to make HTTP requests in the background

Volley request lifecycle



Tools at your disposal

- Loopers/Handlers
  - Gives you the most control
  - Can run arbitrary code on a thread
- AsyncTasks
  - Simple, easy way to run code on a thread
  - May need more control than it offers
- Volley
  - Even simpler, easier way to make HTTP requests in the background
- Let's use AsyncTasks to move SQLite off UI thread