Threads / AsyncTask

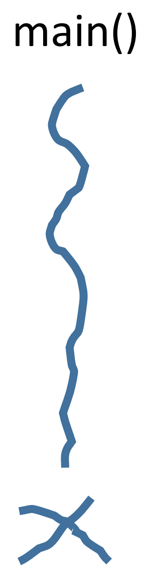
Goal

- Learn everything needed to complete the Family Map Login assignment

- Thread concepts
- Android AsyncTasks
- Family Map Login Architecture

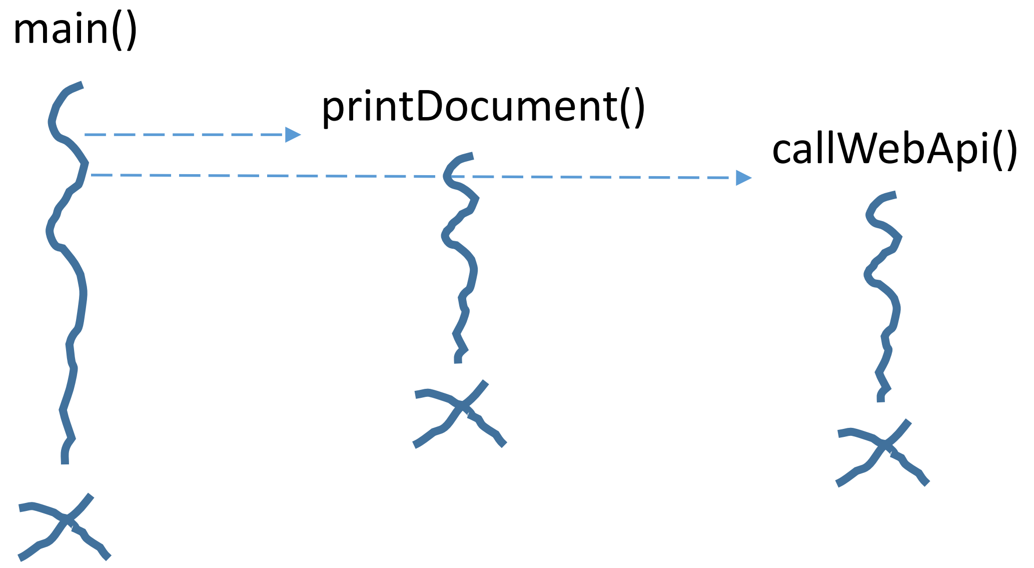
Threads

- By default, programs do one thing at a time
- They start executing in main(), and when main() completes, the program terminates



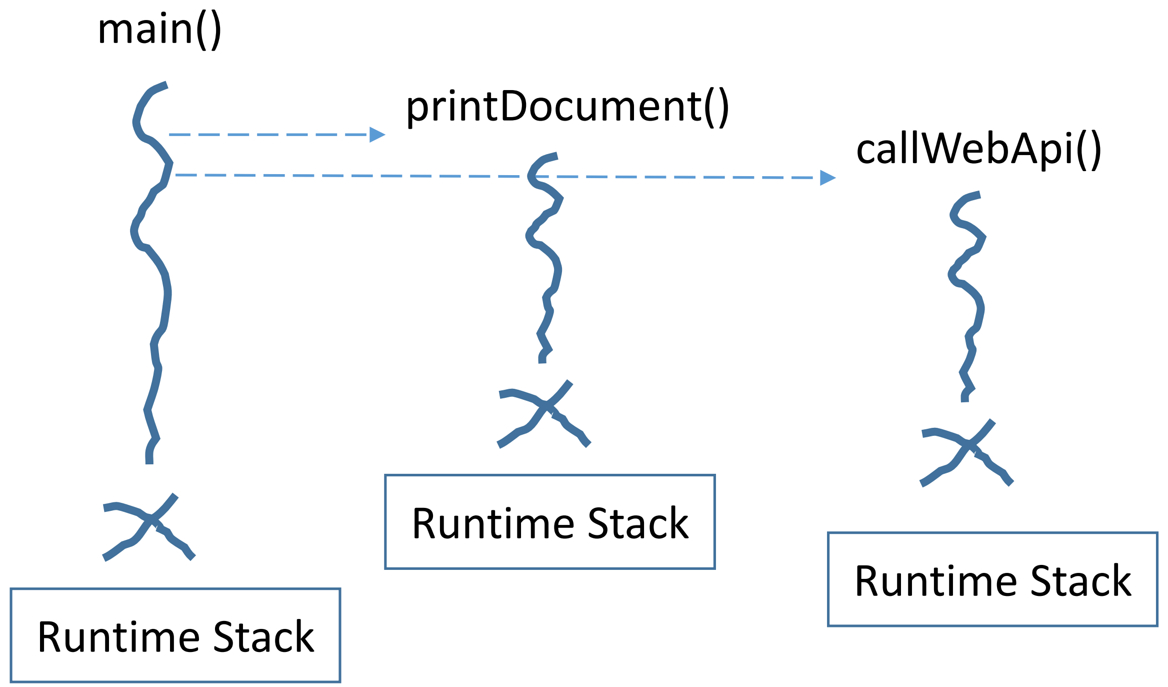
Threads

- Often, it is desirable for a program to do multiple things at the same time (concurrently, in parallel)
- To do this, a program can create multiple “threads” of control, each of which represents something the program is working on



Threads

- A program starts with one “main” thread
- Additional threads can be created as needed
- Each thread has its own runtime stack, so it can run independently from the other threads

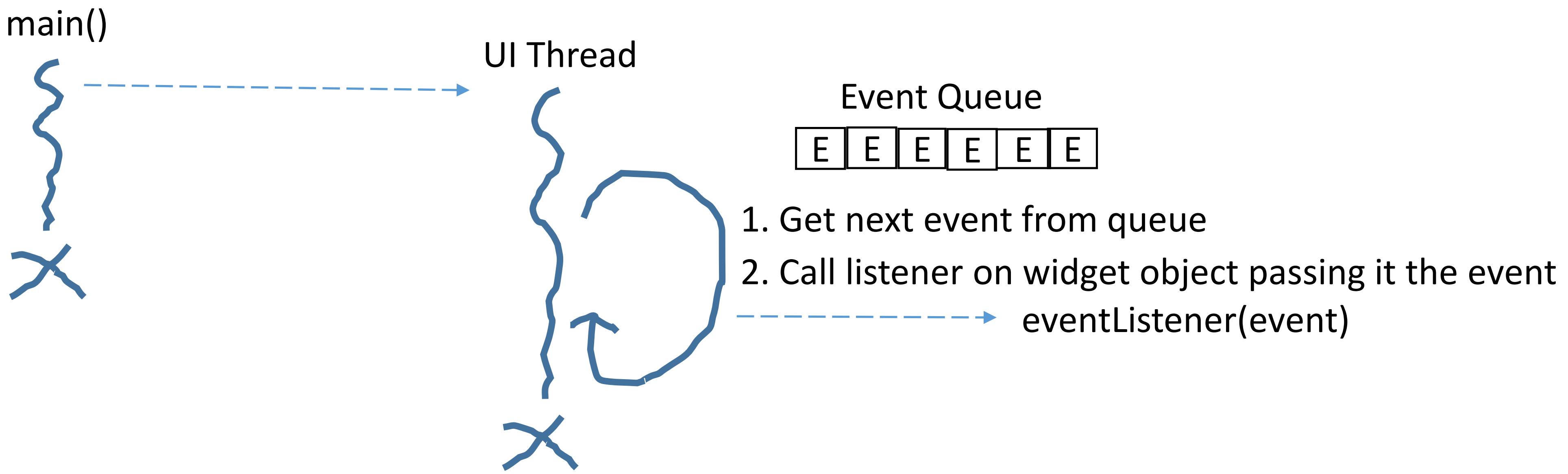


Java Threads Example

```
public class JavaThreadExample {

        public static void main(String[] args) {

                CountingThread countUp = new CountingThread("UP", 0, 50, 1);
                CountingThread countDown = new CountingThread("DOWN", 50, 0, -1);

                countUp.start();
                countDown.start();

                System.out.println("Leaving Main Thread");
        }
}


class CountingThread extends Thread {

        private String _name;
        private int _start;
        private int _stop;
        private int _increment;

        public CountingThread(String name, int start, int stop, int increment) {
                _name = name;
                _start = start;
                _stop = stop;
                _increment = increment;
        }

        public void run() {
                for (int i = _start; i != _stop; i += _increment) {
                        System.out.println(_name + ": " + i);
                }
        }
}
```

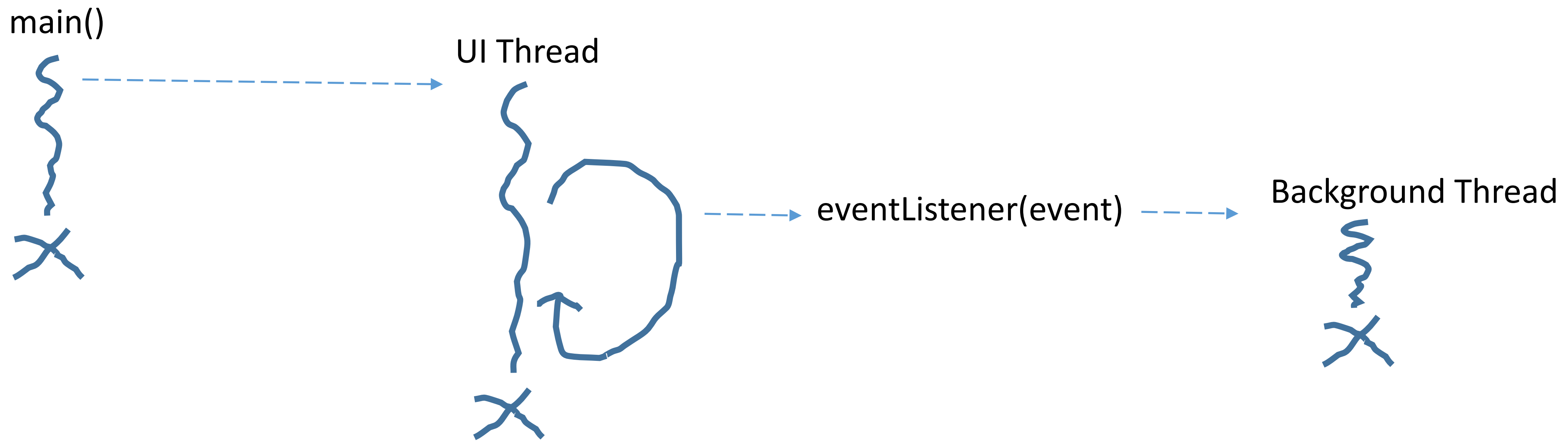
User Interface Thread

- In a program with a graphical user interface (GUI), there is a special "UI thread" that processes all user interface activity
- All method calls on widget objects must be done on the UI thread
- All event listeners are called on the UI thread by the UI system (Android)


main()
UI Thread
Event Queue
E E E E E E
1. Get next event from queue
2. Call listener on widget object passing it the event
eventListener(event)

User Interface Thread

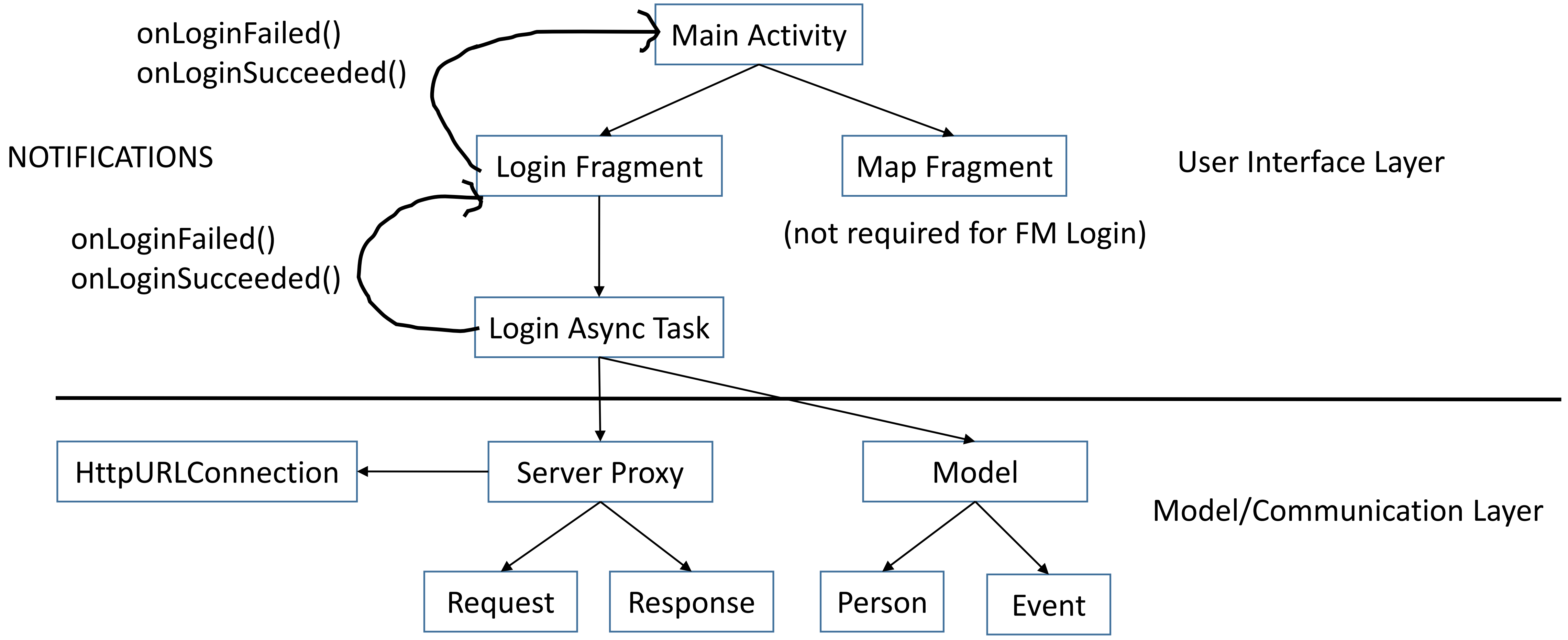
- Event listeners should return quickly so they don't tie up the UI thread (which would cause the UI to "freeze" and become unresponsive)
- If an event listener needs to do something that takes a long time (print a document, call a web API, etc.), it should do so on a background thread



Android AsyncTasks

- In Android, you can use regular Java threads to do background processing
- However, Android also provides a class called AsyncTask, which is better for background threads that need to perform UI operations (e.g., provide user feedback using progress bar, busy icon, etc.)
- AsyncWebAccess example
- AsyncWebAccess with Listener example

Family Map Login Architecture


onLoginFailed()
onLoginSucceeded()
NOTIFICATIONS
onLoginFailed()
onLoginSucceeded()
Main Activity
Login Fragment
Map Fragment
(not required for FM Login)
User Interface Layer
Login Async Task
HttpURLConnection
Server Proxy
Model
Model/Communication Layer
Request
Response
Person
Event