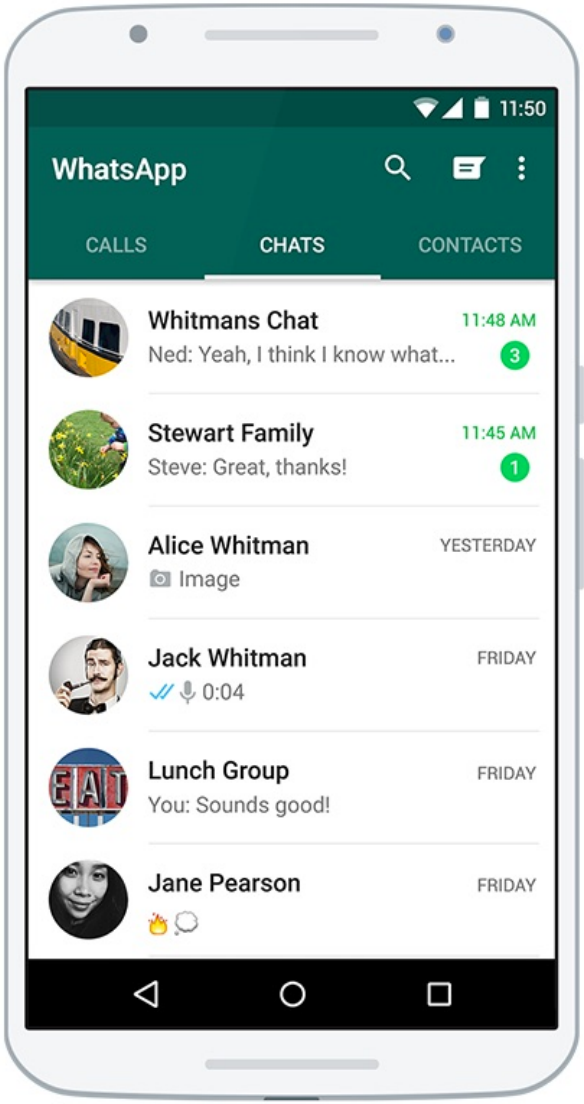
G53MDP
Mobile Device Programming

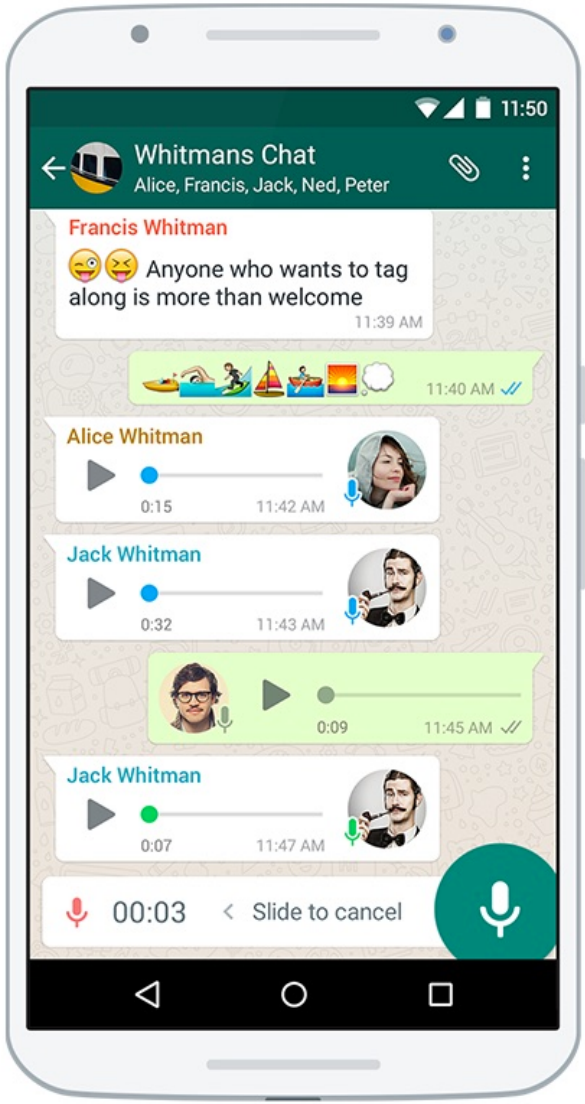
Android Application Components

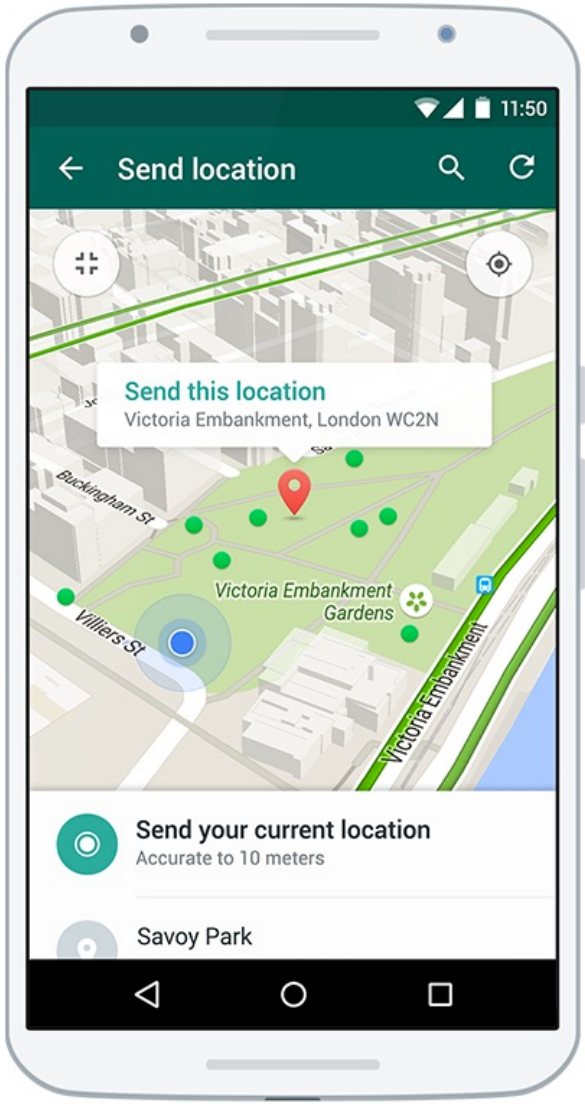
Activities

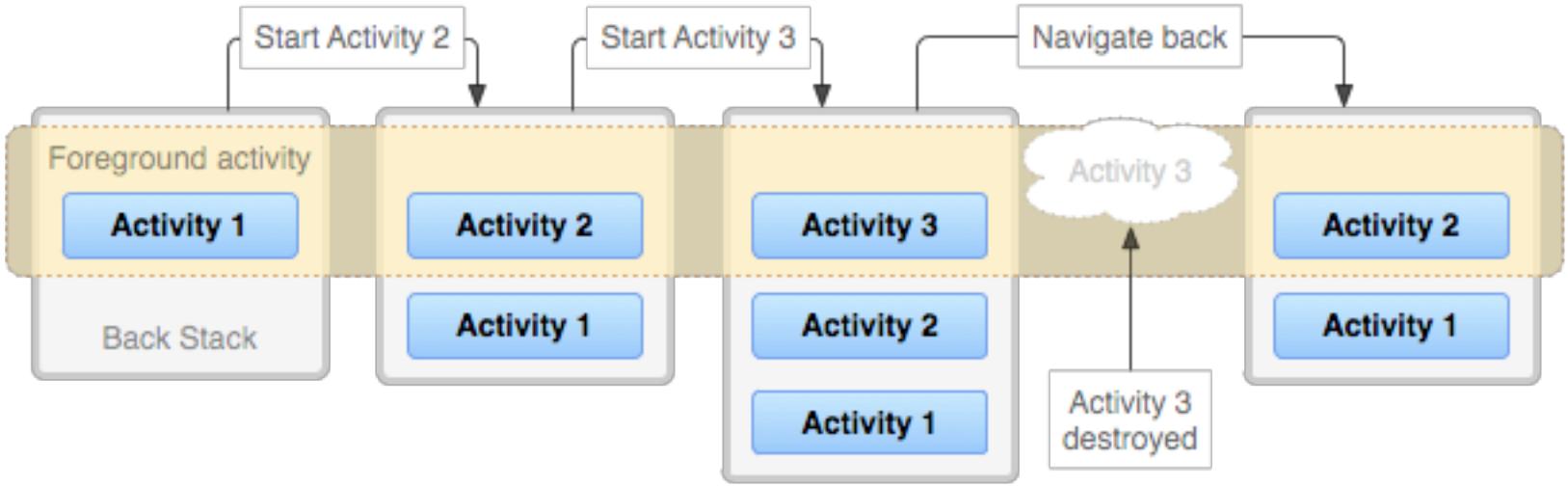
Tasks vs Activities

- Activities can start other activities
  - Forms a stack of Activities — current activity is on the top
    - The ”Back Stack”
- Multiple activities form a Task
  - Such as…?
  - A Task may span multiple applications
    - Make use of *standard* activities
    - Make use of *3rd party* activities
      - User choice
- An activity should be an **atomic** part of a particular task
  - Supports reuse, integration into multiple tasks
- Activities in a task move as a unit from foreground to background and vice versa
  - Switching task = switching stack


11:50
WhatsApp
CALLS
CHATS
CONTACTS
Whitmans Chat
11:48 AM
Ned: Yeah, I think I know what...
3
Stewart Family
11:45 AM
Steve: Great, thanks!
1
Alice Whitman
YESTERDAY
Image
Jack Whitman
FRIDAY
0:04
Lunch Group
FRIDAY
You: Sounds good!
Jane Pearson
FRIDAY

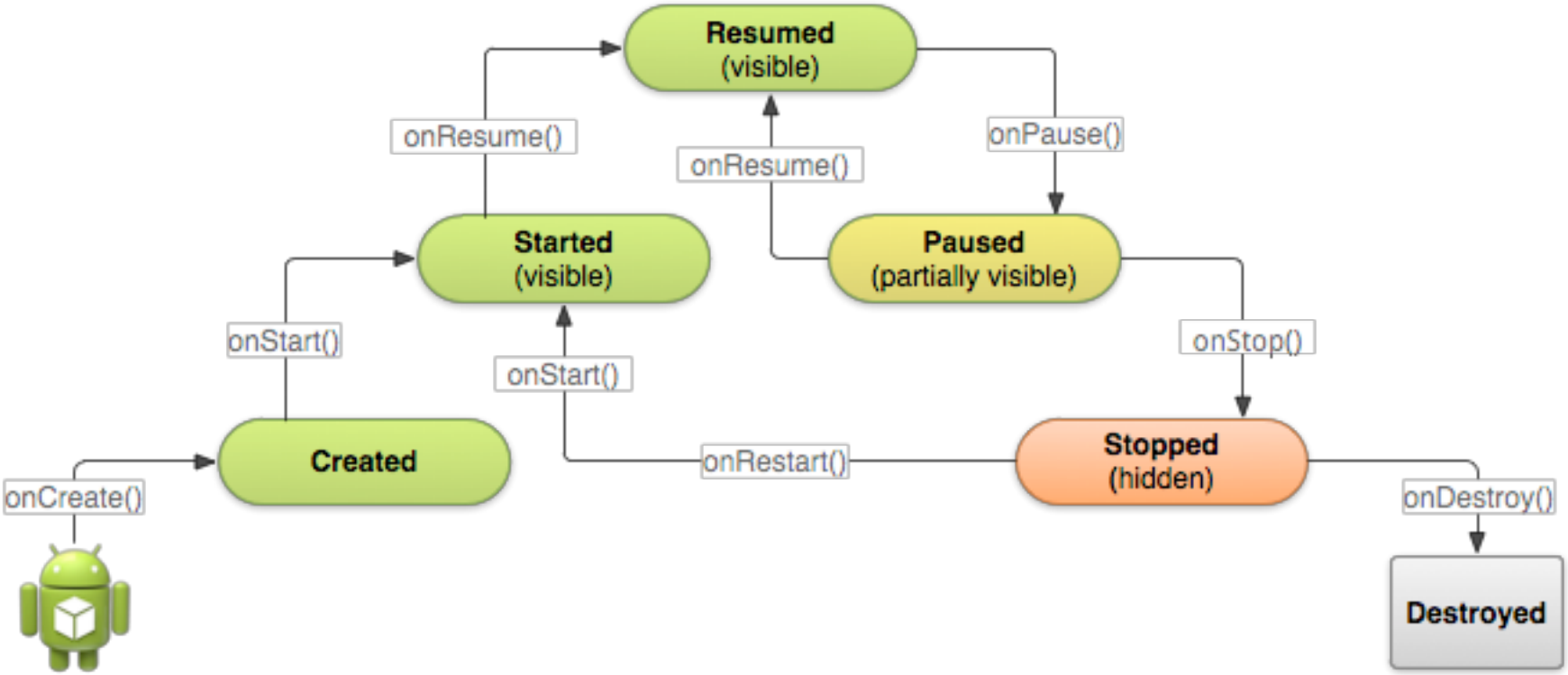

11:50
Whitmans Chat
Alice, Francis, Jack, Ned, Peter
Francis Whitman
Anyone who wants to tag along is more than welcome
11:39 AM
11:40 AM
Alice Whitman
0:15
11:42 AM
Jack Whitman
0:32
11:43 AM
0:09
11:45 AM
Jack Whitman
0:07
11:47 AM
00:03
Slide to cancel


11:50
Send location
Send this location
Victoria Embankment, London WC2N
Buckingham St
Victoria Embankment Gardens
Villiers St
Victoria Embankment
Send your current location
Accurate to 10 meters
Savoy Park


Start Activity 2
Start Activity 3
Navigate back
Foreground activity
Activity 1
Back Stack
Activity 2
Activity 1
Activity 3
Activity 2
Activity 1
Activity 3
Activity 3 destroyed
Activity 2
Activity 1

Activity Lifecycle

- Essentially in one of three states
  - Active
    - in the foreground
  - Paused
    - still visible, but not top
      - Split screen
  - Stopped
    - obscured by another activity
- If paused or stopped, the system can drop the Activity from memory
  - Stopped activities are suspended in memory
    - Consume no processing resources
  - Inactive activities are destroyed if memory is required
    - Last recently used
- Transitioning between events generates events
  - For us to do something
    - Setup UI
    - Save state


Resumed
(visible)
onResume()
onResume()
onPause()
Started
(visible)
Paused
(partially visible)
onStart()
onStart()
onStop()
Created
onRestart()
Stopped
(hidden)
onCreate()
onDestroy()
Destroyed

Intents

- Don’t instantiate the Activity sub-class
  - Android works by passing Intent objects around
    - Intent is used to describe an operation
      - Action and the data to operate on (as a URI)
    - Allows for late runtime binding
    - Glue multiple activities together
  - Android figures out how to honour the Intent
    - Inspecting registered **manifests**
- Starting an Activity
  - Create a new Intent object
  - Specify what you want to send it to
    - Either implicitly, or explicitly
  - Pass the Intent object to **startActivity()**
    - New Activity then started by the runtime
- Stopping an Activity
  - The called Activity can return to the original one by destroying itself
    - By calling the method **finish()**
  - Or when the user presses the back button

Let’s have a look…

Intents

- Explicit Intent

```
Intent myIntent = new Intent(context, otherActivity.class);
startActivity(myIntent);
```

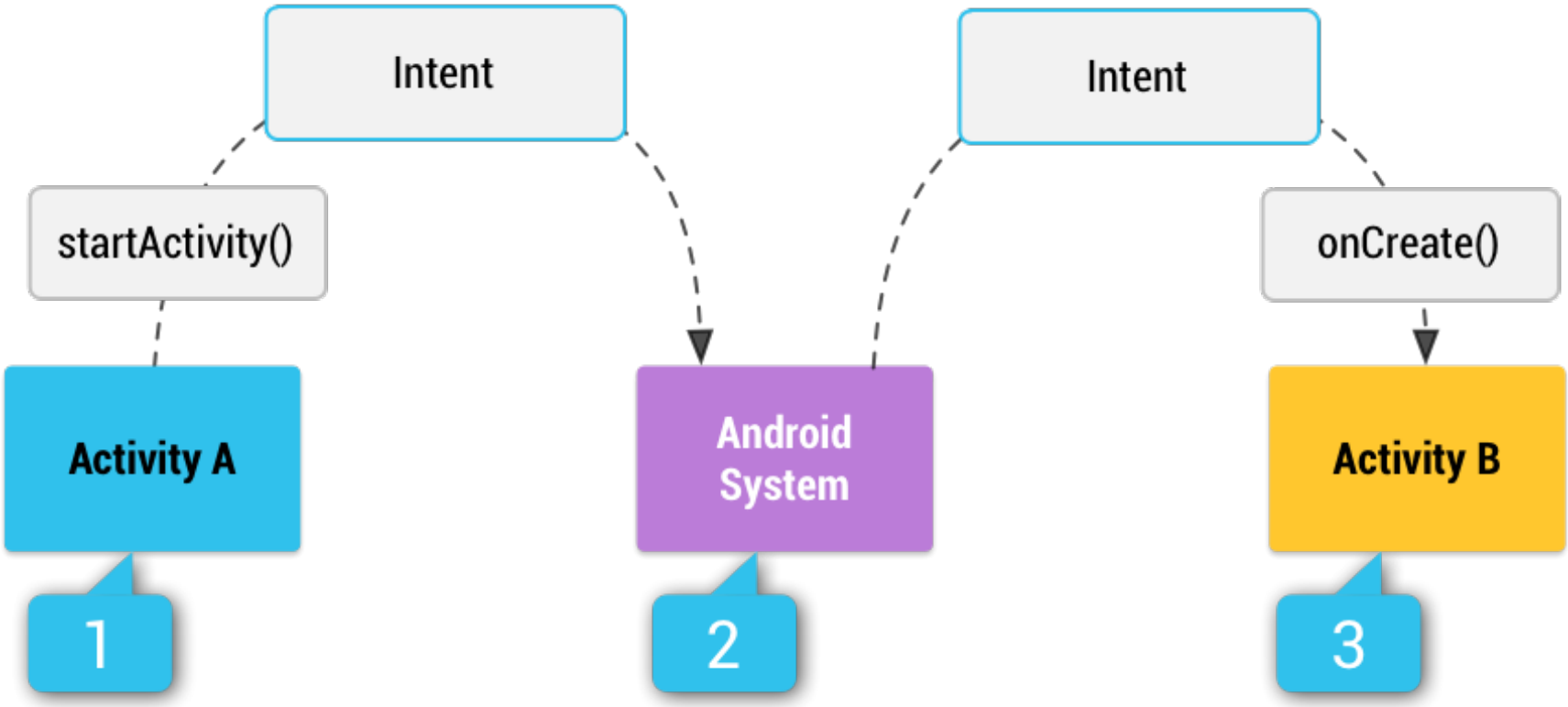
- Implicit Intent
  - Data and action are used to resolve the most appropriate activity

```
Uri webpage = Uri.parse("http://www.cs.nott.ac.uk");
Intent myIntent = new Intent(Intent.ACTION_VIEW, webpage);

Uri number = Uri.parse("tel:01151234567");
Intent myIntent = new Intent(Intent.ACTION_DIAL, number);

startActivity(myIntent);
```

Starting an Activity



Intent Filters

- Manifest specifies *intent filters*
  - Determine which Intents should be handled by the Activity class
  - Match on action, schema of the URI
  - Multiple matches?
    - Allow the user to choose “Application”
    - This could be deliberate

```
<activity android:name="com.example.martinactivities.ForthActivity" >
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:scheme="http" />
    </intent-filter>
</activity>
```

Let’s have a look…

Inter-Activity Communication

- startActivity() doesn't allow the Activity to return a result
  - Applications usually want to maintain state
    - Remember what the user has done across all activities
    - We could store state in the broader Application context
      - But activities may be communicating between processes (IPC)
      - Entry point for other applications
- startActivityForResult()
  - Still takes an Intent object, but also a numerical request code
    - Returns an integer result code, set with setResult()
- onActivityResult() then called on the calling Activity
  - Data can be packaged up in an Intent / Bundle
    - Activity creates an Intent object containing the result
    - Use a Bundle to "bundle" complicated objects
  - Calls setResult() to return the Intent
  - Intent object then passed to onActivityResult() on finish

Inter-Activity Communication

- We've decomposed a task into multiple activities
  - How do Activities communicate?
  - String otherClass.doStuff(String arg2, MyObject b);
  - Potentially cross-process (IPC)
    - Across memory boundaries enforced by the kernel
  - NB! **Classes** vs **Activities**
- startActivity()
- startActivity(send some data to the new activity)
- startActivityForResult()
- startActivityForResult(send some data)
  - …expect some data back

Let’s have a look…

References

- http://developer.android.com/training/basics/activity-lifecycle/index.html
- http://developer.android.com/guide/components/processes-and-threads.html
- https://developer.android.com/guide/components/activities/process-lifecycle.html