SENG440

Week 4 – Intents and Navigation

Ben Adams (benjamin.adams@canterbury.ac.nz)

Intents

- `Intents` are how activities communicate with other activities in Android
- **Implicit** `Intent`
  - Request for a service
  - Any appropriate app on device can fulfil the request
    - E.g. send an email, set an alarm, or capture a picture with the camera
- **Explicit** `Intent`
  - Start a specific named `Activity`
  - E.g. another Activity class defined in your app
- Intents can return values to your activity

Common Intents

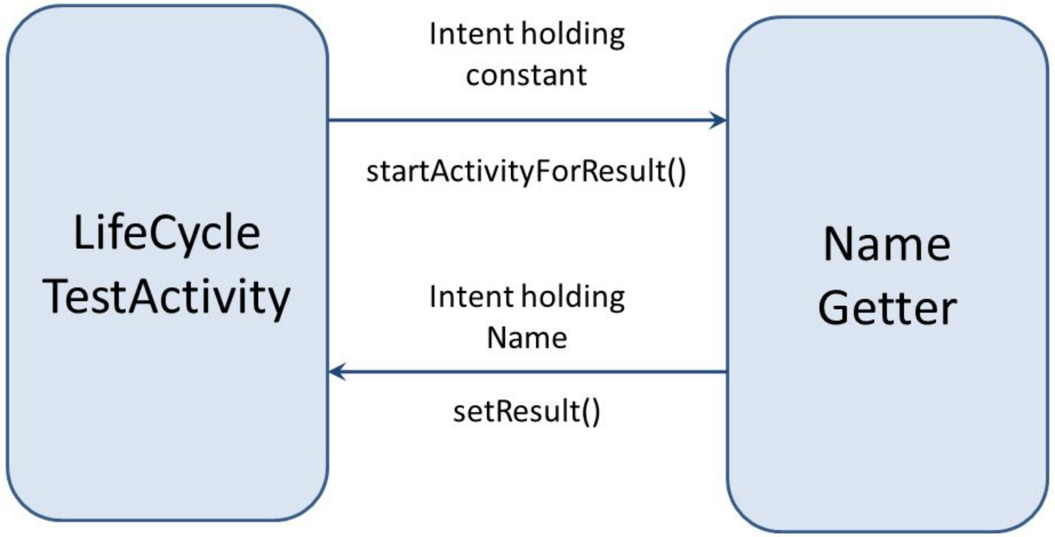
- Allow us to use applications and components that are already part of Android System
  - Start activities
  - Start services
  - Deliver broadcasts
- … and allow other applications to use the components of the applications we create
- Common intents: AlarmClock, Calendar, Camera, Contacts/People App, Email, File Storage, Maps, Music or Video, New Note, Phone, Search, Settings, Web Browser

Intents

- “An intent is an abstract description of an operation to be performed”
- Consist of two elements:
  - **Action** (what to do, example visit a web page)
  - **Data** (to perform operation on, example the url of the web page)
- Use via these methods:
  - `startActivity`
  - `startActivityForResult`
  - `startService`
  - `bindService`

Activation of components

- **3 core application components**: activities, services, and broadcast receivers, are started via *intents*.
- Intents are a **messaging system to activate components** in the same application
- *and* to **start one application** from another



AndroidManifest.xml Purpose

- Contains Java package **name of application** - unique id for application
- Describes components of application:

  activities, services, broadcast receivers, content providers, and ***intent messages each component can handle.***
- Declares **permissions** requested by application
- **Libraries** that application links to

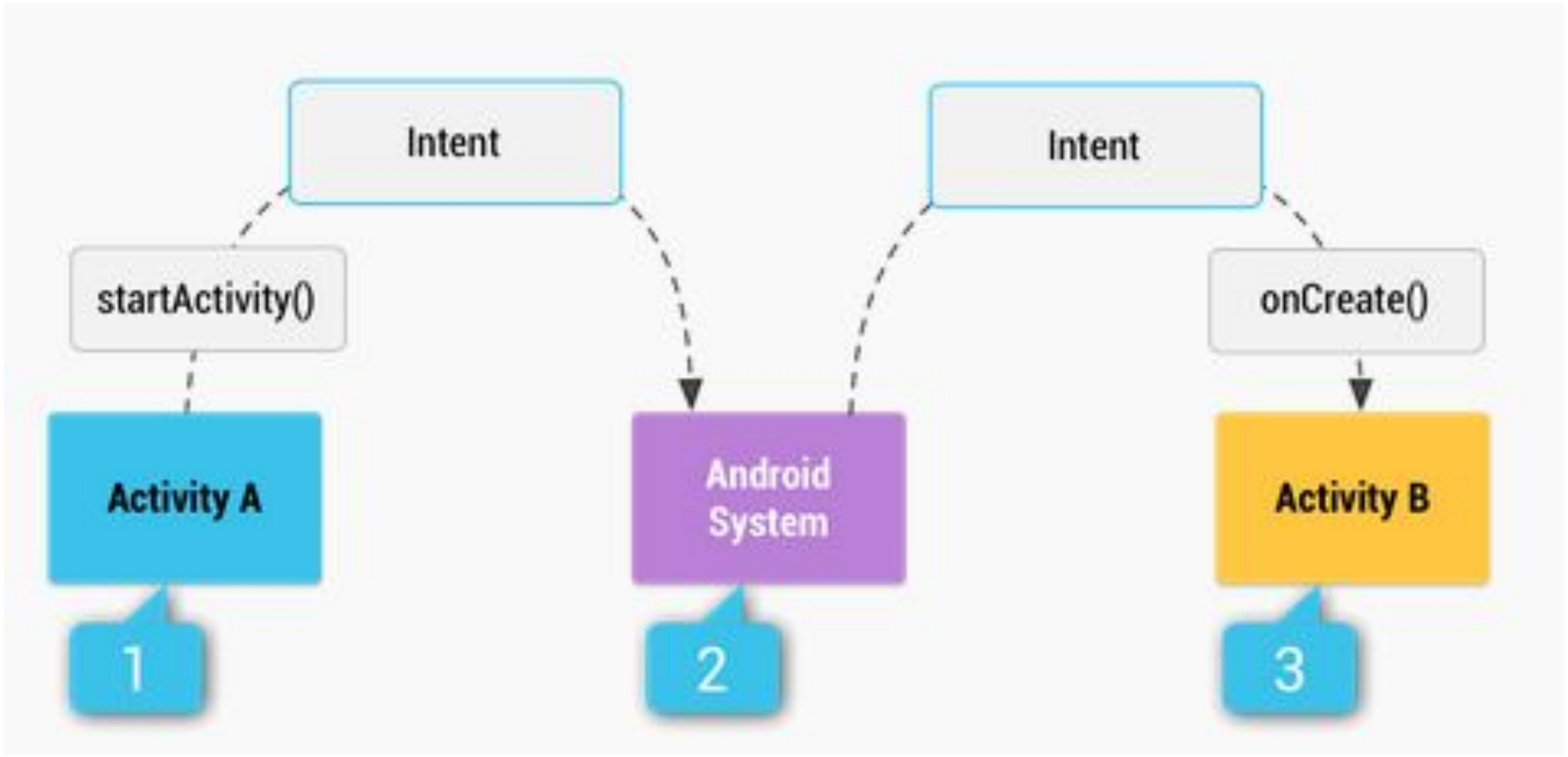
Types of Intents

https://developer.android.com/reference/android/content/Intent.html

- **Action categories:**
  - E.g. `ACTION_VIEW, ACTION_EDIT, ACTION_CALL, ACTION_DIAL, ACTION_SEARCH…`

- **Broadcast actions:**
  - E.g. `ACTION_TIMEZONE_CHANGED, ACTION_POWER_CONNECTED, …`

Implicit Intents

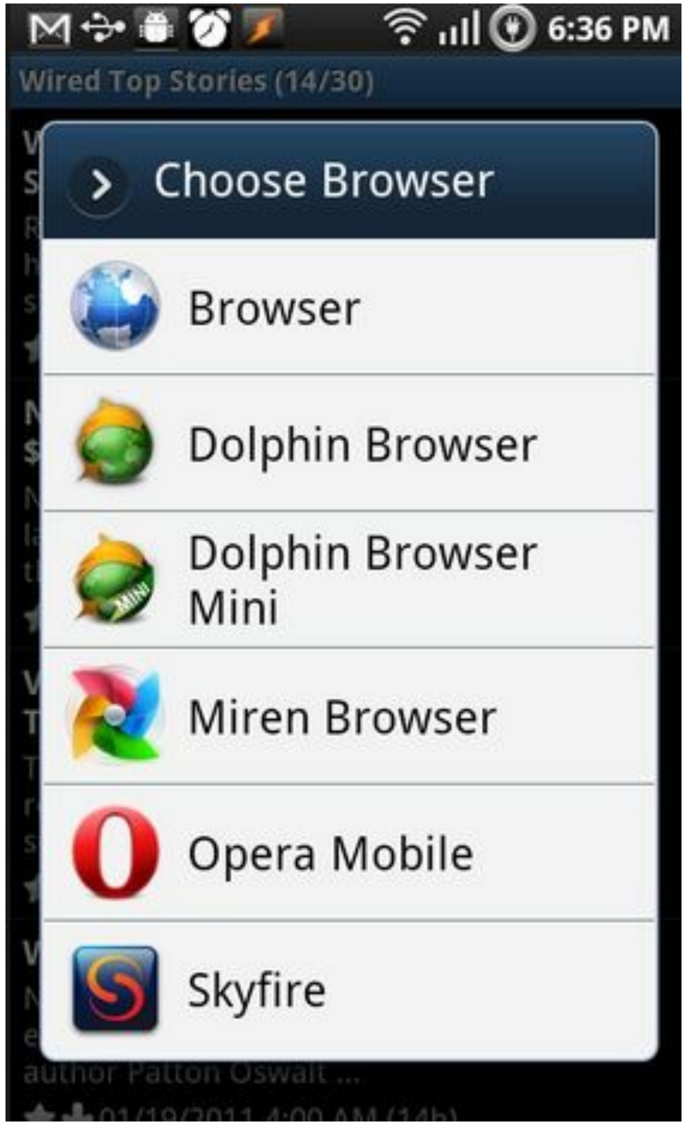


https://developer.android.com/guide/components/intents-filters.html

Multiple apps can handle intent

- Intent class contains **constants for Intents**
- Applications and Activities **list intents they can handle** in manifest
  - Intent Filters
- If multiple Activities are available, the **user is asked to choose**

`android.intent.action.WEB_SEARCH □`



`Intent` class and objects

- `android.content.Intent`
- **Passive** data structure
    - Description of **action to be performed**,
    - Or description of something that has happened and is being **announced to broadcast receivers**.
- Intent objects **carry information**
- **Do not *perform*** any **actions** themselves
    - It is the job of the OS (Android system) to determine what is done with an Intent

Intents and App Components

Intent to Launch Activity or change purpose of existing Activity →

```
Context.startActivity()
Activity.startActivityForResult()
Activity.setResult()
```

Intent to Initiate Service or give new instructions to existing Service →

```
Context.startService()
Context.bindService()
```

Intents sent to Broadcast Receivers →

```
Context.sendBroadcast()
Context.sendOrderedBroadcast()
Context.sendStickyBroadcast()
```

Intent object properties

- **component name** (of desired component)
- **action** (to execute)
- category (of action)
- **data** (to work on)
- type (of intent data)
- **extras** (a Bundle with more data)
- flags (to help control how Intent is handled)

Intent information

- **Information for the component** that receives the intent:
  - Action to take
  - Data to act on
- **Information for the Android system**:
  - **Category of component** to handle intent: activity, service, broadcast receiver
  - **Instructions on how to launch** component, if necessary

Intent **component name**

- Fully-qualified class name of component
- The package name set in the manifest file of the application where the component resides
- Optional!
  - If not provided, then Android system resolves suitable target
- name is set by `setComponent()`, `setClass()`, or `setClassName()`

Intent **action**

- Action desired (or for broadcast intents, the action / event that took place)
  - None for explicit intents
- Many actions defined in Intent class
- Other actions defined through the API
  - E.g., MediaStore class declares `ACTION_IMAGE_CAPTURE` and `ACTION_VIDEO_CAPTURE`
- You can define your own Intent Action names so other applications can activate the components in your application
- Action acts like a method name
- Determines what rest of data in Intent object is and how it is structured, especially the *data* and *extras*
- `setAction()` and `getAction()` methods from Intent class

Intent **data**

- URI (uniform resource identifier) of data to work with / on
  - For content on device a content provider and identifying information, for example an audio file or image or contact
  - E.g., `geo:0,0?q=Christchurch%2C%20New%20Zealand`
- MIME type of data / content
  - E.g., `image/png` or `audio/mpeg`

Intent **extras**

- A `Bundle` (**dictionary of key/value pairs**) of additional information to be given to the component handling the `Intent`
- **Often user-defined** for explicit intents in your App
- Some Actions will have specified extras:
  - `ACTION_TIMEZONE_CHANGED` will have an extra with key of "time-zone" (documentation is your friend)
  - Intent method has `put` method for individual key/value pairs, or
  - Can `put` a whole `Bundle`

Multiple activities in your App

- One model of multi-screen apps is to define **multiple Activities** in your app and use **explicit Intents** to move from one to the other.

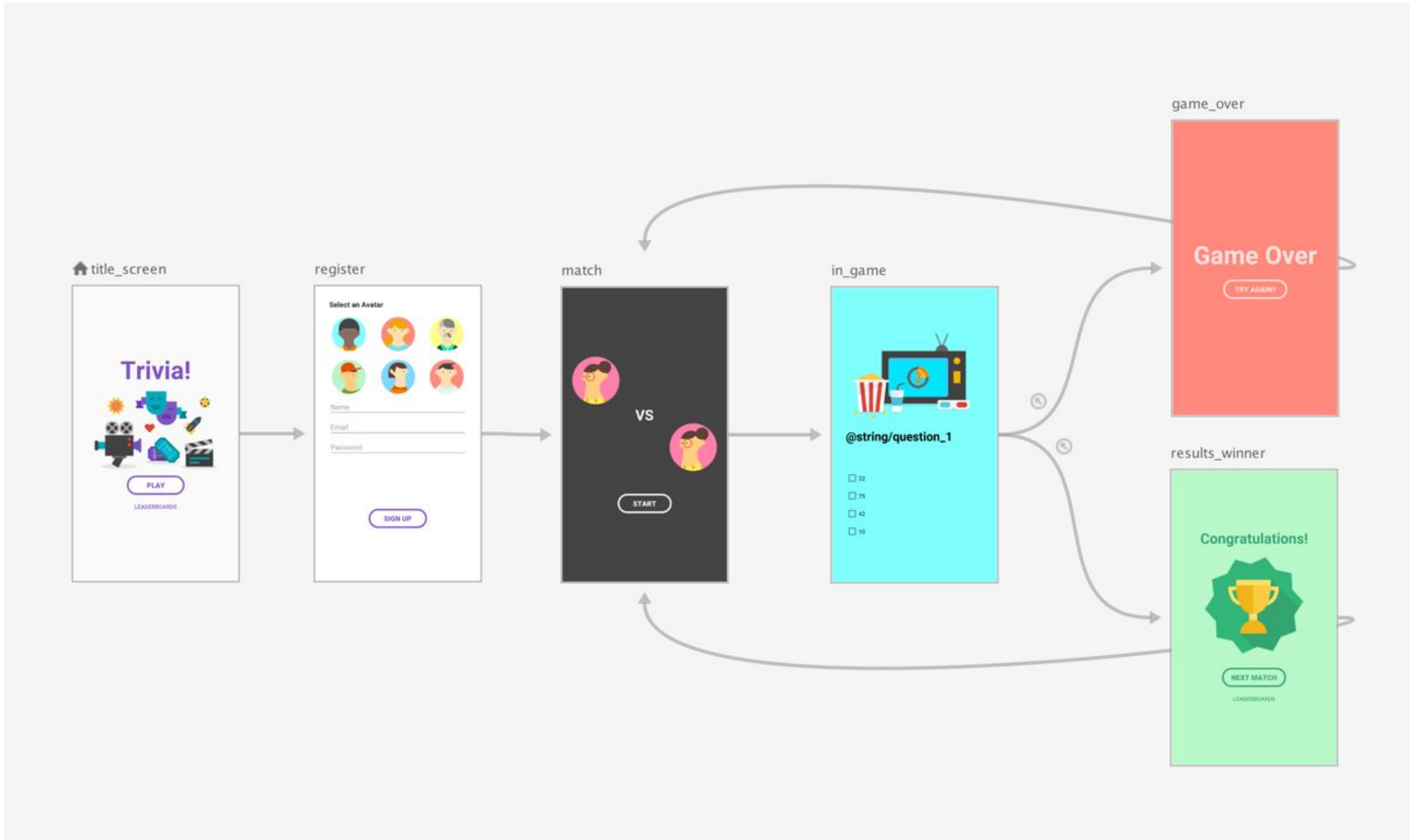
- Intents used to **start activities** and **pass data** between the activities (e.g., using extras).
    - See Kana-san tutorial example this week

Navigation architecture

- Navigation graph
    - **Destinations** are nodes in the graph (pages of your app)
    - **Actions** are edges between nodes denoting logical relationships between destinations
    - Nested graphs
- NavHost
    - Empty container in layout that holds a destination
    - Destinations swapped in and out
    - Default implementation `NavHostFragment`
- Deep links

[Get started with the Navigation component | Android Developers]

Navigation graph



Navigation in Compose

- `NavHost` Composable defines a set of routes for our application
  - cf. routing in web frameworks such as React

- Managed by a `NavController` which can be used to programmatically navigate to different routes based on events

- Argument placeholders allow you to pass parameters to a screen based on data

Ref: Navigating with Compose

Navigation with Fragments and Compose

- Compose code can be added to any XML layout using `ComposeView`
- A `ComposeView` is a View element that implements the `setContent` method where you can add composable functions

XML

```
<androidx.compose.ui.platform.ComposeView
    android:id="@+id/composeView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Kotlin

```
binding.composeView.setContent {
    Text(text = myText)
}
```

- Implement `AbstractComposeView` to create a custom class
- Put `ComposeView` into a Fragment and use XML Navigation graph to organize your application

Todo for next week

- Keep working on your app. Please post updates/questions on the Slack channel
- Read Getting started with Navigation, Navigating with Compose.
- Work your way through Connect440, NavTest, and Kana san app tutorials.
    - For Connect440 you might find it useful to read more of the Android documentation about RecyclerView.
    - For Compose version of Connect440, more information on Lists can be found here: https://developer.android.com/jetpack/compose/lists