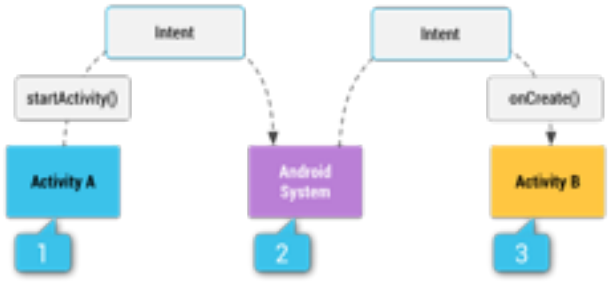
Intent and Intent Filters

- a messaging object that facilitates communication between components i

Fundamental use cases for communication:

1. Starting an Activity
   - start a new instance of an Activity by passing an Intent to startActivity(). The Intent describes the activity to start and carries any necessary data.
2. Starting a service
   - You can start a service to perform a one-time operation (such as downloading a file) by passing an Intent to startService(). The Intent describes the service to start and carries any necessary data.
   - If the service is designed with a client-server interface, you can bind to the service from another component by passing an Intent to bindService()
   - To ensure that your app is secure, always use an explicit intent when starting a Service and do not declare intent filters for your services.
3. Delivering a broadcast
   - You can deliver a broadcast to other apps by passing an Intent to sendBroadcast() or sendOrderedBroadcast().



[1] Activity A creates an Intent with an action description and passes it to startActivity().

[2] The Android System searches all apps for an intent filter declared in the manifest that matches the intent. When a match is found, (If multiple intent filters are compatible, the system displays a dialog so the user can pick which app to use)

[3] the system starts the matching activity (Activity B) by invoking its onCreate() method and passing it the Intent

Building an intent

1. Component name
   - The name of the component to start
   - Optional, but critical information that makes and intent explicit - delivered only to the app component defined by the name
   - Without a component name, the intent is implicit
   - set the component name with setComponent(), setClass(), setClassName(), or with the Intent constructor
2. Action
   - A string that specifics the generic action to perform (view, pick)
   - In the case of a broadcast intent, this is the action that took place and is being reported
   - You can specify the action for an intent with setAction() or with an Intent constructor.
3. Data
   - The URI (a Uri object) that references the data to be acted on and/or the MIME type of that data
   - To set only the data URI, call setData(). To set only the MIME type, call setType(). If necessary, you can set both explicitly with setDataAndType()
   - Caution: don't call setData() and setType() because they each nullify the value of the other, use setDataAndType() to set both URI and MIME type.
4. Category
   - A string containing additional information about the kind of component that should handle the intent.
   - Any number of category descriptions can be placed in an intent, but most intents do not require a category
   - You can specify a category with addCategory().
5. Extras
   - Key-value pairs that carry additional information required to accomplish the requested action
   - You can add extra data with various putExtra() methods, each accepting two parameters: the key name and the value.
   - You can also create a Bundle object with all the extra data, then insert the Bundle in the Intent with putExtras().

6. Flags

   - Flags are defined in the Intent class that function as metadata for the intent. The flags may instruct the system how to launch an activity and how to treat it after it's launched

   - For more information, see the setFlags() method.

Intent Types

- Explicit Intents

  - specify the component to start by name (the fully-qualified class name). You'll typically use an explicit intent to start a component in your own app,

  - For example, if you built a service in your app, named DownloadService, designed to download a file from the web, you can start it with the following code:

```
// Executed in an Activity, so 'this' is the Context
// The fileUrl is a string URL, such as "http://www.example.com/image.png"
Intent downloadIntent = new Intent(this, DownloadService.class);
downloadIntent.setData(Uri.parse(fileUrl));
startService(downloadIntent);
```

  - The Intent(Context, Class) constructor supplies the app Context and the component a Class object. As such, this intent explicitly starts the DownloadService class in the app

- Implicit intents

  - do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it.

  - For example, if you have content that you want the user to share, create an intent with the ACTION_SEND action and add extras that specify the content to share. When startActivity() is called with that intent, the user can pick an app through which to share the content

  - call resolveActivity() on your Intent object, to verify that the intent will be received, If the result is non-null, there is at least one app that can handle the intent and it's safe to call startActivity()

```
// Create the text message with a string
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(sendIntent);
}
```

- Receiving an implicit intent
- Using a pending intent
- Intent Resolution
