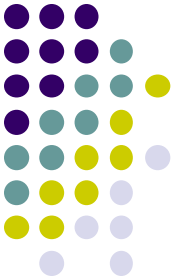
Mobile Application Develpment

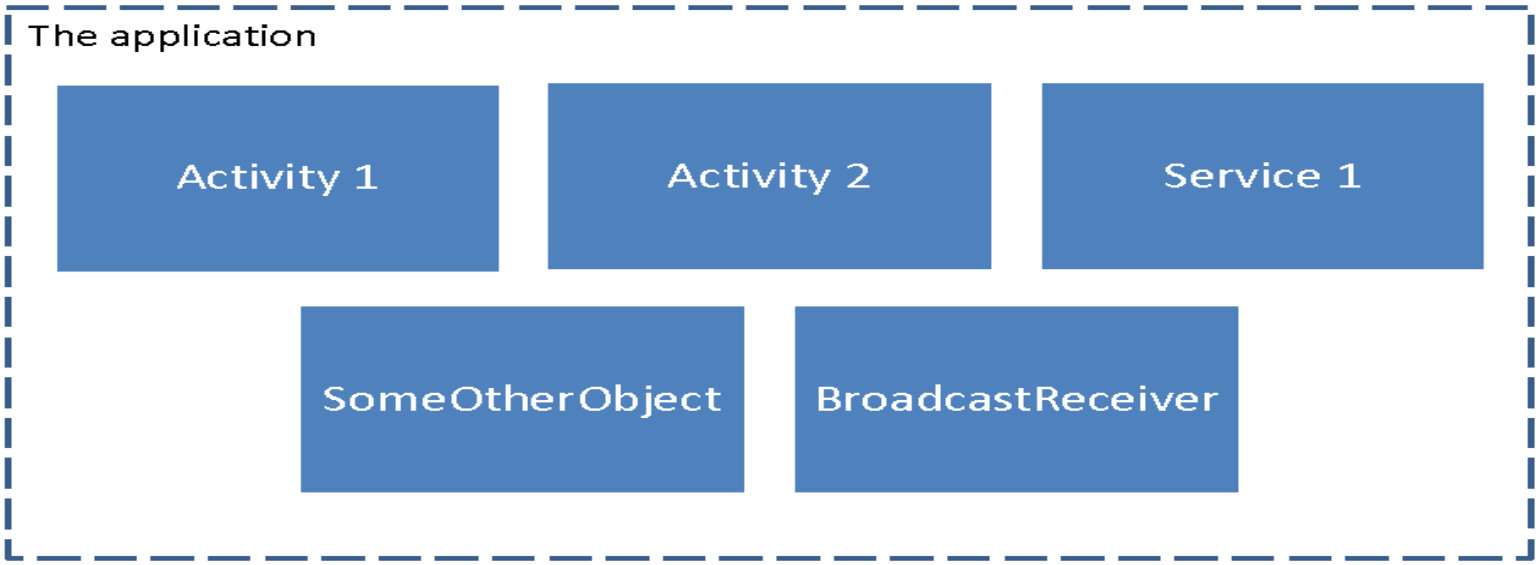
Multiple Activities and Intents

Multiple Activities



Many apps have multiple activities.

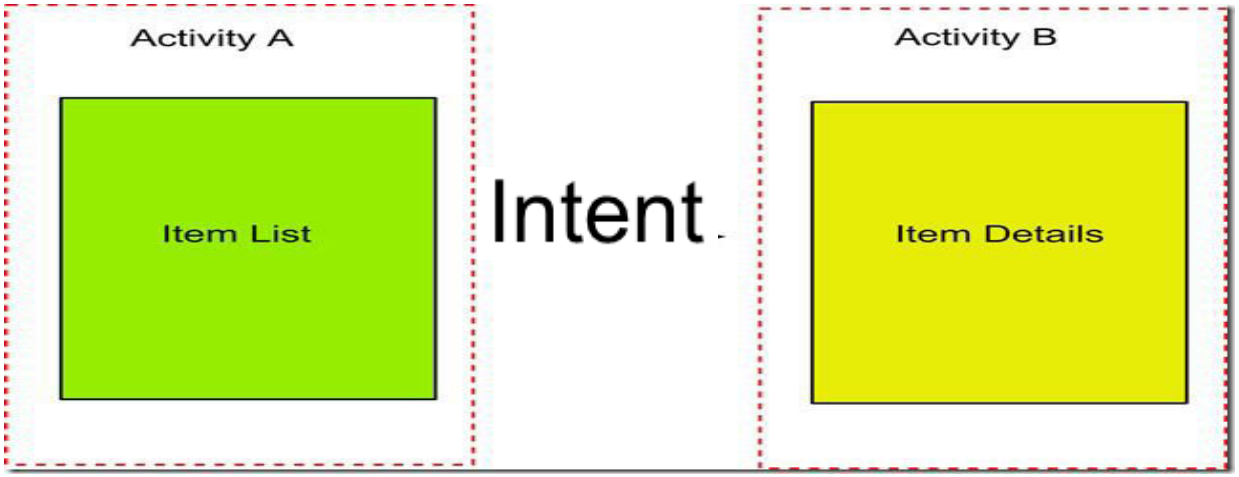
- **Example:** In an **address book app**, the **main activity** is a list of contacts, and clicking on a contact goes to another **activity** for **viewing details**.
- An **activity 1** can **launch another activity 2** in response to an **event.**
- The **activity 1** can **pass data** to **activity 2**.
- The **second activity 2** can **send data back** to **activity 1** when it is done.



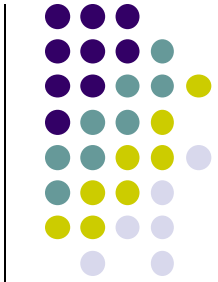
Intent

Intent: **a bridge between activities**; a way for one activity to invoke another

- The **activity** can be in the **same app** or in a **different app.**
- **Intent** can store **extra data** to pass as **"parameters"** to that activity.
- **Second activity** can **"return"** information back to the caller if needed.
- **Intent** is use for **broadcasting messages** between **Android OS** and applications (battery is low).
- Start a service(open email, web browser& calling).



Intent Structure



```
public class Intent
extends Object implements Parcelable, Cloneable
```

java.lang.Object
↳ android.content.Intent

Known direct subclasses
LabeledIntent

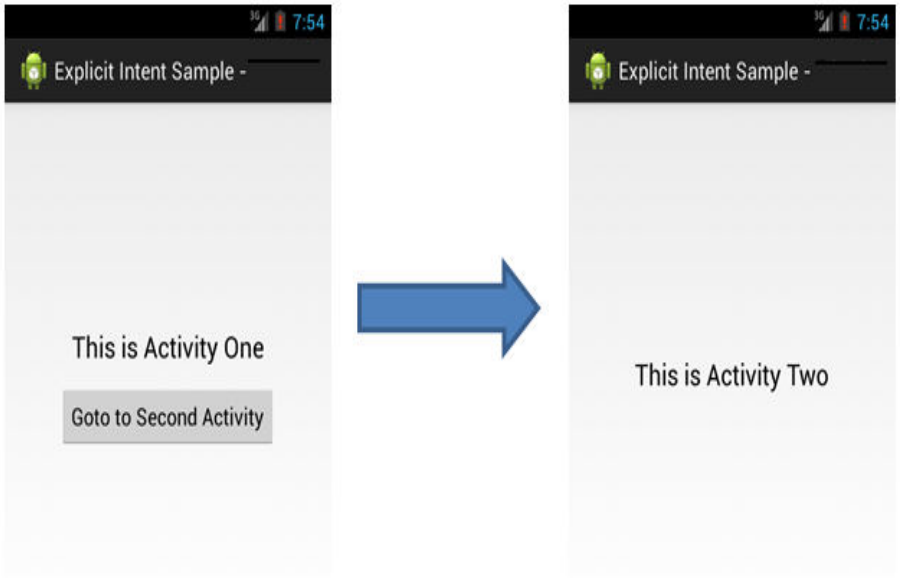
- **Intent Structure:**
  - **Intents are objects** of the **android.content.Intent** type.
  - The primary pieces of information in an intent are:
    - **Action** -- The general action to be performed, such as ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, etc.
    - **Data** -- The data to operate on, such as a person record in the contacts database, expressed as a Uri.

Types of intents

Android supports explicit and implicit intents.

- **Explicit Intents:** explicitly define the **component** which should be called by the Android system, by using the **Java class as identifier**.
- **Example:** The following creates an **explicit intent** and **send** it to the **Android system**. If the class specified in the intent represents an activity, the Android system starts it.
  - `Intent i = new Intent(this, ActivityTwo.class);`
  - `startActivity(i);`

Explicit intents are typically used within on application as the **classes** in an application are controlled by the application developer.



Types of intents

- **Implicit Intents:** specify the **action** which should be performed and optionally data which provides **content for the action**.

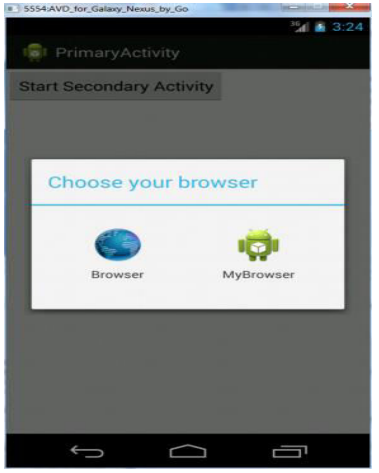
- **For example**, the following tells the **Android system** to view a webpage. All installed web browsers should be registered to the corresponding intent data via an intent filter.

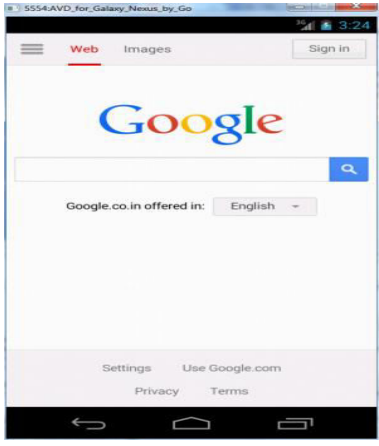
```
Intent i = new Intent(Intent.ACTION_VIEW,
Uri.parse("http://www.google.com"));
startActivity(i);
```

implicit intent : searches for all **components** which are registered for the specific action and the fitting data type.

- one component: Android starts this component directly.

- several components: the user will get a **selection dialog** and can decide which component should be used for the intent.





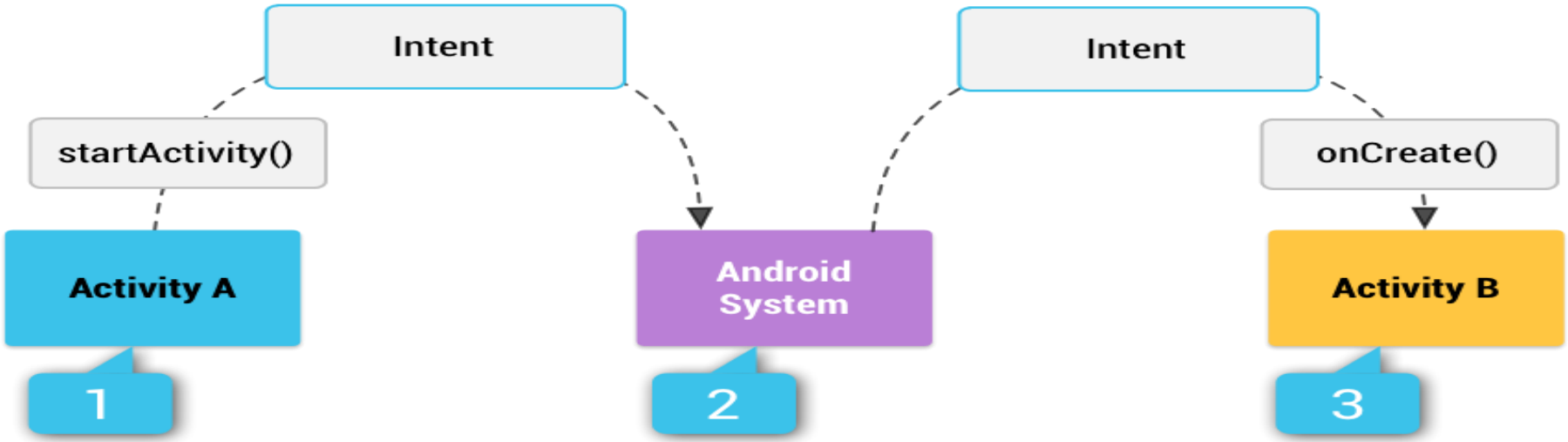
Intents and intent filter

- Illustration of how an **implicit intent** is delivered through the system to start another activity:

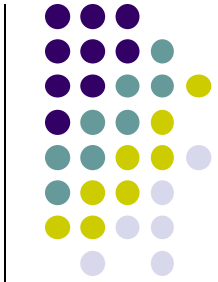
[1] ***Activity A*** creates an `Intent` with an **action description** and passes it to `startActivity().`

[2] The Android System searches all apps for **an intent filter** that matches the intent.

[3] When a match is found the system starts the matching activity (***Activity B***) by invoking its `onCreate()` method and passing it the **Intent**.



passing any parameters



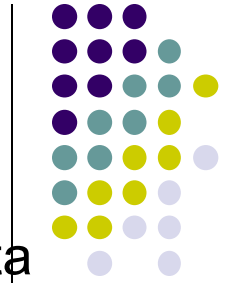
If you need to pass any parameters or data to the **second activity**, call `putExtra` on the **intent**.

- `Intent intent = new Intent(this,ActivityName.class);`
- `intent.putExtra("name1", value);`
- `intent.putExtra("name2", value);`
- `startActivity(intent);`

Example:

```
Intent i = new Intent(this, NewActivity.class);
i.putExtra("firstName", "Mike");
i.putExtra("lastName", "Jones");
startActivity(i);
```

Extracting extra data



In the **second activity** that was **invoked**, you can grab any extra data that was passed to it by the calling act.

– You can **access the Intent** that spawned you by calling `getIntent`.

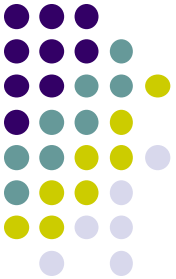
– The Intent has methods like `getExtra`, `getIntExtra`, `getStringExtra`, etc. to **extract** any data that was stored inside the intent.

- `Intent intent = getIntent();`
- `String extra = intent.getExtra("name1");`

Example:

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.view);
         Intent intent = getIntent();
    String fName = intent.getStringExtra("firstName");
    String lName = intent.getStringExtra("lastName");
}
```

Sending back a result



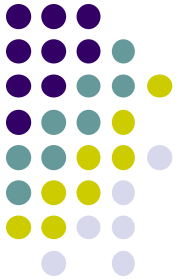
In the **second activity** that was invoked, **send data back:**

– Need to create an **Intent to go back**.

– Store any **extra data** in that intent; call `setResult` and `finish`.

```
public class SecondActivity extends Activity {
    ...
    public void onClick(View view) {
        Intent i = new Intent();
        String message = "abc";
        intent.putExtra("MESSAGE",message);
        setResult(2,i);
        finish(); // calls onDestroy
    }
}
```

Waiting for a result



If calling activity wants to wait for a result from called activity:

– Call `startActivityForResult` **rather than** `startActivity`.

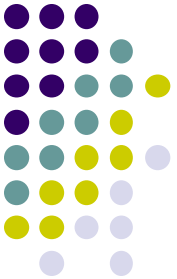
- `startActivityForResult` requires you to pass a **unique ID number** to represent the action being performed.

- **By convention**, you declare a **final int constant** with a value of your choice.
- The call to `startActivityForResult` will not wait; it will return immediately.

– Write an `onActivityResult` **method** that will be called when the second activity is done.

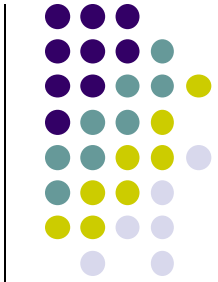
- **Check** for your **unique ID** as was passed to `startActivityForResult`.
- If you see your unique ID, you can **ask** the **intent** for any **extra data**.

In First Activity uses startActivityForResult



```
Intent i = new
Intent(MainActivity.this,SecondActivity.class);
    //suppose RequestCode == 2;  MUST be 0-65535
    // Call Back method  to get the Message form other Activity
    startActivityForResult(i,2);
@Override
protected void onActivityResult(int requestCode, int
resultCode, Intent i)  {
      super.onActivityResult(requestCode, resultCode, data);
          if(requestCode==2) {
             if(resultCode == Activity.RESULT_OK) {
                String message= i.getStringExtra("MESSAGE");
                   Toast.makeText(this,"Got back: " +message,
                                        Toast.LENGTH_SHORT).show();}
          }
   }
```

Implicit Intent (link)



implicit intent: One that launches **another app**, without naming that specific app, to handle a given type of **request or action**.

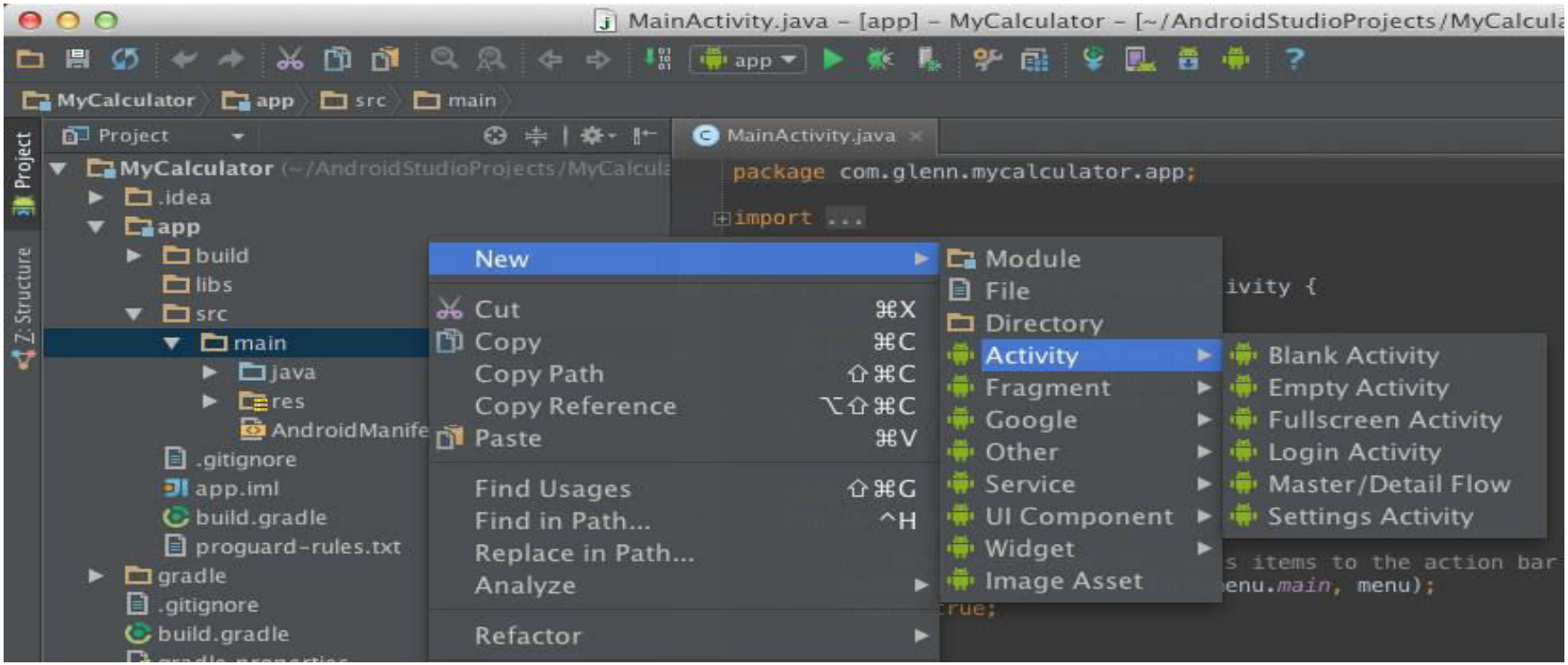
– **Examples:** invoke default browser; load music player to play a song

```
// make a phone call
    Uri number = Uri.parse("tel:5551234");
    Intent callIntent = new Intent(Intent.ACTION_DIAL, number);

// go to a web page in the default browser
    Uri webpage = Uri.parse("http://www.stanford.edu/");
    Intent webIntent = new Intent(Intent.ACTION_VIEW, webpage);

// open a map pointing at a given latitude/longitude (z=zoom)
    Uri location = Uri.parse("geo:37.422219,-122.08364?z=14");
    Intent mapIntent = new Intent(Intent.ACTION_VIEW, location);
```

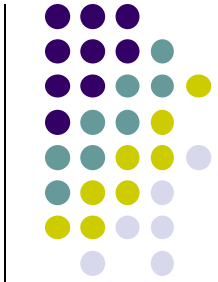
Adding an Activity

in Android Studio, right click **"app" at left: New -> Activity**

- creates a new **.XML file in res/layouts**
- creates a new **.java class in src/java**
- adds information to **AndroidManifest.xml about the activity**
- (without this information, the app will not start the activity)



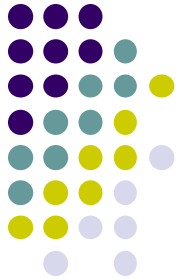
Activities in Manifest



- Every activity has an entry in project's AndroidManifest.xml, added automatically by Android Studio:

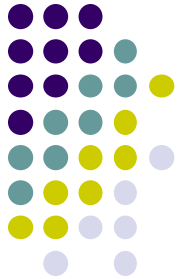
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.myusername.myapplication" >
<application android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
<activity android:name=".MainActivity"
    android:label="@string/app_name" >
    <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Activities in Manifest Continues



```
<activity android:name=".SecondActivity"
    android:label="@string/title_activity_second"
    android:parentActivityName=".MainActivity" >
    <meta-data android:name="android.support.PARENT_ACTIVITY"
    android:value="com.example.myusername.myapplication.MainActivity" />
</activity>
</application>
</manifest>
```

You can find more information about



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

- **App Manifest Overview**
- https://developer.android.com/guide/topics/manifest/manifest-intro