CSE 162 Mobile Computing Lecture 6: Intents and Fragments

Hua Huang

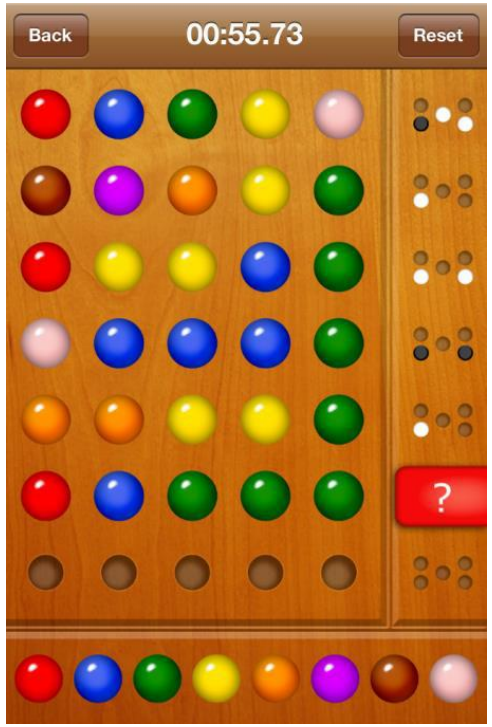
Saving State Data

Activity Destruction

- App may be destroyed
  - On its own by calling finish
  - If user presses **back button and quit it**
- Before Activity destroyed, system calls **onSaveInstanceState**
  - Can save state required to recreate Activity later
  - E.g. Save current positions of game pieces

onSaveInstanceState: Saving App State

- Systems write info about views to Bundle
- Programmer must save other app-specific information using **onSaveInstanceState( )**
  - E.g. board state in a board game such as mastermind



onRestoreInstanceState( ): Restoring State Data

- When an Activity recreated saved data sent to **onCreate**and **onRestoreInstanceState()**
- Can use either method to restore app state data

Saving Data Across Device Rotation

- override **onSaveInstanceState**method

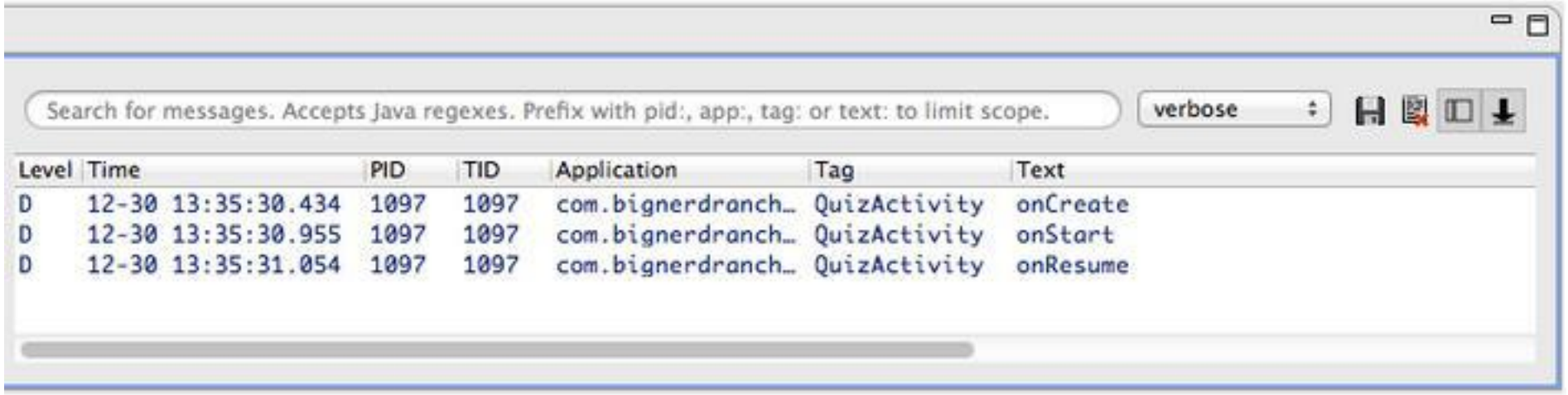
```
private static final String KEY_INDEX = "index";
```

```
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    super.onSaveInstanceState(savedInstanceState);
    Log.i(TAG, "onSaveInstanceState");
    savedInstanceState.putInt(KEY_INDEX, mCurrentIndex);
}
```

Logging Errors in Android

Logging Errors in Android

- Android can log and display various types of errors/warnings in Android Studio Window



- Error logging is in **Log** class of **android.util** package, so need to

  import android.util.Log;

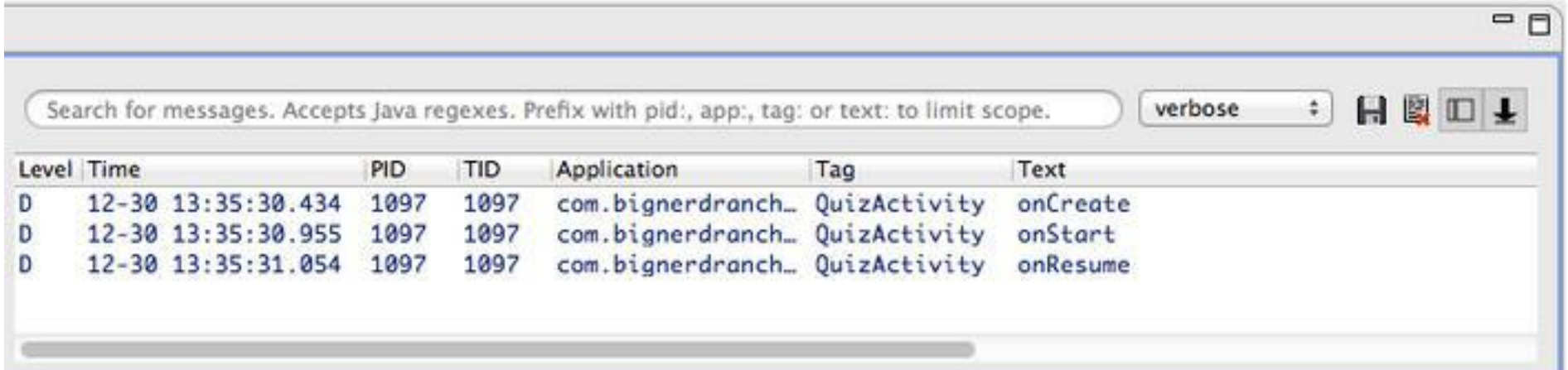
- Turn on logging of different message types by calling appropriate method

Method	Purpose
`Log.e()`	Log errors
`Log.w()`	Log warnings
`Log.i()`	Log informational messages
`Log.d()`	Log debug messages
`Log.v()`	Log verbose messages

Example

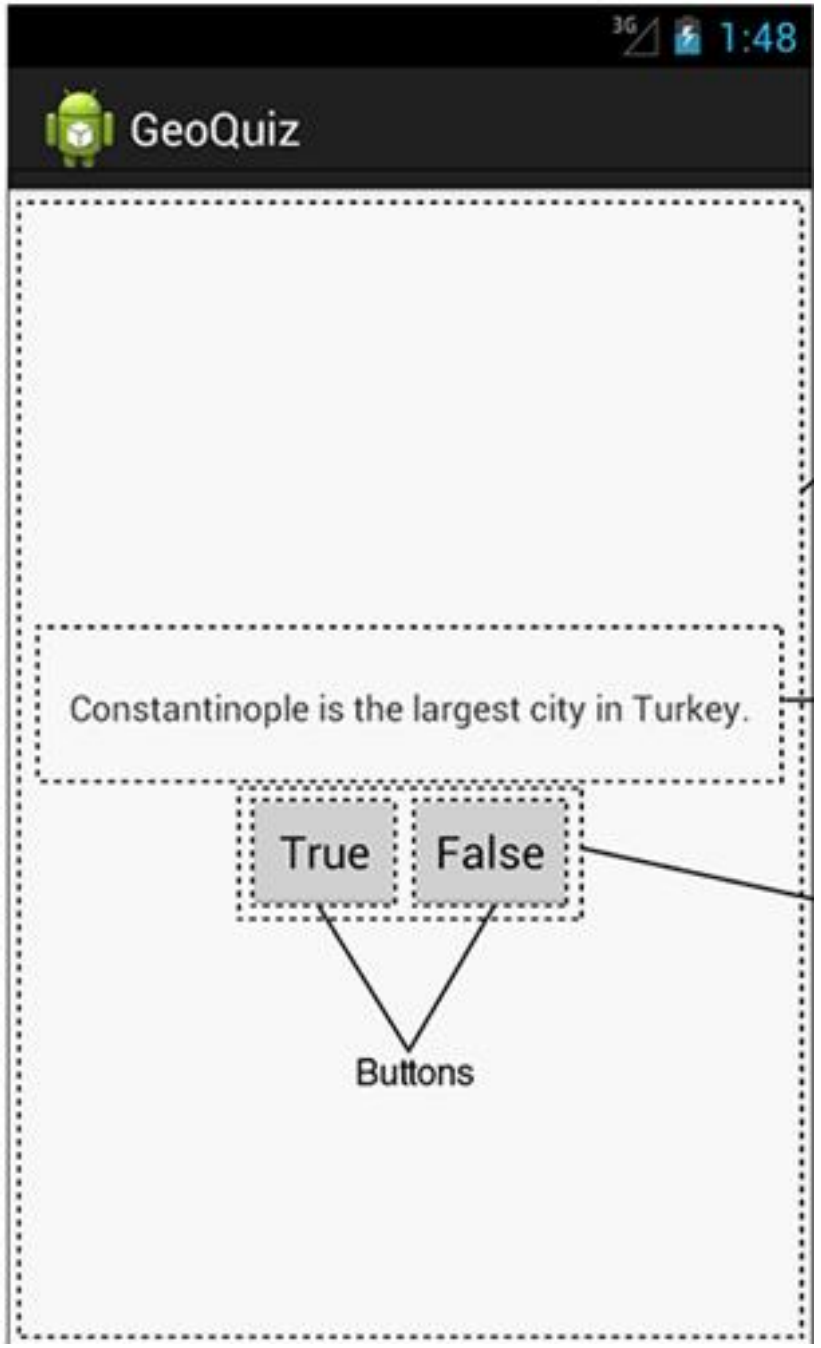
- A good way to understand Android lifecycle methods is to print debug messages in Android Studio when they are called

```
onCreate( ){
   ... print message “OnCreate called”...

}

onStart( ){
   ... print message “OnStart called”...
```



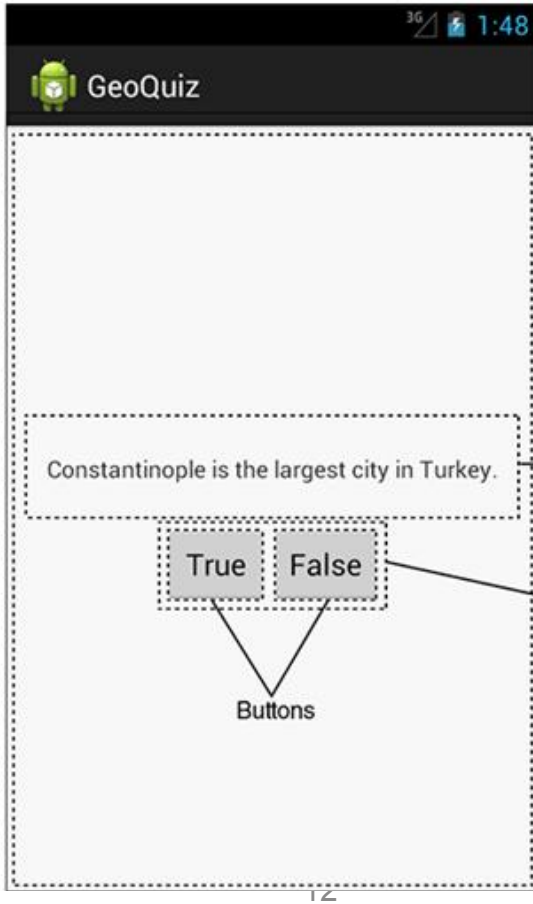
- Example: print debug message from onCreate method below

```
package com.bignerdranch.android.geoquiz;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;

public class QuizActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_quiz);
    }
```



- Add the debug message

```
public class QuizActivity extends Activity {

    ...

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d(TAG, "onCreate(Bundle) called");
        setContentView(R.layout.activity_quiz);

        ...
```



- Debug (d) messages have the form

```
public static int d(String tag, String msg)
```

- E.g.

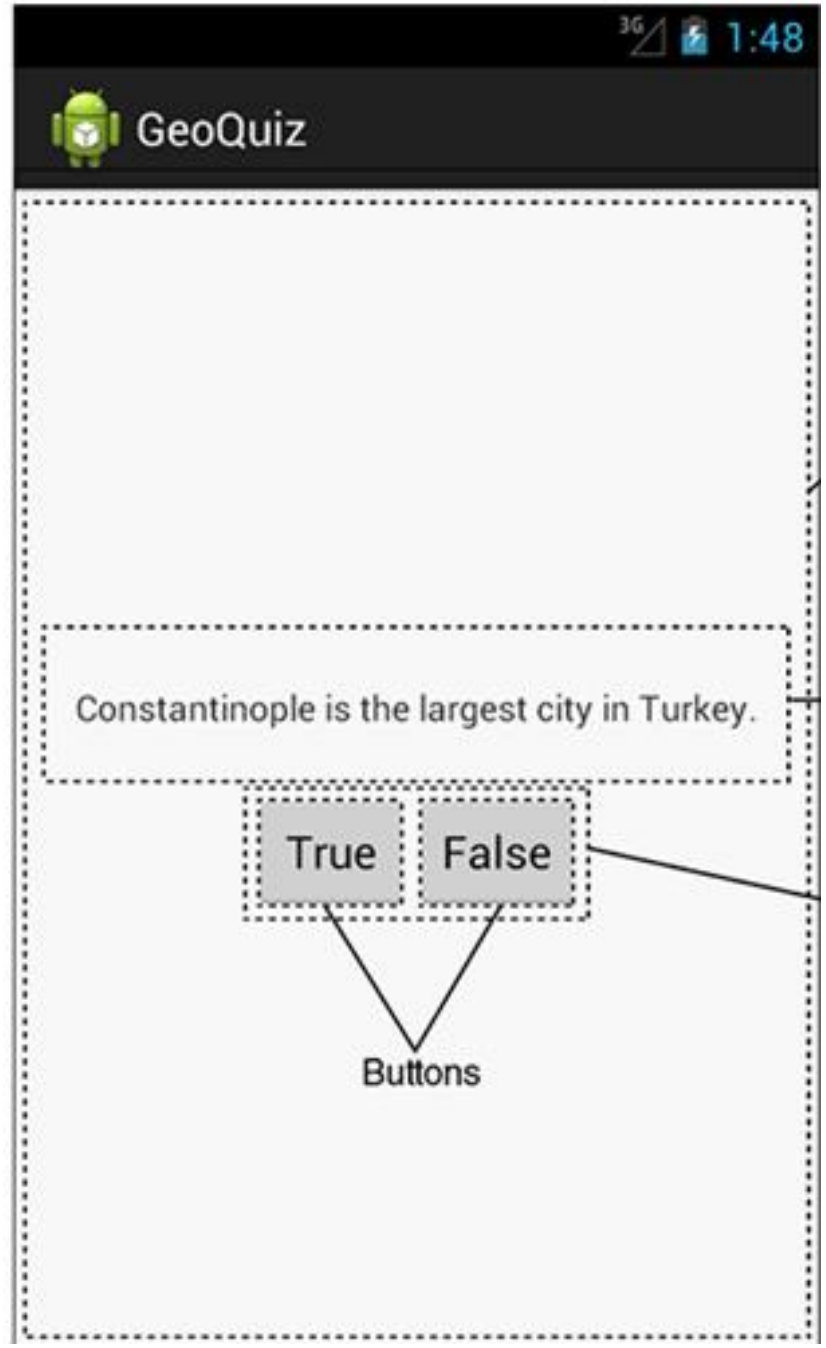
Tag → QuizActivity: Message → onCreate(Bundle) called

- Example declaration:

```
Log.d(TAG, "onCreate(Bundle) called");
```

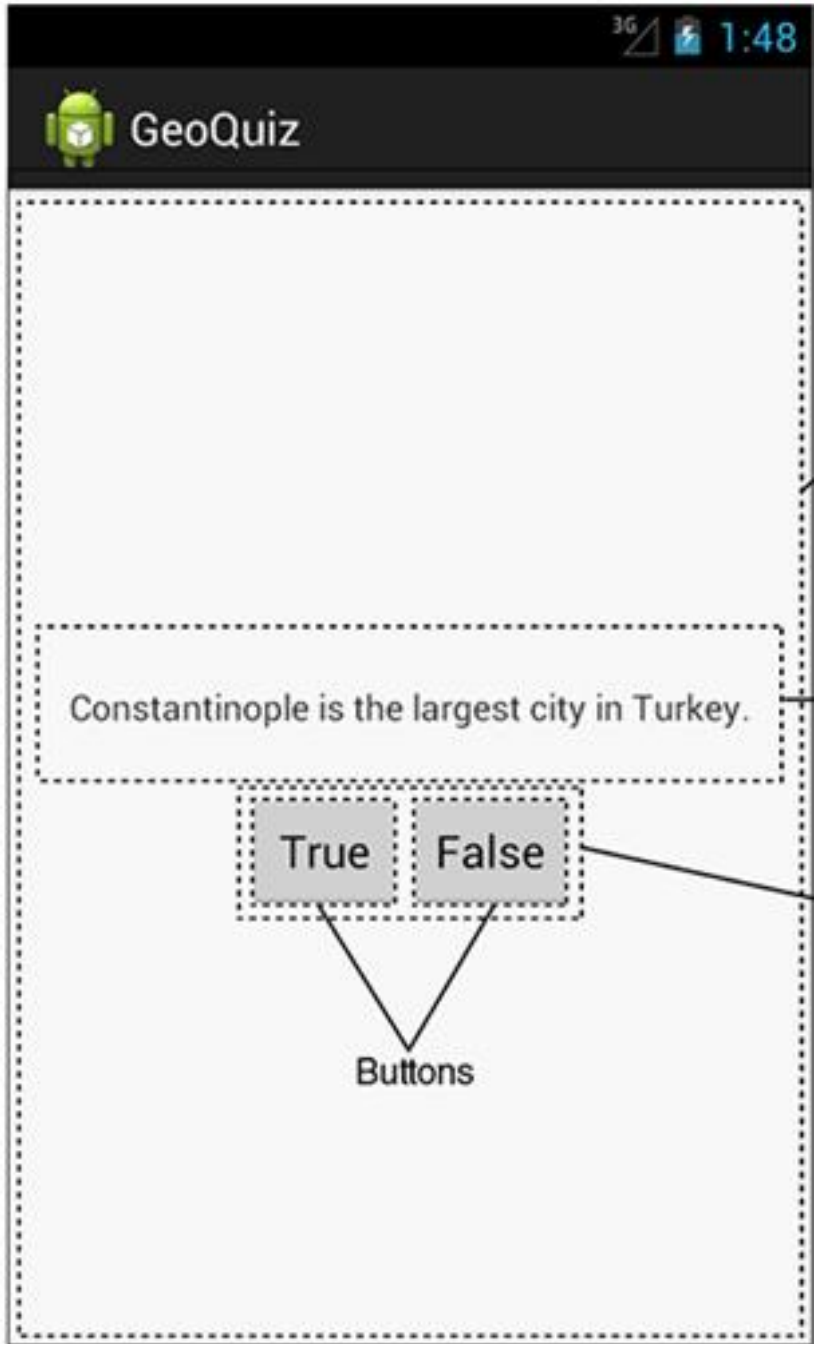
- Then declare string for **TAG**

```
public class QuizActivity extends Activity {

    private static final String TAG = "QuizActivity";

    ...

}
```

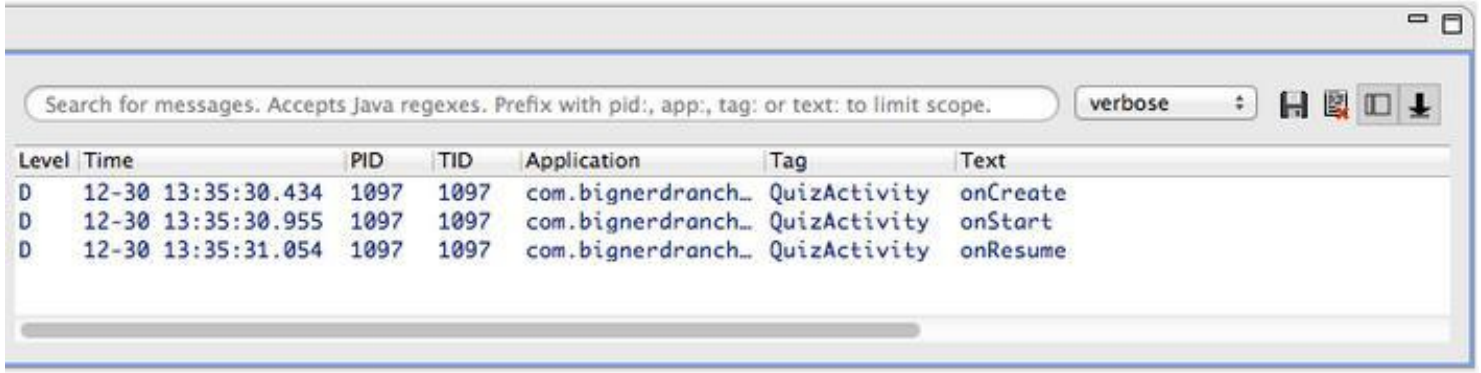


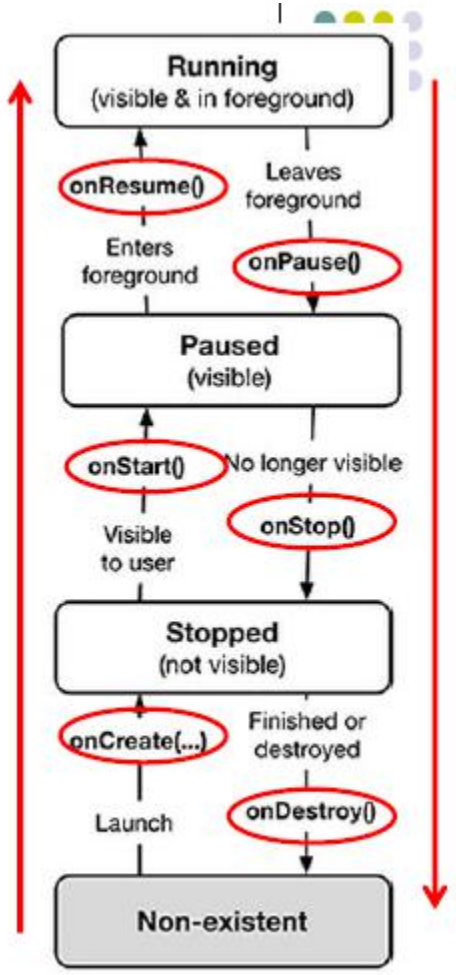
Print debug messages from each method

```
} // End of onCreate(Bundle)

@Override
public void onStart() {
    super.onStart();
    Log.d(TAG, "onStart() called");
}

@Override
public void onPause() {
    super.onPause();
    Log.d(TAG, "onPause() called");
}

@Override
public void onResume() {
    super.onResume();
    Log.d(TAG, "onResume() called");
}

@Override
public void onStop() {
    super.onStop();
    Log.d(TAG, "onStop() called");
}

@Override
public void onDestroy() {
    super.onDestroy();
    Log.d(TAG, "onDestroy() called");
}
```



- Homework: Run the program, make sure you see all the debug messages





Intents

What is an Intent?

- Intent is an intention to do something
- Intent contains an action carrying some information
- Intent is used to communicate between android components
  - Start an activity
  - Start a service
  - Deliver a broadcast

Example

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

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

Why intent?

- Intent is used to communicate, share data between components
- Intent contains the following things
  - Component name
  - Action
  - Data
  - Extras
  - Category
  - flags

Intent Types

- Explicit and Implicit Intents

Explicit Intents

- **Explicit Intent:** If components sending and receiving Intent are in same app
  - E.g. Activity A starts Activity B in same app
  - Activity A explicitly says what Activity (B) should be started
- It’s faster
- Used if you know the specific activity to perform

Explicit Intent Example

```
// 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);
```

Implicit Intents

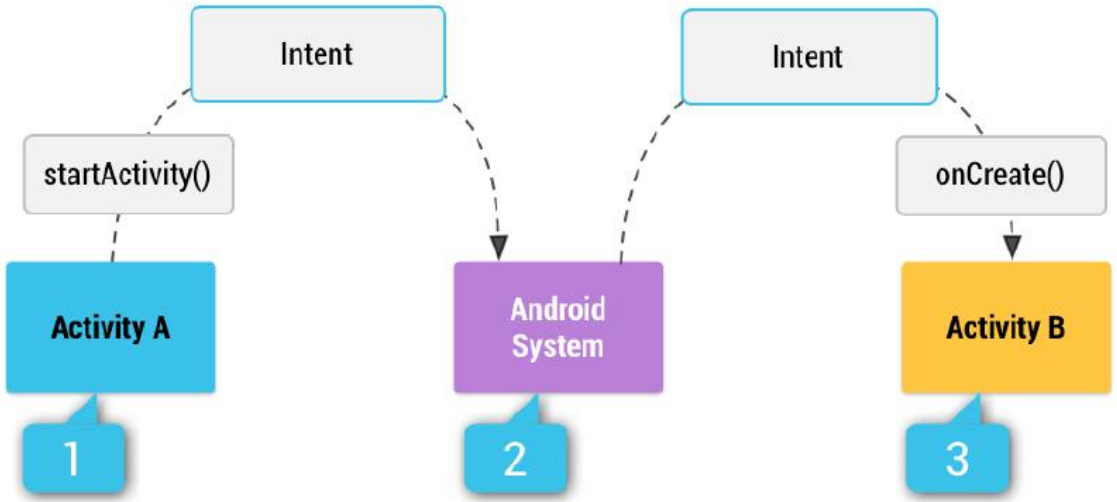
- **Implicit Intent:** If components sending and receiving Intent are in **different apps**
  - Activity B specifies what ACTION it needs done, doesn't specify Activity to do it
  - Example of Action: take a picture, any camera app can handle this
- Useful when your app cannot perform the action
  - But other apps can do it
- Exception Handling: it is possible there isn't any app that handles your implicit intent

Example

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

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

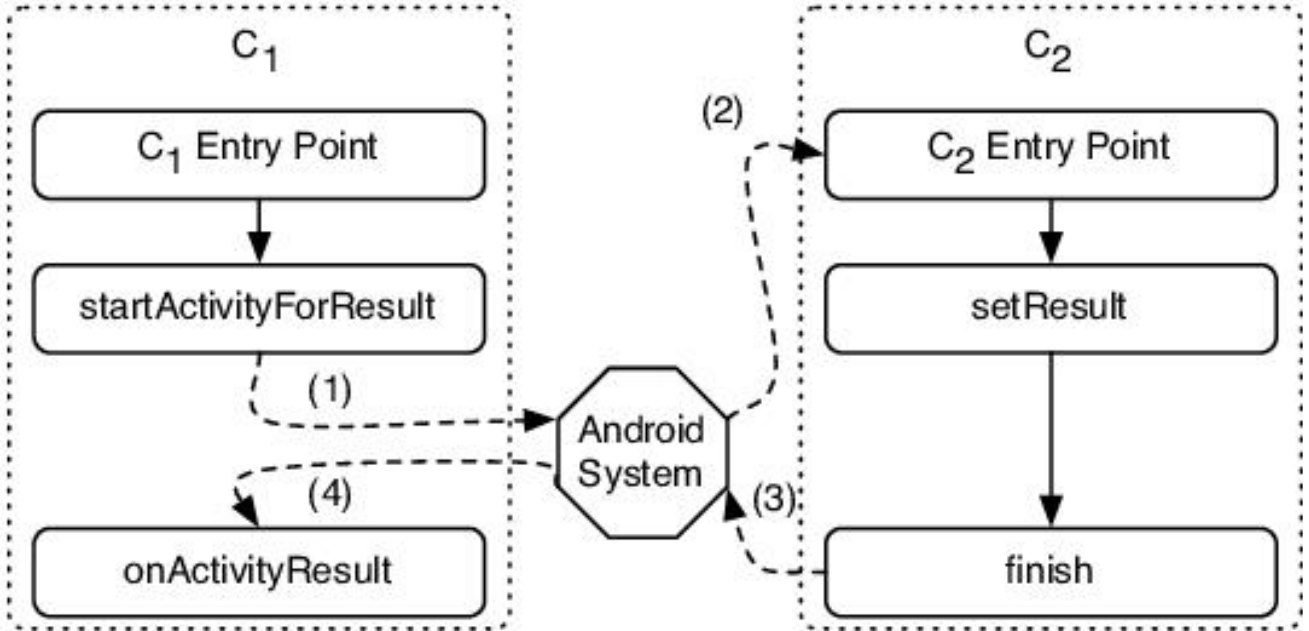
Intent Uses

- 3 main use cases for Intents
- **Case 1 (Activity A starts Activity B, no result back):**
  - Call **startActivity( )**, pass an Intent
  - Intent has information about Activity to start, plus any necessary data



Intent: Result Received Back

- **Case 2 (Activity A starts Activity B, gets result back):**
  - Call **startActivityForResult( )**, pass an Intent
  - Separate Intent received in Activity A's **onActivityResult( )** callback



Intent: Result Received Back

- **Case 3 (Activity A starts a Service):**
  - E.g. Activity A starts service to download big file in the background
  - Activity A calls **StartService( )**, passes an Intent
  - Intent contains information about Service to start, plus any necessary data

Explicit Activity launch

Manifest Activity Registration

```
<activity android:name= ".ViewActivity"  android:label="View Tests">
    <intent-filter><action android:name  =
    "com.acorns.intent.action.ShowView"/>
                <category android:name ="android.intent.category.DEFAULT" />
    </intent-filter></activity>
```

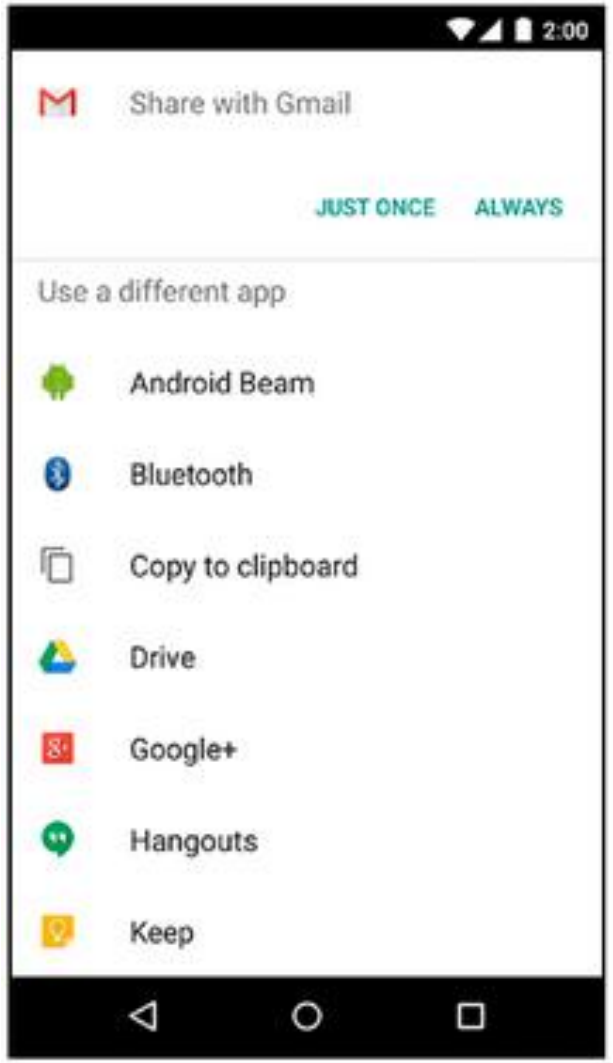
Default category
Needed to allow implicit intents

Basic Java Code

```
public class ViewActivity extends Activity
{  @Override  public void onCreate(Bundle state)
   {        super.onCreate(state);
            setContentView(R.layout.main);
            Intent  intent = this.getIntent();
            if (intent==null) {    log.d("ViewActivity","invoked without an intent");   }
}    }
```

Launch

```
ctivity)
 {    parentActivity.startActivity(new
 Intent("com.acorns.intent.action.ShowView"));   }
```




```
mReportButton.setOnClickLisener
    (  new View.OnClickListener() {
            public void onClick(View v) {
            Intent I = new
               Intent(Intent.ACTIONS_SEND);
            i.setType("text/plain");
            i.putExtra.EXTRA_TEXT,
                                    getCrimeReport());
            i.putExtra(Intent.EXTRA_SUBJECT,
                        getString(

r.string.crime_report_subject));
            startActivity(i);
     });
}
```

Launching the Implicit Intent

Intents with Results

```
// Launch the sub activity expecting a result
private static final int SELECT_HORSE = 1,  SELECT_GUN = 2;
private void startSubActivity(int option)
{    startActivityForResult(new Intent(this, Other.class),  option);  }

// Receive Result with a call back
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data)
{    super.onActivityResult(requestCode, resultCode, data);
     if (resultCode == Activity.RESULT_OK)
     {    switch(requestCode)
          {    case (SELECT_HORSE):   selectedHorse = data.getData();  break;
               case (SELECT_GUN):       selectedGun = data.getData();     break;
}    }    }
```

How Intents are received

- Receiving Implicit Intents
- Receiving Explicit Intents

Implicit Intent Reception

- In your manifest.xml file, declare what intents your app can handle
  - Use <intent-filter> tag
- You can mention three elements in <intent-filter>
  - Action
  - Data
  - category

Implicit Intent Reception

- **Action:** Action to be performed must match a registered a manifest intent filter
- **Data:** Detailed specifications of data that must match in intent filter
  - **Host**: for example, google.com
  - **Mime type**: vnd.android.cursor.dir/ or vnd.android.cursor.item/ matches all or specific rows of an android SQL cursor
  - **Path, Path Pattern, Path Prefix:** match if specified (ex: /data authority/note)
  - **Data authority:** match filter specification (ex: com.google.provider.NotePad)
  - **Port**: listening port on the host machine
  - **Scheme**: examples: content or http

Implicit Intent Reception

Intent Categories: Indicate to Android the general nature of an activity

- CATEGORY_DEFAULT: Can invoke through implicit intents
- CATEGORY_BROWSABLE: Will not violate browser security restructions
- CATEGORY_TAB: Embeddable in a tabbed view of a parent activity
- CATEGORY_ALTERNATIVE: Alternative action to displayed data
- CATEGORY_SELECTED_ALTERNATIVE: Alternative action to selected data
- CATEGORY_LAUNCHER: included on launcher screen lists
- CATEGORY_HOME: The home screen view (One per device)
- CATEGORY_PREFERENCE: Shown on the preference screen