Android

Operating System and Architecture

Android

- Operating System and development system from Google and Open Handset Alliance since 2008
  - At the lower level is based on the Linux kernel and in a higher level is based on a Java virtual machine
  - Several versions in quick succession

    - 1.0, 1.1, 1.5 (cupcake), 1.6 (donut), 2.0, 2.1 (éclair), 2.2 (froyo), 2.3 (gingerbread), 3.0-2 (honeycomb) (tablets), 4.0 (ice cream sandwich), 4.1-3 (jelly beans), 4.4 (kitkat), 5.0-1 (lollipop), 6.0 (marshmallow), 7.0-1 (nougat), 8.0-1 (oreo), 9.0 (pie), 10.0 (Q), 11.0 (R)
  - Supports a high hardware variability
  - Integrates a sensor collection (gps, accelerometer, compass, gyroscope …)
  - High graphics and sound quality
    - Screen densities of 120, 160, 240, 320 and 480 dpi and higher
    - Resolutions from 240x320 to 1600x2520 pixels (4K 2160x3840)

Screens

	Low	Med	TV	High	Xhigh	XXhigh
Small	0.1%				0.1%	
Normal		0.4%	0.3%	17.0%	41.1%	25.9%
Large		1.8%	2.0%	0.7%	2.6%	2.1%
Xlarge		3.5%		1.9%	0.5%	
	120 dpi	**160 dpi**	**213 dpi**	**240 dpi**	**320 dpi**	**480 dpi**

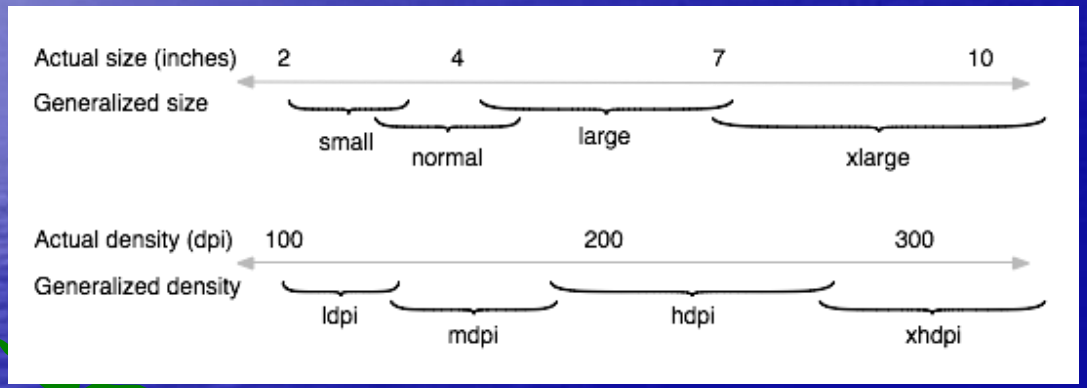
Aug. 2020

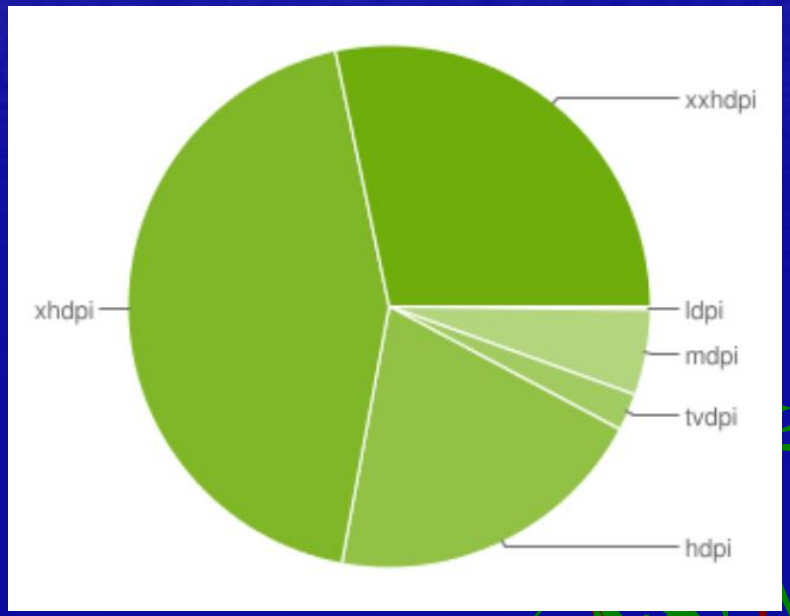
< 3.0”

from 3.0” to 5.5”

from 4.5” to 7.0”

from 7.0” to 10.0”





Actual screen resolutions:
from 240x320 pixels
to (4K = 2160x3840)

Main features

- Framework based in reuse and extension patterns
- Optimized virtual machine (Dalvik VM) → ART
- Integrates a browser (based on WebKit) → Blink
- 2D and 3D graphics (OpenGL ES 1, 2, 3) → 3.2
- Local relational data base based on SQLite
- Several standard multimedia formats supported (MPEG4, H.264, MP3, AAC, JPG, PNG, GIF, …)
- Comms in GSM, 3/4/5G, WiFi, Bluetooth and NFC
- Camera, GPS, compass and accelerometer
- Application development uses Java or Kotlin

Versions in use

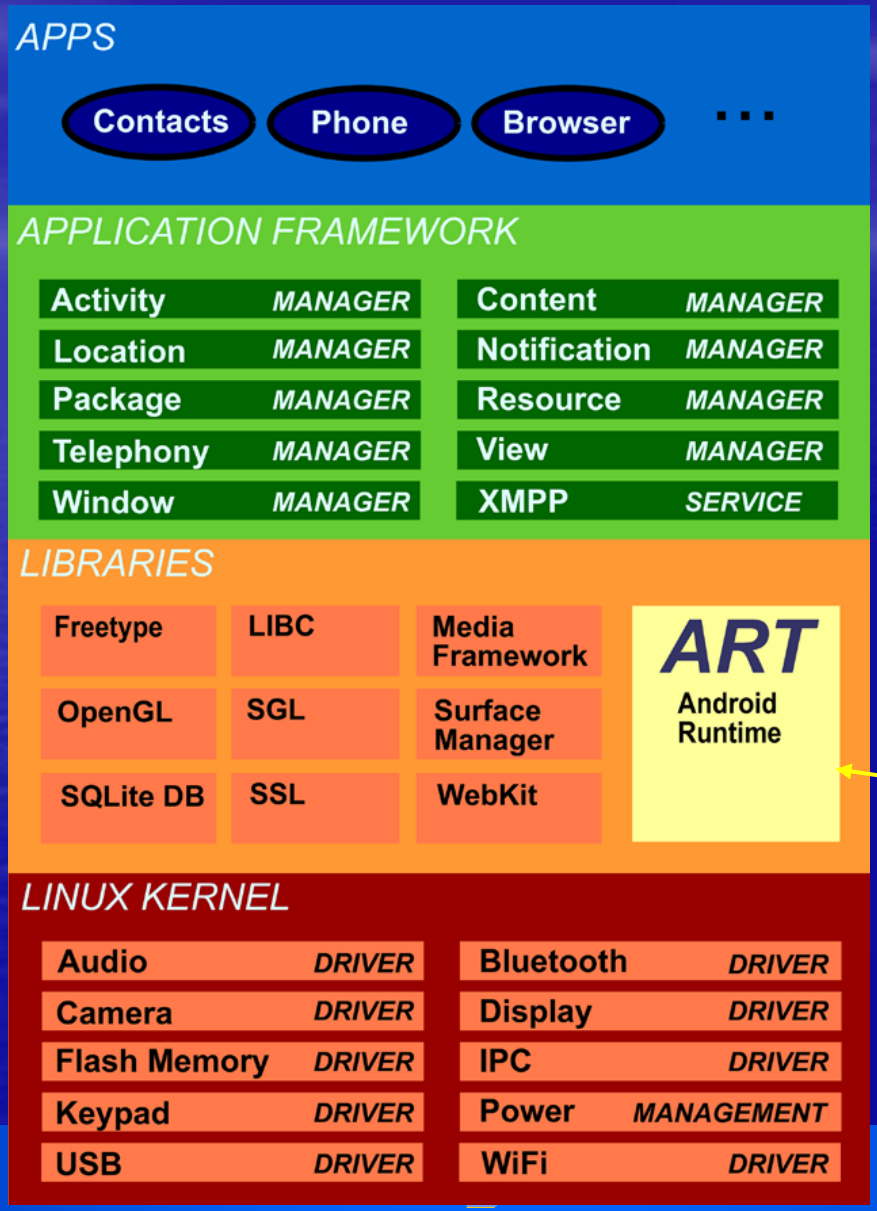
ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99,8%
4.2 Jelly Bean	17	99,2%
4.3 Jelly Bean	18	98,4%
4.4 KitKat	19	98,1%
5.0 Lollipop	21	94,1%
5.1 Lollipop	22	92,3%
6.0 Marshmallow	23	84,9%
7.0 Nougat	24	73,7%
7.1 Nougat	25	66,2%
8.0 Oreo	26	60,8%
8.1 Oreo	27	53,5%
9.0 Pie	28	39,5%
10. Android 10	29	8,2%

Sep-2020

Platform	API Level	Distribution
Android 2.3.3-7	10	0.1%
Android 4.0-1	15-18	1.8%
Android 4.4	19	4.0%
Android 5.0-1	21-22	9.2%
Android 6.0	23	11.2%
Android 7.0-1	24-25	12.9%
Android 8.0-1	26-27	21.3%
Android 9.0	28	31.3%
Android 10.0	29	8.2%

Software Architecture

Android OS Layers and Components



Java code

C/C++ native compiled code

Java virtual machine
Java libraries
JIT and AOT compilers

OS

Operating system components (1)

- **Linux kernel**
  - Low level OS services (memory and process management, communications and network, files) and hardware access (peripheral and sensor drivers)
- **Native libraries**
  - Written in C/C++ and compiled to the processor native instructions
    - Surface management, 2D and 3D graphics, multimedia codecs, DBMS SQL, Web engine
    - Wrapped in Java
    - It is possible to develop and install new native libraries using the NDK (Native Development Kit)
- **Android runtime**
  - Java virtual machine optimized for small devices and processors (Dalvik VM)
  - Java base library (with some superposition with Java SE and Java ME)

Operating system components (2)

- **Application Framework**
  - **High level library, in Java, suitable to the creation of user Android applications (Android API); also the higher-level Android management services (Java)**
    - **Activity manager – controls the application lifetime and navigation between 'screens'**
    - **Content manager – control shared data between applications providing a standard format and access**
    - **Resource manager – Management of non-code specifications and assets in applications**
    - **Location manager – Android device position determination (the device knows always its position in the world using GPS Wi-Fi or GSM receptors)**
    - **Notification manager – External event management like messages, to-do's, alerts, etc**

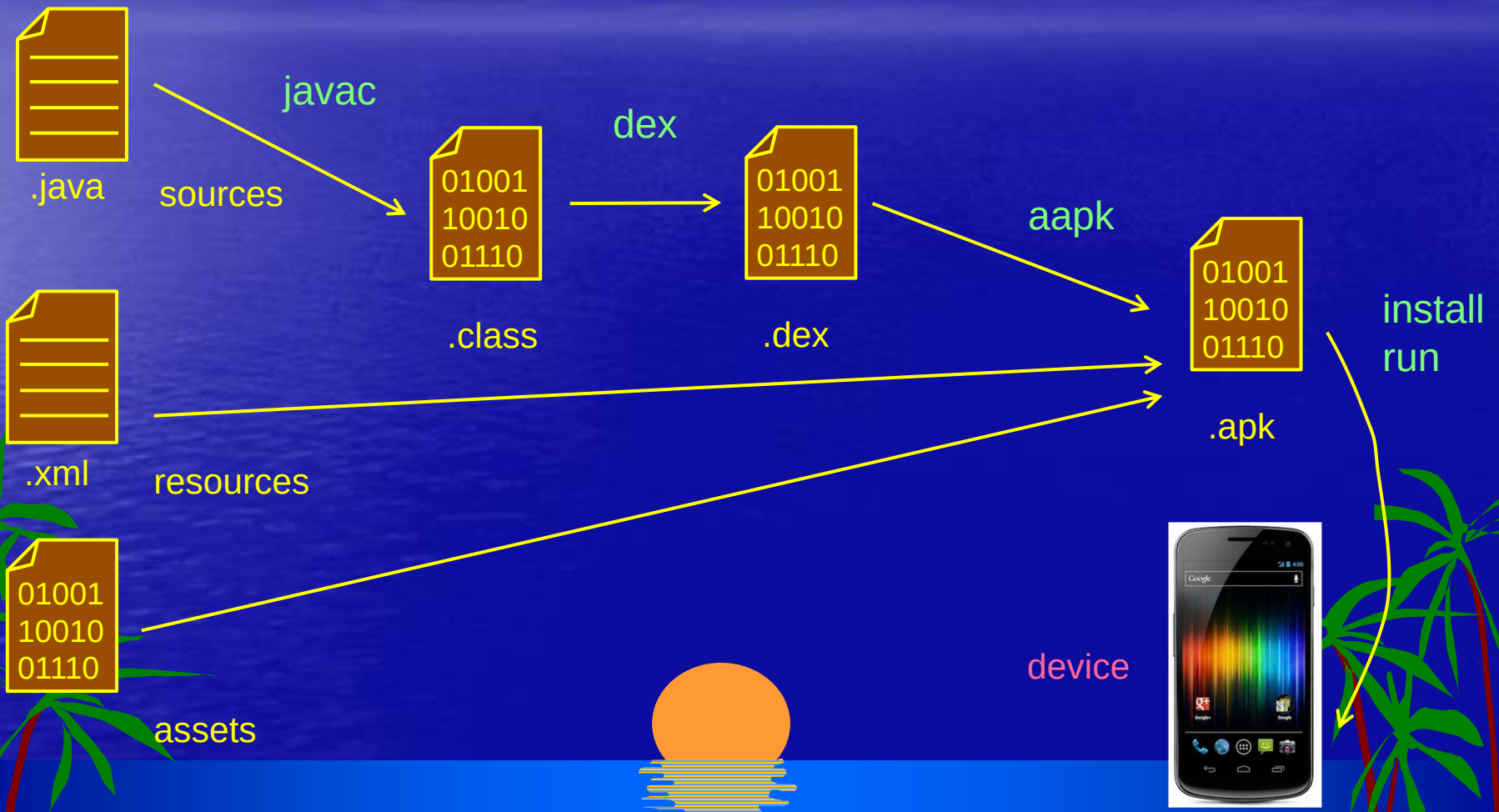
The Android system

- **Applications (Apps)**
  - Programs that control the full screen to interact with the user
    - An Android device contains some pre-defined applications which are mandatory:
      - Home
      - Launcher
      - Phone dialer
      - Calendar and Email
      - Contacts
      - Web browser
      - Play Store

- **Widgets**
  - Operate only on a small rectangular portion of the screen, inside the Home application

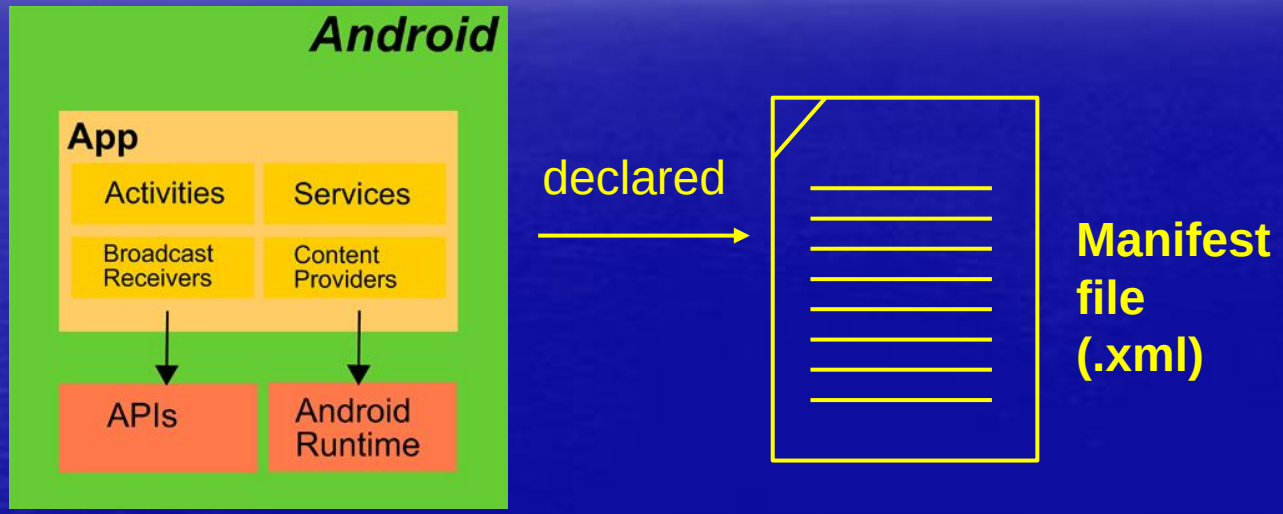
Building Applications

- **Built from Java source code, Resources and Assets**



Application components (1)

- The Android applications can contain several independent components
  - Activities, Services, Broadcast Receivers and Content Providers

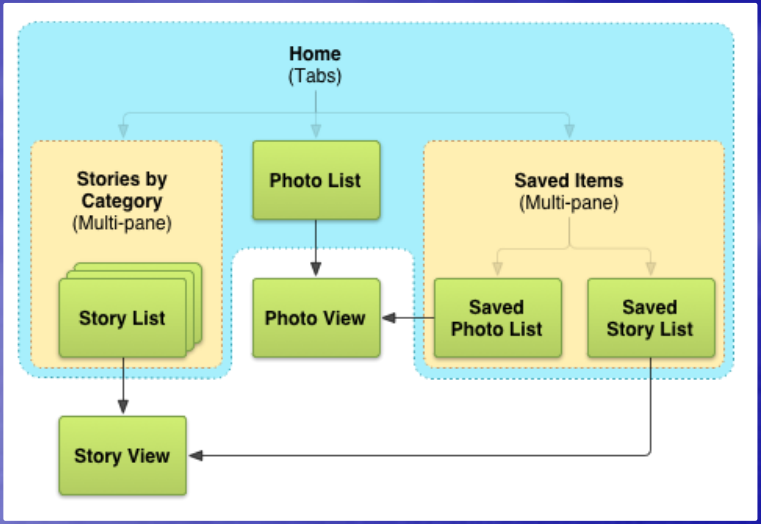


- Components
  - Activities
    - Contain a portion of the user interface (full screen or window)
    - Execute a well-defined task inside the application
    - One application can contain one or more activities
    - Are independent but can be invoked by others
    - Are subclasses of the android.app.Activity class
    - Are usually composed by a hierarchy of Views
    - One activity must be the starting activity of an application

Application components (2)

- Other components
  - Services
    - Don't have user interface
    - Can execute in background for an indeterminate period
    - It's possible to establish a connection with the service and communicate through a well-defined programming interface
  - Broadcast receivers
    - Can receive and react to notifications originated in the system or other applications
    - Applications can initiate a notification 'broadcast'
  - Content providers
    - Make available to other applications a data collection maintained by this application
    - Define an interface to access, add and update the supported data types

Application planning



Activity Diagram

Navigation

Sketch and
Wireframe

Activity Launching

action.MAIN
category.LAUNCHER

Intent

Activity

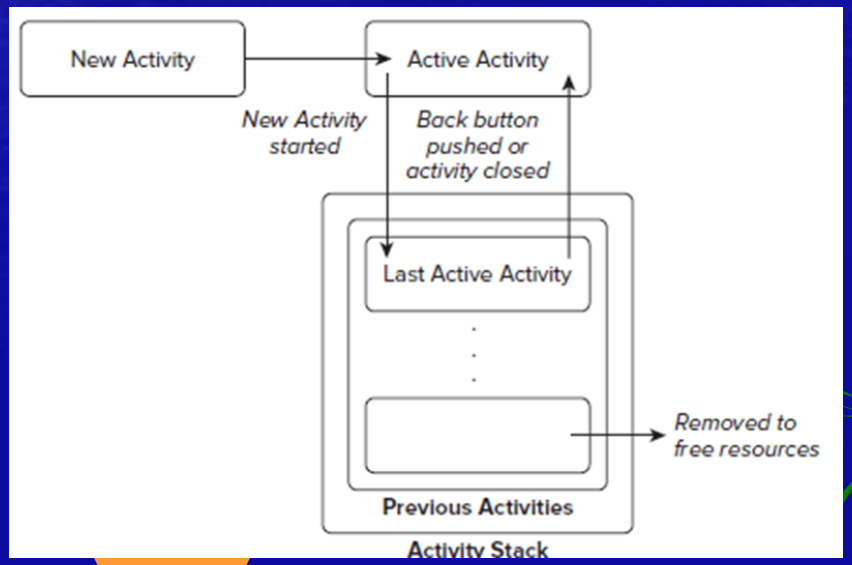
Android Launcher

startActivity()

Intent

Other Activity

Back button



Component activation (1)

- **Activities and Services**
  - Are activated through an *Intent*
  - Intents identify the component or specify actions for activities and services
    - Explicit intents contain the class name of the destination
    - Implicit intents can specify an *action*, a *category*, *data* (in the form of an uri) and possibly *extra* information
      - Android will try to find a matching component capable of executing the action in the data (or data type) specified
    - When activities and services are declared in the manifest they can specify '*intent-filters*' describing their accepting intents

Inside an activity, when we need to navigate to other activity of the same application, we normally use an *explicit* intent:

```
Intent otherActivity = new Intent(thisActivity, OtherActivity.class);
thisActivity.startActivity(otherActivity);
```

Component activation (2)

- **Broadcast receivers**
  - **Intents for activating broadcast receivers identify a 'message' to be delivered to matching receivers**
    - The 'message' is specified using an action, category, data and extra info (put together in an intent object)
    - It is sent by a call to `sendBroadcast(Intent)`
    - A broadcast receiver that has been installed matching the intent (with a compatible intent-filter) will be then activated (runs its `onReceive(Context, Intent)` method)

Component activation (3)

- Content providers
  - When declared in the manifest they must have an ‘authority’ (which is a kind of provider name)
  - Also they must recognize a name for its data collection
  - Usually they support CRUD operations on that collection
  - They are activated through a ContentResolver object
    - Obtained by getContentResolver() method from an activity
    - ContentResolvers have operations (methods) like: query, insert, update and delete
    - These methods require a Uri identifying the provider and data collection following the format
      - content://<authority>/<data collection>[/<item>]

Intents

- **Component invocation mechanism**
  - **For Activities (UI), Services (background), Broadcast Receivers (notifications)**
  - **All the *intents* have a name (action) and can have more data associated (uri, category, extra info)**
    - **They can be also *explicit* with a class name (inside an app)**
  - **An *intent* is a class in the Android API**
  - **There are many pre-defined *intents* in the Android API**

Example 1:

```
Intent intent = new Intent(Intent.ACTION_DIAL);
startActivity(intent);
```

Matches the Android activity (in the Dialer application) that allows the user to make a phone call, declared in an <intent-filter>, that can handle this action

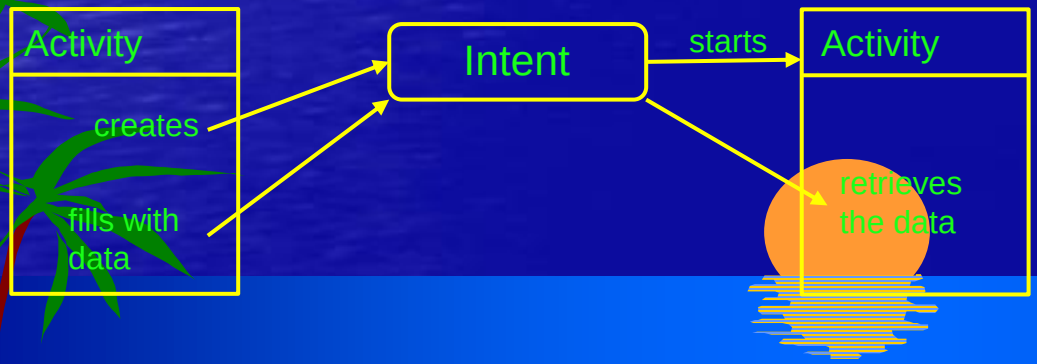
Example 2:

```
Intent intent = new Intent(Intent.ACTION_CALL);
intent.setData(Uri.parse("tel:555-555-5555"));
startActivity(intent);
```

Matches the activity that makes a phone call from a number

Intents data

- Intents can transport data between components
  - The Data field can be used for any type of an Uri
    - The calling component uses the method setData(Uri data), while the new component can read it with Uri getData()
  - The Extra field is used for arbitrary data types
    - It is a Bundle – set of (name, value) pairs organized as an hash table
      - The value can be a String, simple type, or an array
      - Can also be any Serializable or Parcelable (more efficient) object
      - The values are inserted with some PutExtra(String name, ... value) method and retrieved with ... get...Extra(String name)

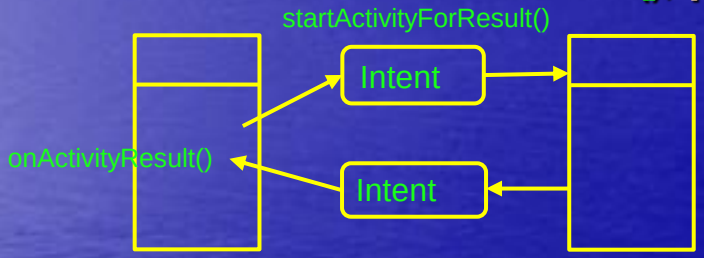



```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
     ...
    Intent intent = getIntent();

    String fName = intent.getStringExtra("firstName");
    String lName = intent.getStringExtra("lastName");
}
```

Getting result data from other Activities

- **Specially invoked activities can return data**
  - **Invocation using startActivityForResult(...)**
    - **Besides the intent, it has a requestCode (int) as a parameter**
      - **The new activity should create a result Intent, fill it with the result data, and call setResult(), passing this intent, before finishing**




```
...
Intent resultIntent = new Intent();
resultIntent.putExtra("some_key", "String data");
setResult(Activity.RESULT_OK, resultIntent);
finish();
```

      - **The original activity can retrieve the result intent and get the data, in the callback method onActivityResult()**

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
  super.onActivityResult(requestCode, resultCode, data);
  switch(requestCode) {
    case (MY_CHILD_ACTIVITY) : {
      if (resultCode == Activity.RESULT_OK)
          String returnValue = data.getStringExtra("some_key");
      break;
    }
  }
}
```