Understanding Heart and Body Status from a Smartphone

– Sensor Sensitive Programing on Android–

Guillaume Lopez
Living Environment Laboratory
The University of Tokyo, School of Engineering

Smartphone Sensor Web #3

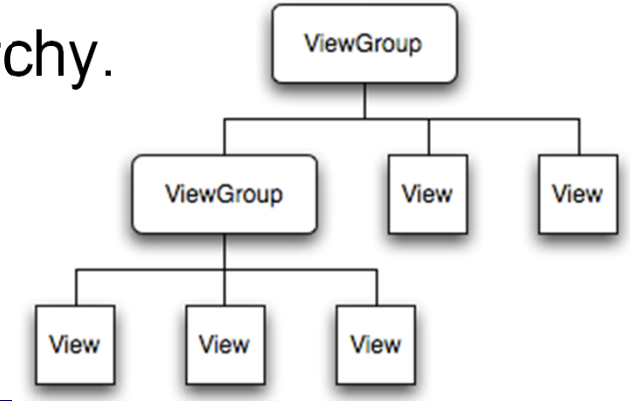
「Table of Contents」

Activity

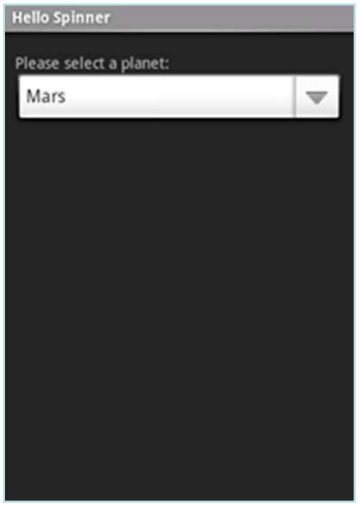
- Displays a **user interface** component and responds to system/user.
- When an application has a user interface, it contains one or more “Activities”.
- An existing “Activity” *can be replaced* with a new one that fulfill the same contract (intent).
- Each “Activity” *can be invoked* from other applications.
- Adding a new “Activity” in an Android project
  - The new Java class must extend the framework “Activity” class.
  - Created “Activity” must be defined into the application’s Manifest.xml.

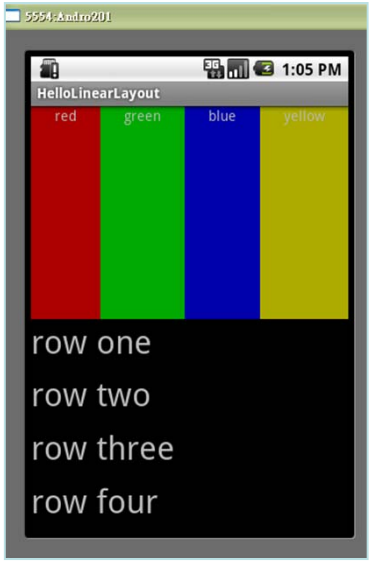
Activity (cont'd)

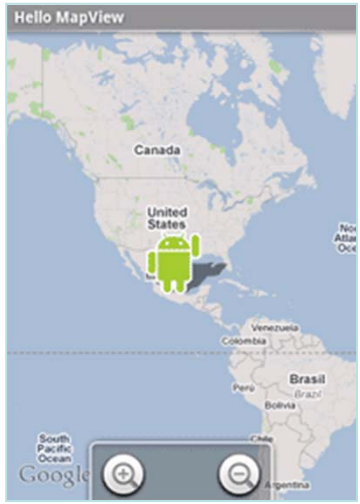
- Each activity is given a default window to draw in. Typically, the window fills the screen, but it might be smaller than the screen and float on top of other windows.
- The visual content of the window is provided by a hierarchy of views — objects derived from the base View class.
  - Android has a number of ready-made views that you can use — including buttons, text fields, scroll bars, menu items, check boxes, and more.
- A view hierarchy is placed within an activity's window by the ***Activity.setContentView***() method. The content view is the View object at the root of the hierarchy.



User Interface

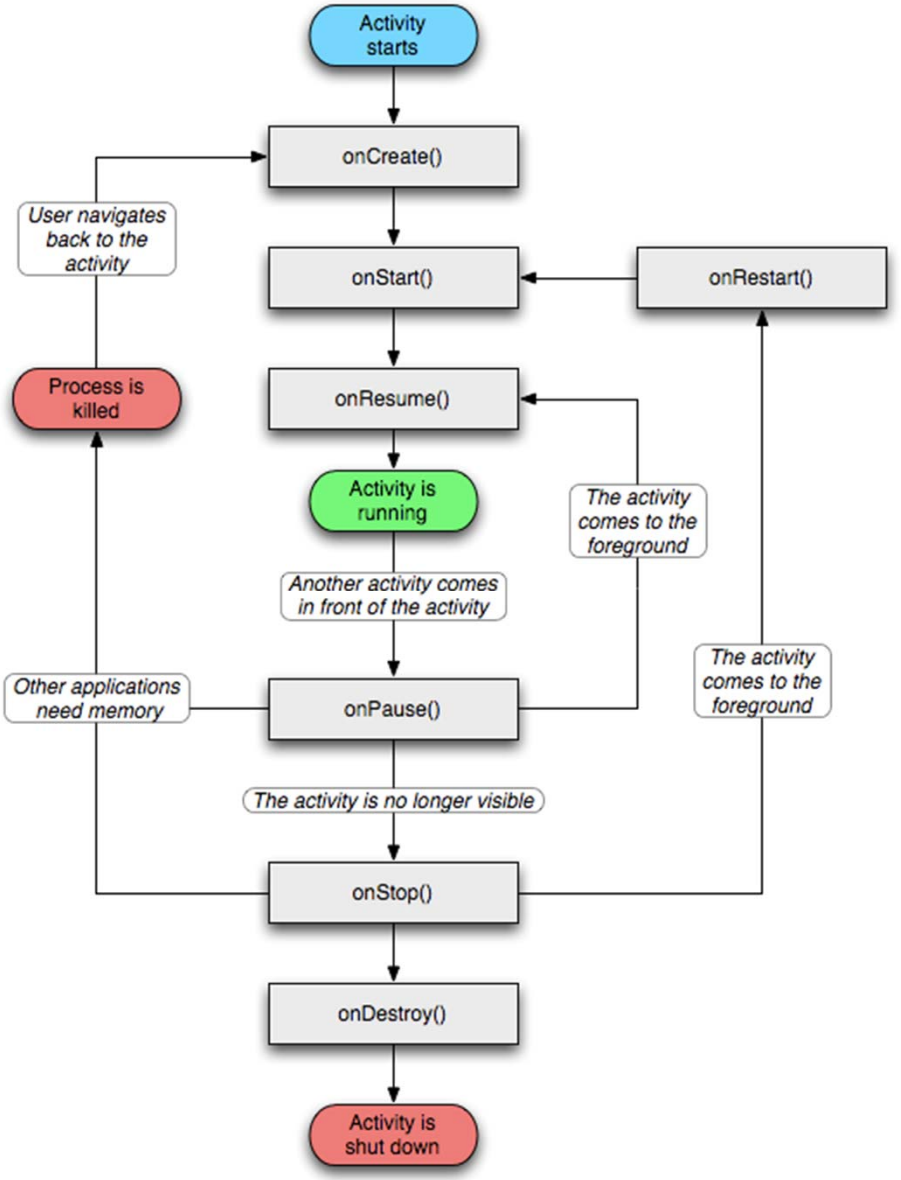






Life Cycle of an Activity

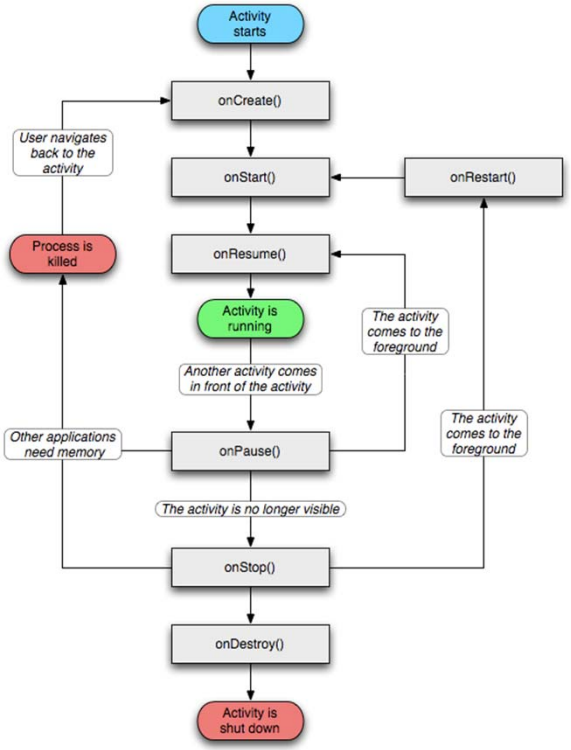
- Life cycle not directly controlled by application
- System can kill an application to free up memory.
- Controll through *onCreate*(), *onPause*(), *onStop*() ... methods



An Example

```
public class LifeCycleTest extends Activity {
    private static final String TAG = "ActivityLifeTest";

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Log.v(TAG, "onCreate");

        Uri uri = Uri.parse("http://maps.google.com/maps?f=d&saddr" +
                "=startLat%20startLng&daddr=endLat%20endLng&hl=en");
        Intent it = new Intent(Intent.ACTION_VIEW, uri);
        startActivity(it);
    }

    public void onStart() {
        super.onStart();
        Log.v(TAG, "onStart");
    }

    public void onResume() {
        super.onResume();
        Log.v(TAG, "onResume");
    }

    public void onPause() {
        super.onPause();
        Log.v(TAG, "onPause");
    }
```

```
    public void onPause() {
        super.onPause();
        Log.v(TAG, "onPause");
    }

    public void onStop() {
        super.onStop();
        Log.v(TAG, "onStop");
    }

    public void onRestart() {
        super.onRestart();
        Log.v(TAG, "onReStart");
    }

    public void onDestroy() {
        super.onDestroy();
        Log.v(TAG, "onDestroy");
    }
}
```



Intents/Intent Filters

- Provide a **late runtime (asynchronous) binding** between the code in different applications (for activities, services, and broadcast receivers)
- **Intents**: Simple message objects that represent an intention to do something
- **Intent Filters**: A declaration of capacity and interest in offering assistance to those in need
- An "intent" is made up a number of pieces of information describing the action or the service. (Specifically for activities and services)
  - **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 (Universe Resource Identifier).

Example:
*Intent newActivity = new **Intent**(MyActivity.this, OtherActivity.class);*
***startActivity(**newActivity);*

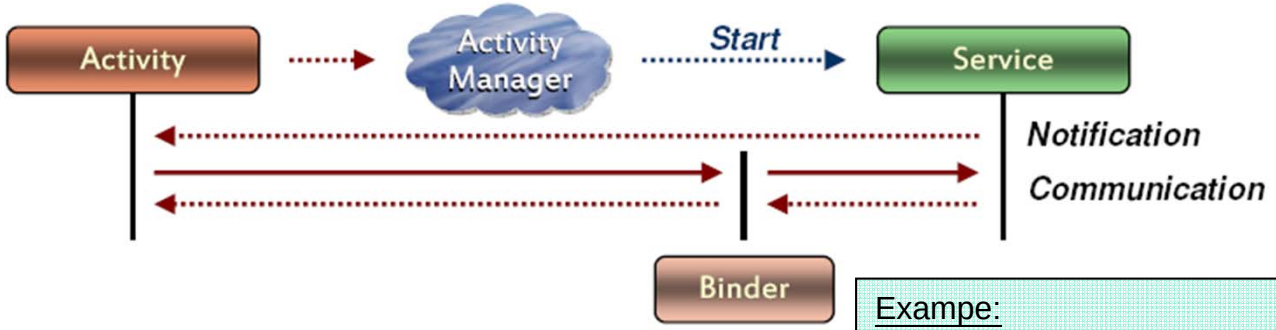
REF: Slides from "Android Anatomy and Physiology," Patrick Brady ©

Services

- Similar to activities, but without UI
- For long-running background tasks
- Framework “Service” class must be extended

❖ Core Services
- Activity Manager
- Package Manager
- Window Manager
- Resource Manager
- Content Providers
- View System

❖ Hardware Services
- Telephony Service
- Location Service
- Bluetooth Service
- WiFi Service
- USB Service
- Sensor Service

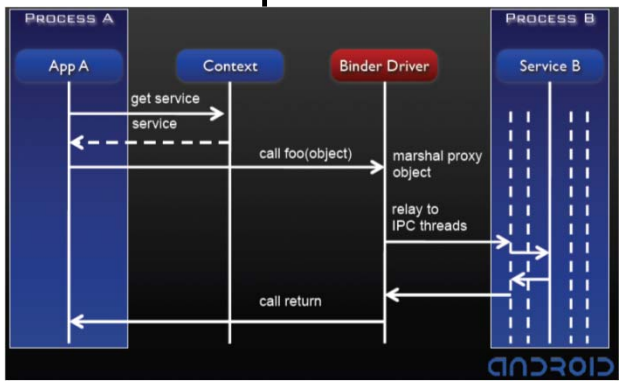


Exampe:
startService(new Intent(this_activity.this, some_service.class));

REF: Slides from "Android Anatomy and Physiology," Patrick Brady ©

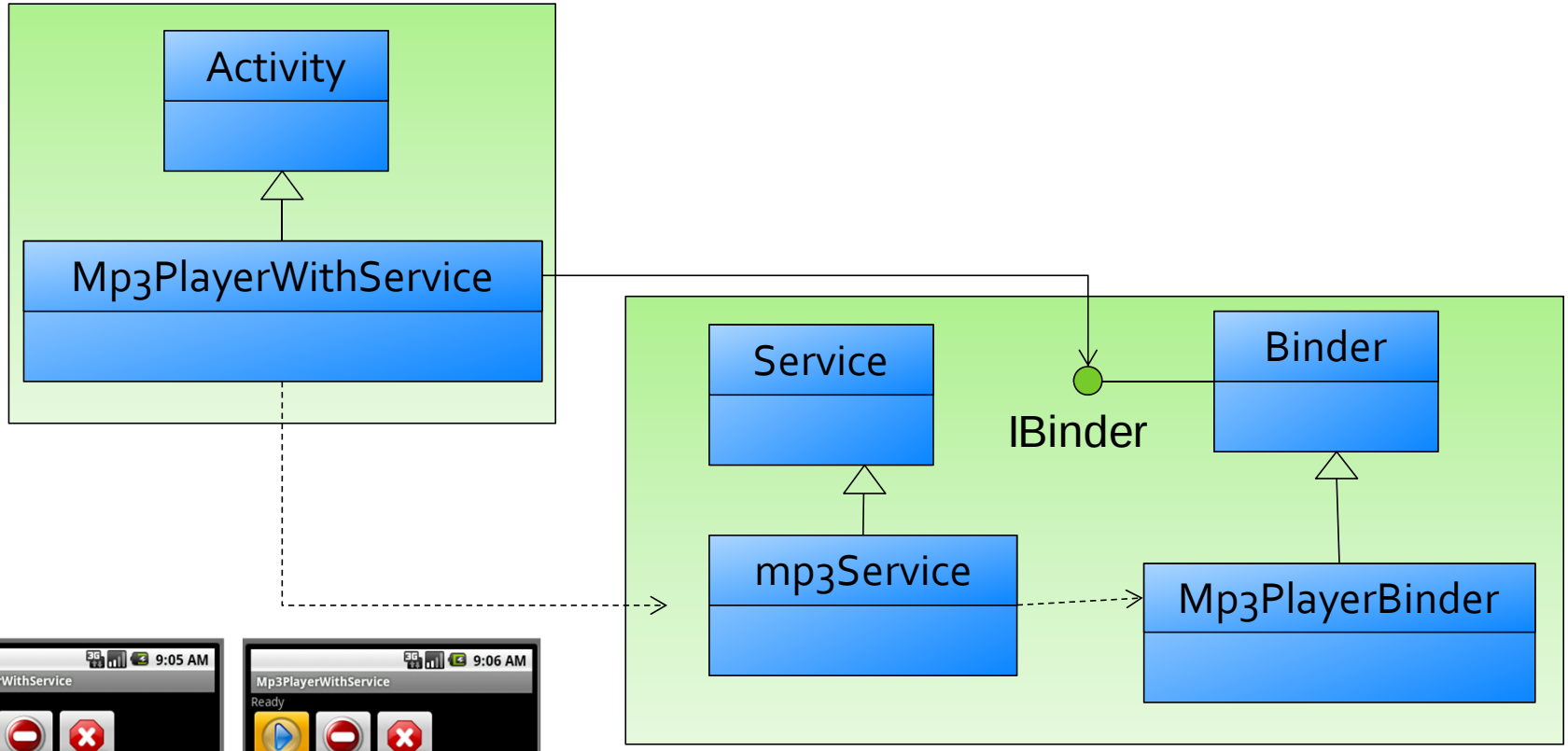
Binder

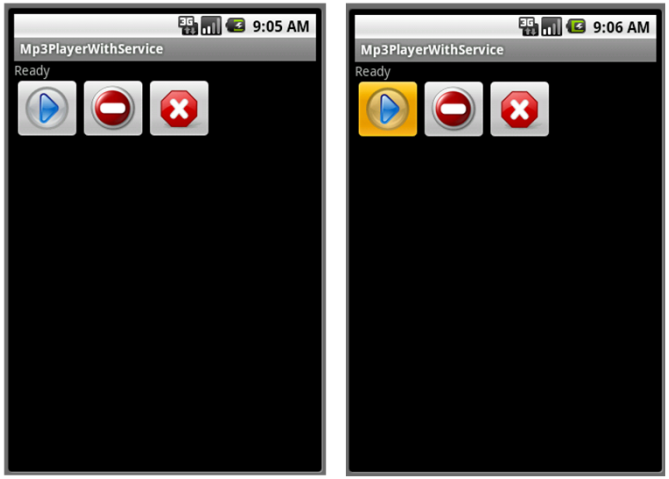
- Problem to solve
  - Applications and Services may run in separate processes but must communicate and share data.
  - IPC can introduce significant processing overhead and security holes.
- Solution
  - Driver to facilitate inter-process communication (IPC)
  - High performance through shared memory
  - Reference counting, and mapping of object references across processes
  - Synchronous calls between processes



REF: Slides from "Android Anatomy and Physiology," Patrick Brady ©

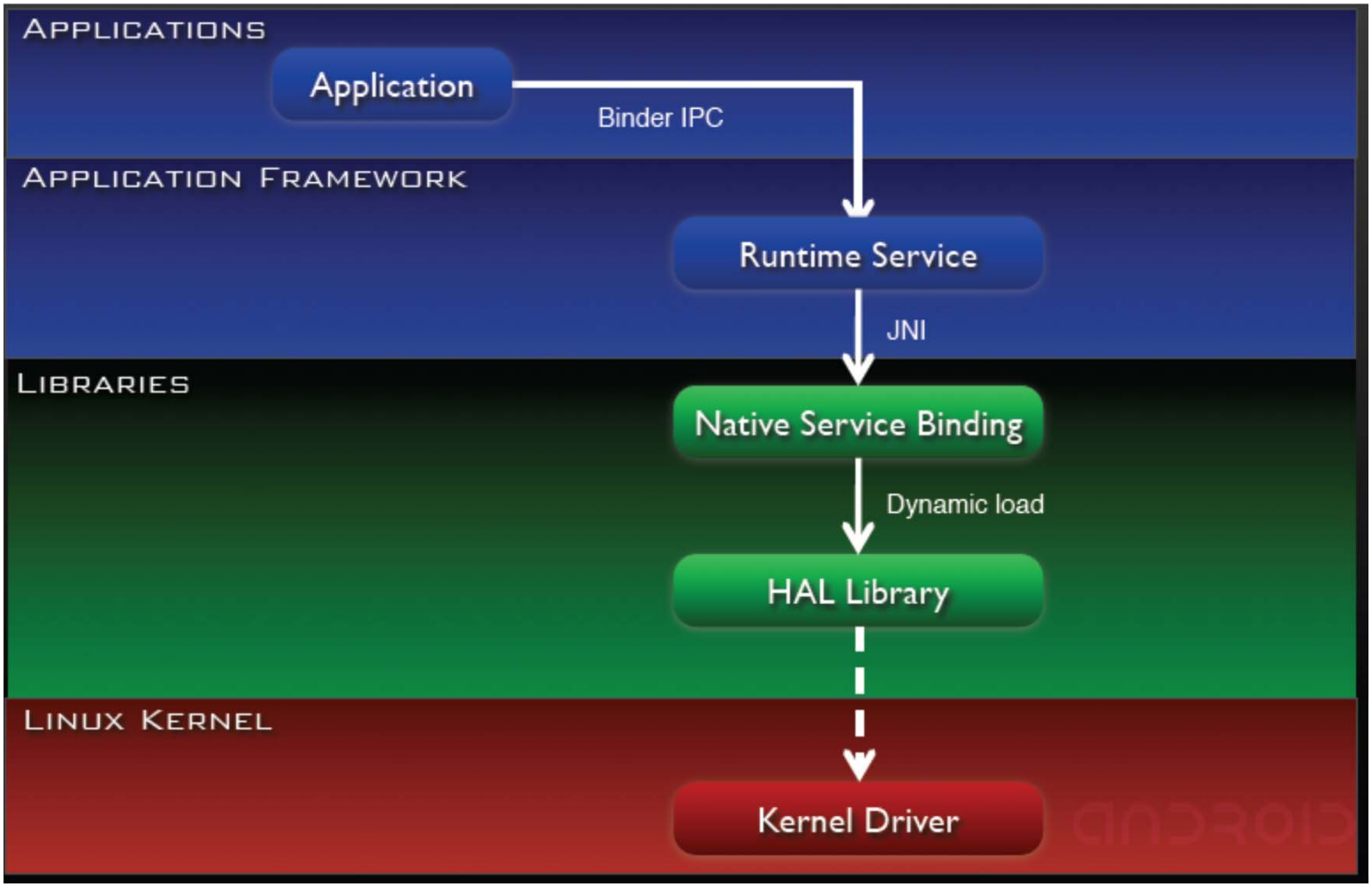
Binder Example





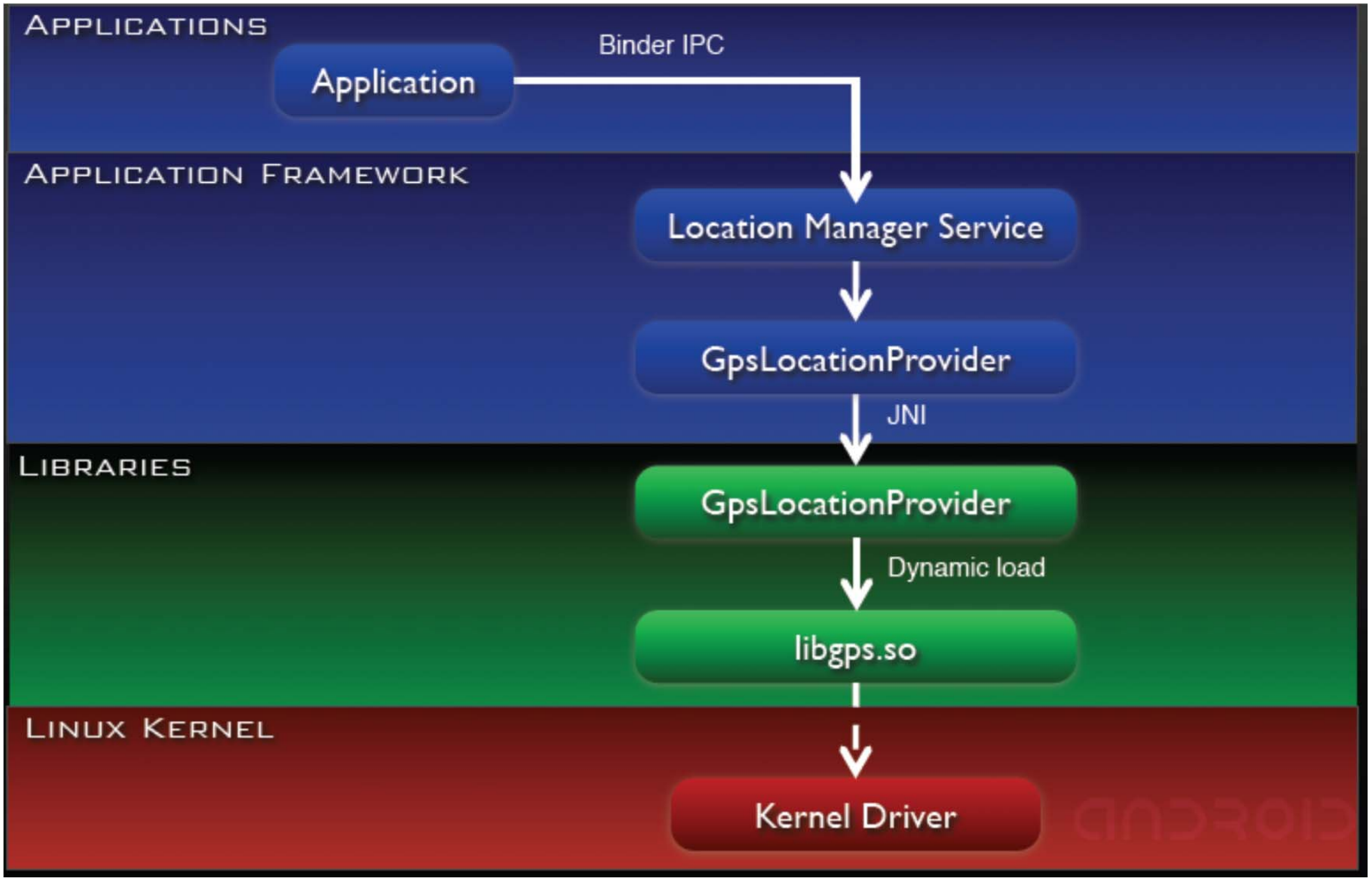
A more complex IPC: AIDL - Android Interface Definition Language

App → Runtime Service → lib



REF: Slides from "Android Anatomy and Physiology," Patrick Brady ©

Example



REF: Slides from "Android Anatomy and Physiology," Patrick Brady ©

Sensor and SensorManager

- Sensor type (Sensor class)
  - Digital Compass (Orientation), Magic Desire Aria
  - Accelerometer & Gravitation, Magic Desire Aria
  - Absolute position (GPS), Magic Desire Aria
  - Proximity, Desire Aria
  - Electromagnetic spectrum (WIFI) Magic Desire Aria
  - Sound, Magic Desire Aria
  - Light, Desire Aria
  - Temperature, etc.
- Sampling rate
  - Fastest, game, normal, user interface.
  - ! Requested sampling rate is not guaranteed !
- Accuracy
  - High, low, medium, unreliable.

Use *SensorManager.*getSensorList(int) to get the list of available Sensors.

Hardware-oriented Features

Feature	Description
Camera	**A class that enables your application to interact with the camera to snap a photo, acquire images for a preview screen, and modify parameters used to govern how the camera operates.**
Sensor	**Class representing a sensor. Use getSensorList(int) to get the list of available Sensors.**
SensorManager	**A class that permits access to the sensors available within the Android platform.**
SensorEventListener	**An interface used for receiving notifications from the SensorManager when sensor values have changed. An application implements this interface to monitor one or more sensors available in the hardware.**
SensorEvent	**This class represents a sensor event and holds information such as the sensor type (e.g., accelerometer, orientation, etc.), the time-stamp, accuracy and of course the sensor's data.**
MediaRecorder	**A class, used to record media samples, that can be useful for recording audio activity within a specific location (such as a baby nursery). Audio clippings can also be analyzed for identification purposes in an access-control or security application. For example, it could be helpful to open the door to your time-share with your voice, rather than having to meet with the realtor to get a key.**
GeomagneticField	**This class is used to estimated estimate magnetic field at a given point on Earth, and in particular, to compute the magnetic declination from true north.**
FaceDetector	**A class that permits basic recognition of a person's face as contained in a bitmap. Using this as a device lock means no more passwords to remember — biometrics capability on a cell phone.**

Add SensorManager to get Updates

```
import android.view.View.OnClickListener;

public class LifeLog extends Activity implements OnClickListener
{
    private SensorManager mSensorManager;
    private LinearLayout mainLayout;
    private ToggleButton tbStart;
    private ToggleButton tbStop;

    // Called when the activity is first created.
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        tbStart = new ToggleButton();
        tbStart.setOnClickListener(this);
        mainLayout.addView(tbStart);
        tbStop = new ToggleButton();
        tbStop.setOnClickListener(this);
        mainLayout.addView(tbStop);

        setContentView(mainLayout);

    }

    @Override
    protected void onClick(View v){
      if(v == tbStart)
        //Do Something

      else if(v == tbStop)
        //Do Something Else
    }
}
```

Add MySensorView

```
private class MySensorView extends View implements SensorEventListener {

  public MySensorView(Context context) {
  }

  @Override
  public void onDraw(Canvas canvas) {
    // TODO Auto-generated method stub
  }

  @Override
  public void onPaint() {
    // TODO Auto-generated method stub
  }

  @Override
  public void onStatusChanged() {
    // TODO Auto-generated method stub
  }
}
```

Add ability to graph sensors output

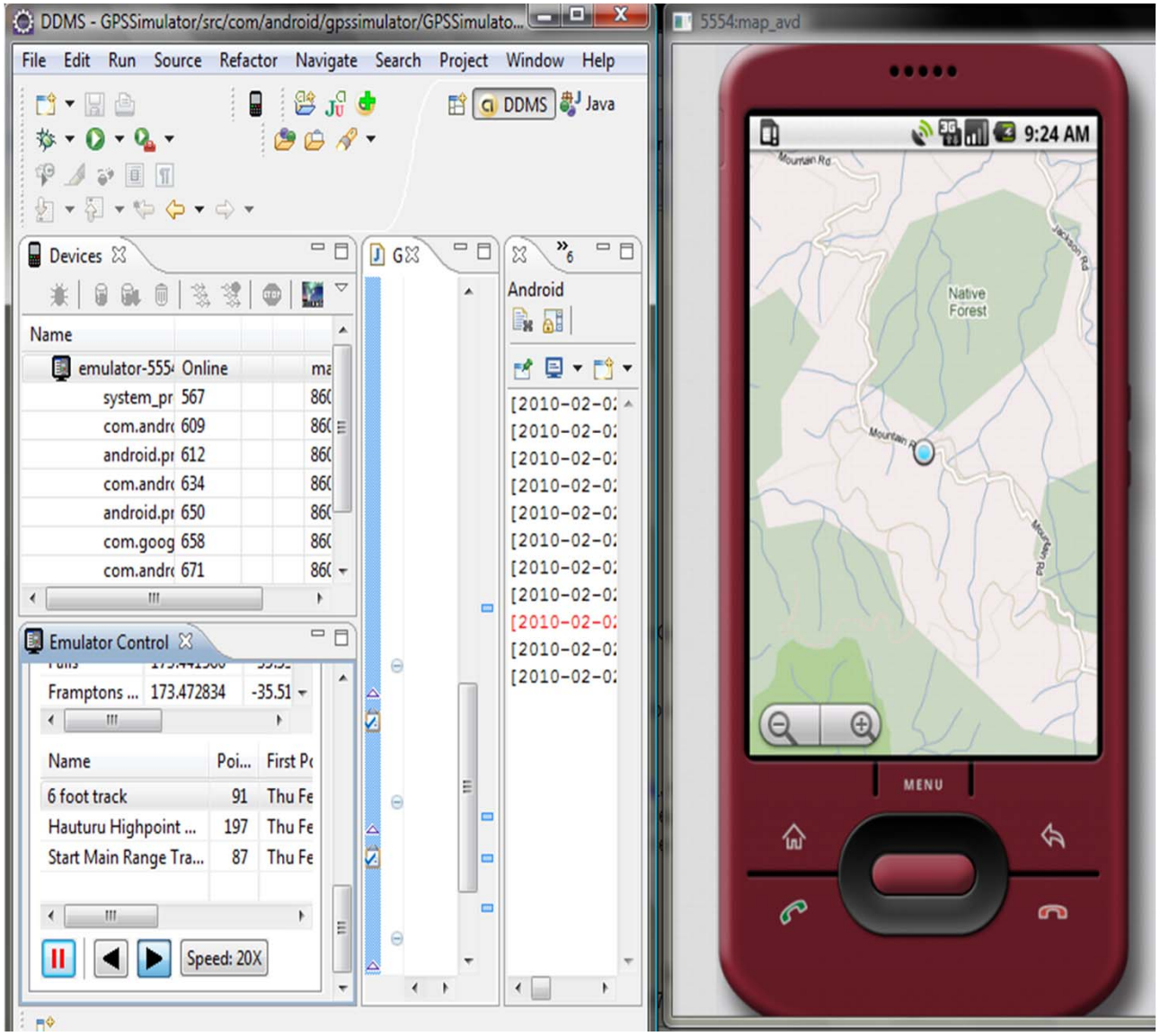
- Override the onDraw() & onPaint() methods.

```
private class MySensorView extends View implements SensorEventListener {

  @Override
  public void onLocationChanged(Location loc) {
  }

  @Override
  public void onDraw(Canvas canvas) {
    // TODO Auto-generated method stub
  }

  @Override
  public void onPaint() {
    // TODO Auto-generated method stub
  }

  @Override
  public void onStatusChanged() {
    // TODO Auto-generated method stub
  }
}
```

Add MySensorView to mainLayout

```
import android.view.View.OnClickListener;

public class LifeLog extends Activity implements OnClickListener
{
    private SensorManager mSensorManager;
    private LinearLayout mainLayout;
    private ToggleButton tbStart;
    private ToggleButton tbStop;

    // Called when the activity is first created.
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        tbStart = new ToggleButton();
        tbStart.setOnClickListener(this);
        mainLayout.addView(tbStart);
        tbStop = new ToggleButton();
        tbStop.setOnClickListener(this);
        mainLayout.addView(tbStop);

        setContentView(mainLayout);

    }

    @Override
    protected void onClick(View v){
      if(v == tbStart)
        mySensorView = new MySensorView(this);
        mainLayout.addView(mySensorView);
      else if(v == tbStop)
        //Do Something Else
    }
}
```

View the sensor values on your phone



Smartphone Sensor Web #3

「Introduction to Android Platform/Smartphone」

🗁. References

★. Android Dev Guide
http://developer.android.com/guide/topics/fundamentals.html

★. Pro Android by Hashimi & Komatineni (2009)

★. Patrick Brady, "Android Anatomy and Physiology,"

★. William W.-Y. Liang, National Taipei University of Technology, "System Integration for the Android Operating System,"

Questions?

REF: *http://www.pocket-lint.com/news/30712/android-powered-microwave-cooking-google*

Questions?

1.

Smartphone Sensor Web #3

「Table of Contents」

Smartphone Sensor Web #4~#11

「Table of Contents」

🗁. **Class #4~#10 Individual projects**

★. **Sensor-based Twit using a smartphone**

🗁. **Class #? Industrial Visit**

★.

🗁. **Class #11 Final Demonstration**

What can be done ?

スマートフォン搭載センサを活用してみる

他の無線センサを接続する

例：心電センサからの心拍数と血圧計算
加速度センサの信号と体動による変化

- Embedded sensors
  - ✓ Automatic motion detection from accelerometers (sit, walk, run)
  - ✓ Speaking/Listening time count from microphone
  - ✓ Location sensitive (GPS) sensing application
  - ✓ . . .
- Outside sensors (wireless)
  - ✓ Simple stress checker from ECG and/or Pulse
  - ✓ Chewing real-time counting from bone-conduction microphone
  - ✓ . . .