System Services: Sensors and GPS

Jianjun Chen (Jianjun.Chen@xjtlu.edu.cn)

Sensors

- Most Android-powered devices have built-in sensors that measure the physical environment
- Commonly used smart phone sensors
- Motion sensors
  - 3-axis Accelerometer
  - 3-axis Magnetometer
  - Gyroscope
- Environmental sensors
  - Light sensor
  - Noise

https://www.livescience.com/40103-accelerometer-vs-gyroscope.html

Access Sensors

- You can access sensors available on the device and acquire raw sensor data by using the Android sensor framework. Steps involved:
  1. Determine which sensors are available on a device.
  2. Determine an individual sensor's capabilities, such as its maximum range, ~~manufactu~~rer, power requirements, and resolution.
  3. Register and unregister sensor event listeners that monitor sensor changes.
  4. Acquire raw sensor data and define the minimum rate at which you acquire sensor data.

Step 1: Get List of Sensors

- First identify the sensors that are on the device
  - obtain an instance of the `SensorManager` class by calling the `getSystemService()` method.
  - passing in the `SENSOR_SERVICE` argument.

```
private SensorManager sensorManager;
...
sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

- We can (optionally) get a listing of every sensor:

```
List<Sensor> allSensors = sensorManager.getSensorList(Sensor.TYPE_ALL);
```

Step 2: Check Sensor Capabilities

- Test if certain type of sensor exists.

```
if (sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD) != null){
    // Success! There's a magnetometer
} else {
    // Failure! No magnetometer.
}
```

- Sensor types constants:

let you determine how close the face of a device is to an object*

- TYPE_ACCELEROMETER
- TYPE_GRAVITY
- TYPE_GYROSCOPE
- TYPE_LIGHT
- TYPE_MAGNETIC_FIELD
- TYPE_PRESSURE ← Air pressure
- TYPE_PROXIMITY
- TYPE_TEMPERATURE

*during a telephone call, proximity sensors play a role in detecting (and skipping) accidental touchscreen taps when mobiles are held to the ear.

Step 3-4: Register Listener

- To monitor sensor data you need to implement two callback methods in `SensorEventListener`:
  - A sensor's accuracy changes.
    - `onAccuracyChanged(Sensor s, int accuracy)`
  - A sensor reports a new value.
    - `onSensorChanged(SensorEvent e)`
- Then, register the listener with sampling rate.

```
sensor = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
sensorManager.registerListener(listener, sensor,
                                   SensorManager.SENSOR_DELAY_NORMAL);
```

The rate sensor events are delivered at

The Sensor Object

- The sensor object allows you to get a lot of information about the related sensor.
  - `getResolution()`
  - `getMaximumRange()`

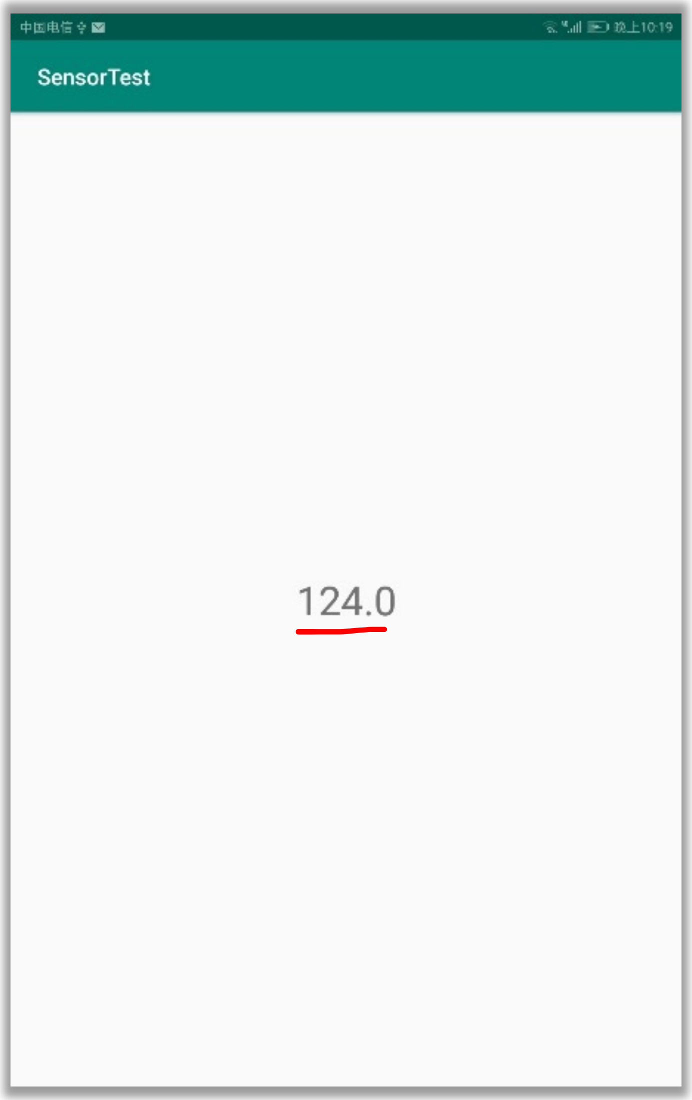
  - `getMinDelay()`: returns the minimum time interval (in microseconds) that a sensor can use to sense data.
    - For a streaming sensor (reports data repeatedly at a frequency), it will return a valid value.
    - For a non-streaming sensor (only report data when sensor feels the change), it will return 0.

Sensors Access: Summary

- Four types of interfaces
  - `SensorManager`: You can use this class to create an instance of the sensor service.
  - `Sensor`: You can use this class to create an instance of a specific sensor.
  - `SensorEvent`: used to create a sensor event object, which provides information about a sensor event.
  - `SensorEventListener`: used to create two callback methods that receive notifications (sensor events)
- Two basic tasks based on the above interfaces
  - Identifying sensors and sensor capabilities
  - Monitor sensor events

Sensor Example

- We will implement an app that tracks the current light level received from the ambient light sensor.



Full example: Listener

```java
public class LightSensorTest extends AppCompatActivity {
    SensorEventListener listener = new SensorEventListener() {
        @Override
        public void onSensorChanged(SensorEvent sensorEvent) {
            float lux = sensorEvent.values[0];
            TextView tv= (TextView) findViewById(R.id.luxIndicator);
            tv.setText(String.valueOf(lux));
        }

        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy) {
            // Do something here if sensor accuracy changes.
        }
    };
......
```

Full example: Activity

```
private SensorManager sensorManager;
private Sensor sensor;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_light_sensor_test);

    sensorManager =
        (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    sensor = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
    sensorManager.registerListener(listener, sensor,
                                   SensorManager.SENSOR_DELAY_NORMAL);
}

@Override
protected void onPause() {
    super.onPause();
    sensorManager.unregisterListener(listener);
}
```

Location Service

How to obtain current location

Accuracy issues

Location Service

- To be able to use location service, you will need:

```
LocationManager locationManager = (LocationManager)
                   this.getSystemService(Context.LOCATION_SERVICE);
```

```
Activity::getSystemService(String name)
```

- Based on this `LocationManager`, there are two ways of obtaining location information:
  - Network-based: Using cell tower and WiFi signal.
  - Satellite-based: Using GPS.

Permissions

- Declare permissions in the manifest file first.

```
<uses-permission android:name=
    "android.permission.ACCESS_COARSE_LOCATION"/>
```

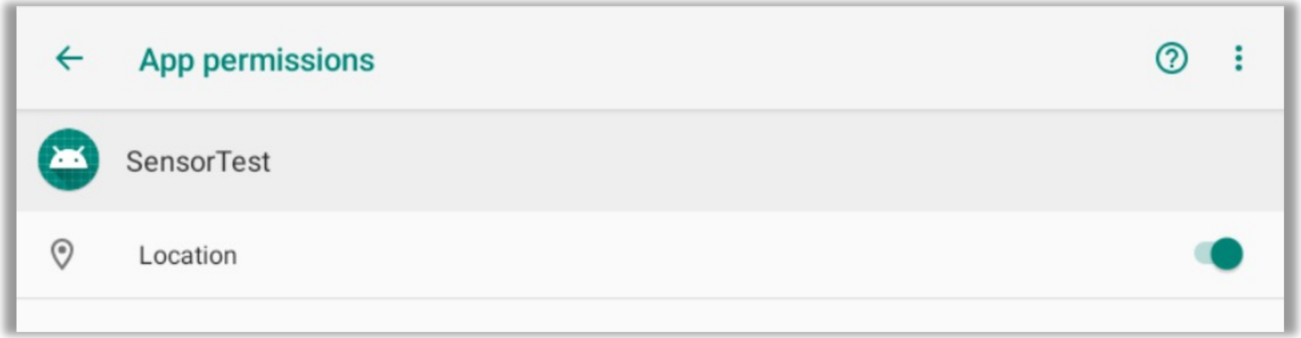
Grants permission to use network-based location service

```
<uses-permission android:name=
    "android.permission.ACCESS_FINE_LOCATION"/>
```

Grants permission to use network-based and GPS-based location service

Other Requirements

- Also remember to enable the permission for your app


App permissions

SensorTest

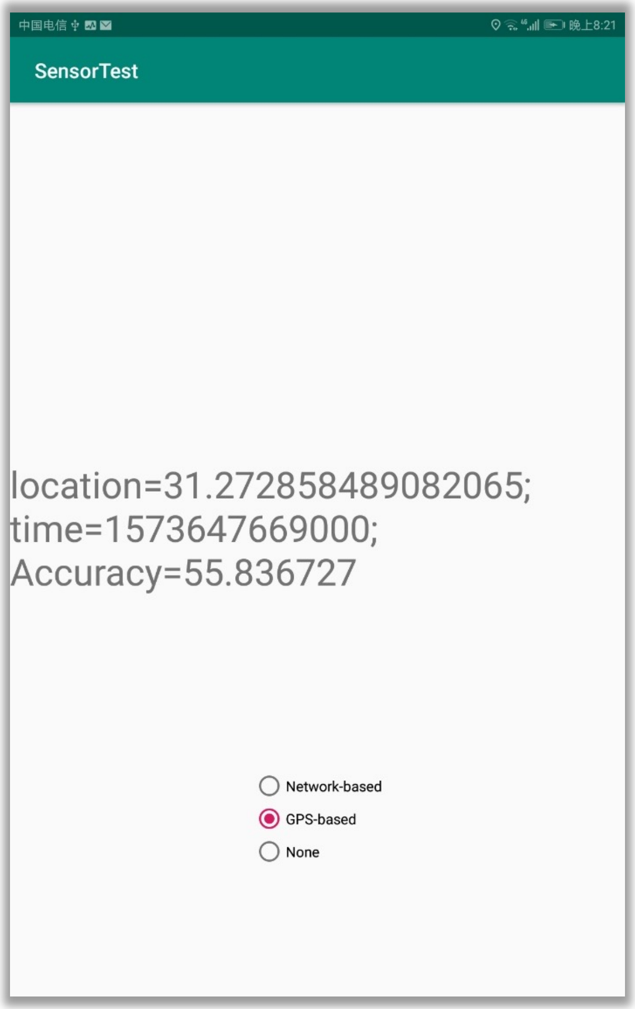
Location

- Then turn on GPS on your phone (unless you decide to use network-based location service)

The Example

- In the next example, we will build an app that allows you to:
  - Locate the current position using cell phone & WiFi signal.
  - Locate the current position using GPS-based location service.
  - Turn off location service.

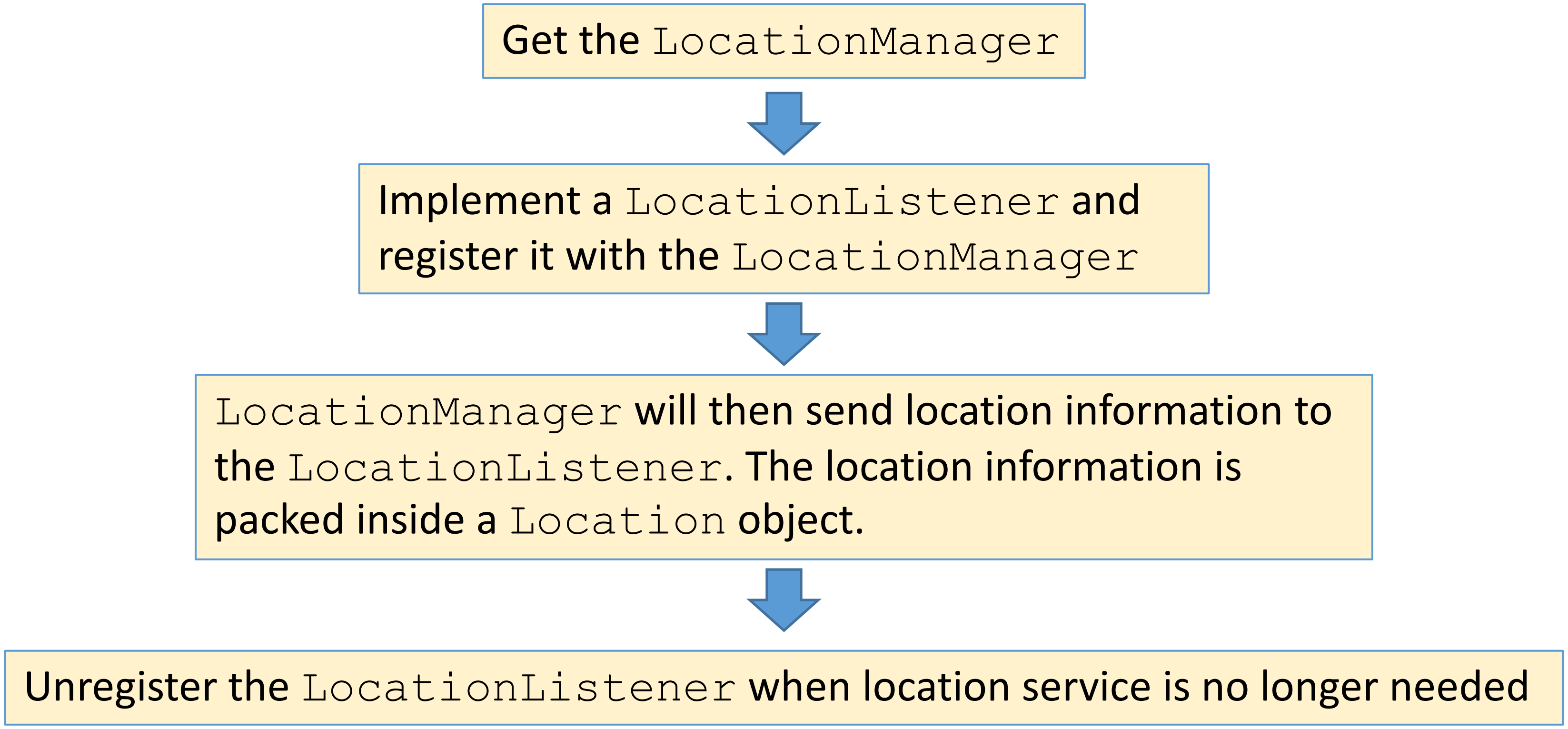
- The information will be shown in the middle of the screen.



location=31.272858489082065;
time=1573647669000;
Accuracy=55.836727

Network-based
[✓ GPS-based]
None

The logic Flow



Get the `LocationManager`

Implement a `LocationListener` and register it with the `LocationManager`

`LocationManager` will then send location information to the `LocationListener`. The location information is packed inside a `Location` object.

Unregister the `LocationListener` when location service is no longer needed

Step 1: Create the Activity

- We first create the activity and its layout.
  - Here, radio group is used for single choice

```
<RadioGroup
    android:id="@+id/locationMethod“ >
    <RadioButton
        android:id="@+id/nwBasedBtn"
        android:text="Network-based" />
    <RadioButton
        android:id="@+id/gpsBasedBtn"
        android:text="GPS-based" />
    <RadioButton
        android:id="@+id/noLocation"
        android:text="None" />
</RadioGroup>
```

```
public class LocationActivity extends AppCompatActivity
```

Step 2: Creating Callback Class

*This `Listener` is defined inside `LocationActivity`

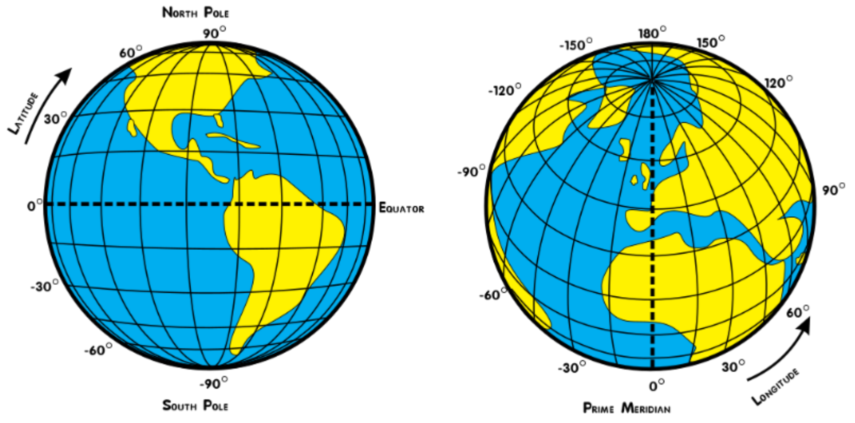
```
LocationListener listener = new LocationListener() {
    public void onLocationChanged(Location location) {
        TextView tv = (TextView) findViewById(R.id.locationLabel);
        tv.setText("location=" + location.getLatitude() + "; " +
                   "time=" + location.getTime() + "; " +
                   "Accuracy=" + location.getAccuracy() + "\n");
    }
    public void onStatusChanged(String provider,
                                int status, Bundle extras) {
        Log.i("LocationTester", "provider status is changed");
    }
    public void onProviderEnabled(String provider) {
        Log.i("LocationTester", "provider is enabled");
    }
    public void onProviderDisabled(String provider) {
        Log.i("LocationTester", "provider is closed");
    }
};
```

Deprecated

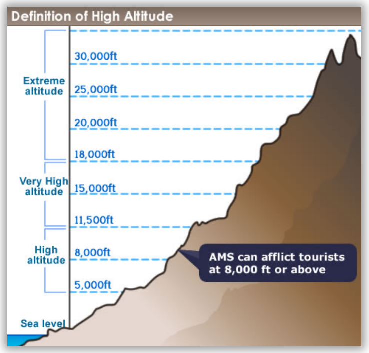
Can be found out by calling
`LocationManager.getAllProviders()`

The Location Class

- It's a data class representing a geographic location.
  - `getLatitude()`
  - `getLongitude()`
  - `getAltitude()`
  - `getSpeed()`



  - `getAccuracy()`
    - Get the estimated accuracy of this location, in meters.



- Check the API for details.

Step 3: Configure Activity

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    RadioGroup methodChoice = findViewById(R.id.locationMethod);
    final LocationManager locationManager = (LocationManager)
        LocationActivity.this.getSystemService(Context.LOCATION_SERVICE);

    methodChoice.setOnCheckedChangeListener(new
                    RadioGroup.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(RadioGroup radioGroup, int i) {
         See the next slide

        }
    });
}
```

Step 4: Register Callback Class

```
@Override
public void onCheckedChanged(RadioGroup radioGroup, int i) {
    if (i == radio1.getId()) {
        try {
            locationManager.requestLocationUpdates(
                LocationManager.NETWORK_PROVIDER, 0, 0, listener);
        } catch (SecurityException e) { ... }
    } else if (i == radio2.getId()) {
        try {
            locationManager.requestLocationUpdates(
                LocationManager.GPS_PROVIDER, 0, 0, listener);
        } catch (SecurityException e) { ... }
    } else {
        locationManager.removeUpdates(listener);
        TextView tv = (TextView) findViewById(R.id.locationLabel);
        tv.setText("Location Service OFF");
    }
}
```

Happens when permission not granted

minimum time interval between location updates, in milliseconds

minimum distance between location updates, in meters

```
requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, listener)
```

The type of location service:
`LocationManager.NETWORK_PROVIDER`
`LocationManager.GPS_PROVIDER`

a `LocationListener` whose `onLocationChanged(Location)` method will be called for each location update

Getting the Last Location

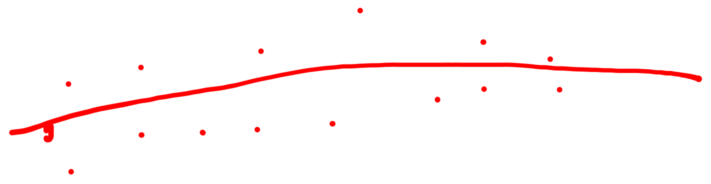
- A part from getting the location anew, you can obtain the last known location by calling:

```
Location lastLocation =
locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
```

- This method returns a Location object, from which you can get all needed information.
- You can use this to quickly get a (possibly inaccruate) position.

Obtain the Best Performance

- Flow for obtaining user location
  - Start application, get previous location stored locally
  - Start listening for updates from the location providers.
  - Maintain a "current best estimate" of location by filtering out new, but less accurate fixes.
  - Stop listening for location updates.
  - Take advantage of the last best location estimate.



Obtain the Best Performance

- Is the most recent location the most accurate?
  - Not necessarily
- You can validate the accuracy of a location via:
  - Check if the location retrieved is significantly newer than the previous estimate.
  - Check if the accuracy claimed by the location is better or worse than the previous estimate.
  - Check which provider the new location is from and determine if you trust it more.

Location Services: Considerations

- Location results may not be accurate
  - Multitude of location sources such as GPS, Cell-ID, and Wi-Fi have different errors
  - User movement could affect location accuracy.
- Power consumption-location measurement really drains the battery. To limit battery use:
  - Return updates less frequently
  - Restrict the set of Location Providers
  - Always check last known measurement
  - Turn off updates in `Activity::onPause()`.

Extended Reading

- In this lecture, both sensors and location services uses the `getSystemService()` function.
- There are many more types of system services.
- You should definitely check for those if your app is involved with any one of these system services.
- Check the API document for `Context.getSystemService()`

Practice

- Try to collect gravity sensor data
- Test GPS location.

*Only after finishing the previous lab.