SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

IAT 381 Mobile Computing

Summer 2015

Instructors: Helmine Serban

TA: Dolapo Toki

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Lecture 4: Intents. Accessing and Interacting with the Hardware of the Device.

Announcements

- Quiz 1
- Quiz 2
- Assignment 1: marking is in progress
- Assignment 2: released – explain in class, today
- Review course schedule

Quiz 1

- Review of the quiz questions - today

- Marked quizzes are not returned to students, however quizzes can be viewed in Helmine's office during office hours, or by appointment

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Quiz 2

- Will take place next week (Week 5) in class, during lecture
- Covers all content from Units 3 and 4
  - UI (lecture and lab)
  - Intents
  - Sensor framework

Course Schedule

- Review the schedule
- Have a look at the various deliverables and plan your work
- Specific due dates (day / time) are posted on Canvas

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Topics for Today

Intents

- Explicit intents
- Implicit intents

Interacting with device hardware

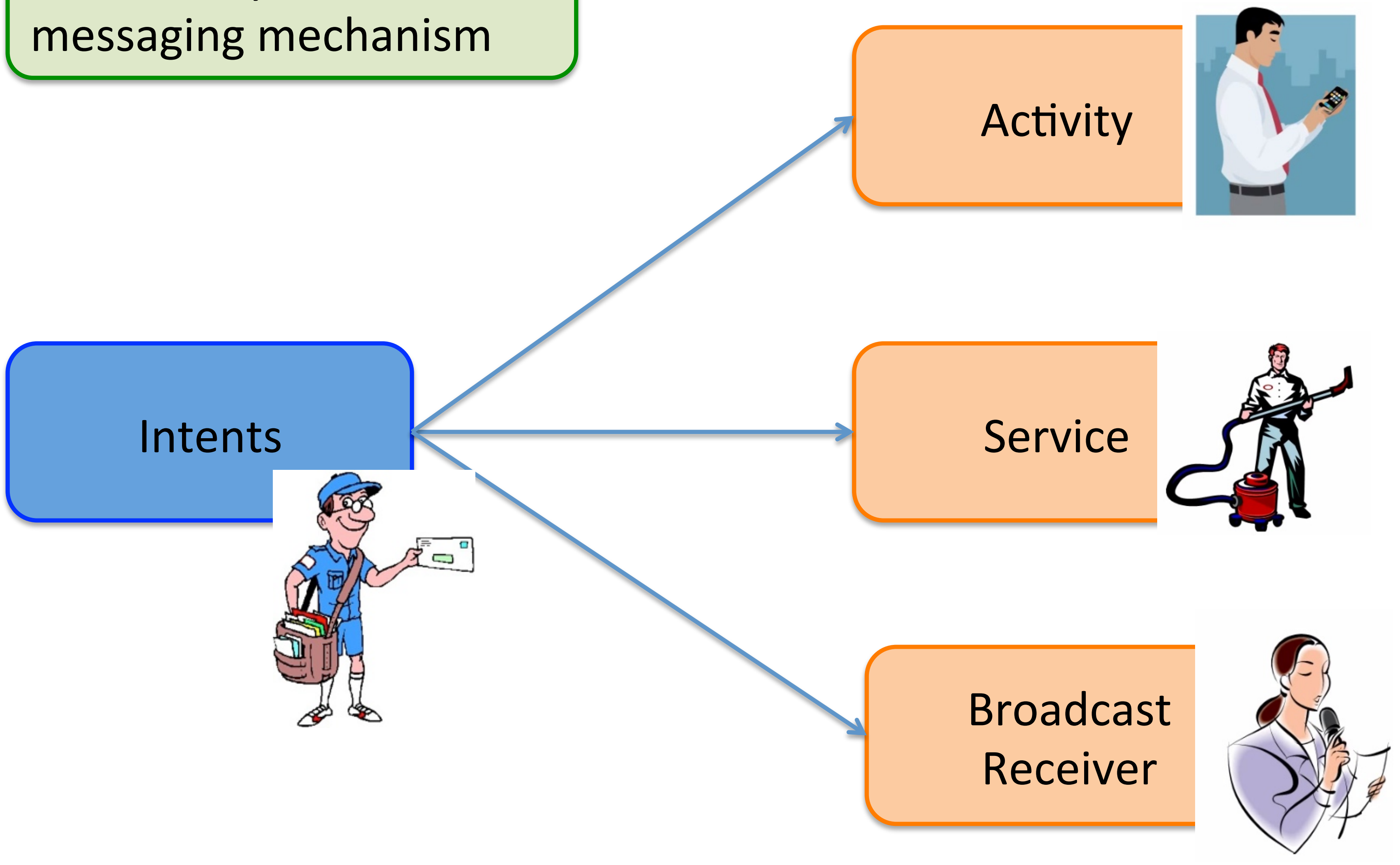
Monitoring the battery

What are Intents?

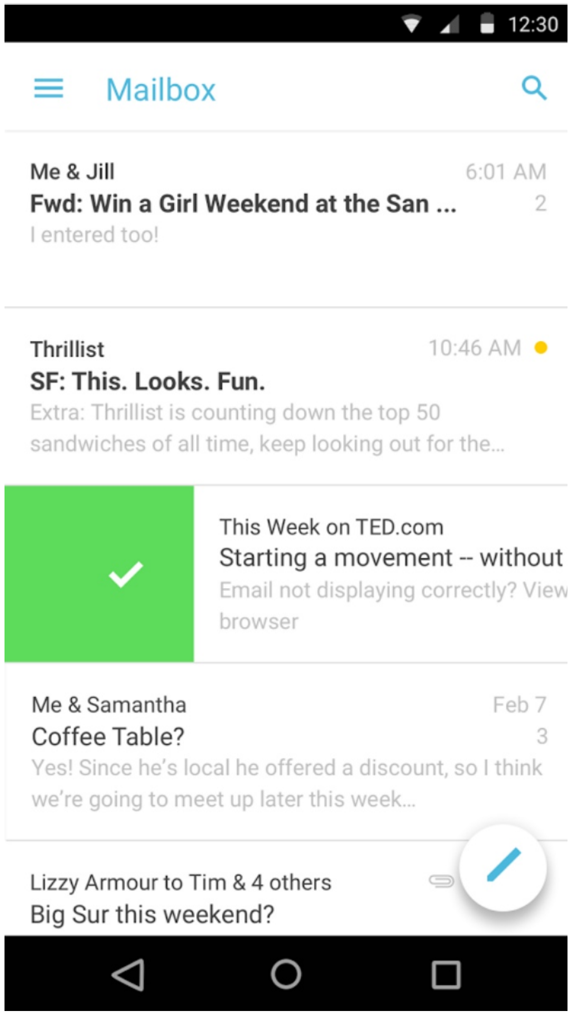
- Asynchronous messages
  - Allow Android components to request functionality from other components of the Android system
  - Can also be used to signal the Android system that a certain event has occurred – other components can register to this event via an intent filter

Intents are instances of the android.content.Intent class

Intent = asynchronous messaging mechanism



Example: Use Intent to Start a New Activity



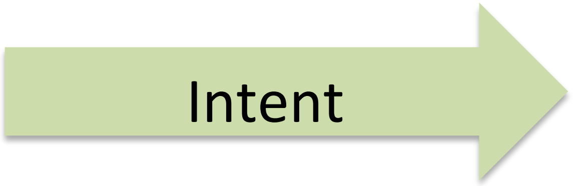
Me & Jill 6:01 AM
Fwd: Win a Girl Weekend at the San ... 2
I entered too!

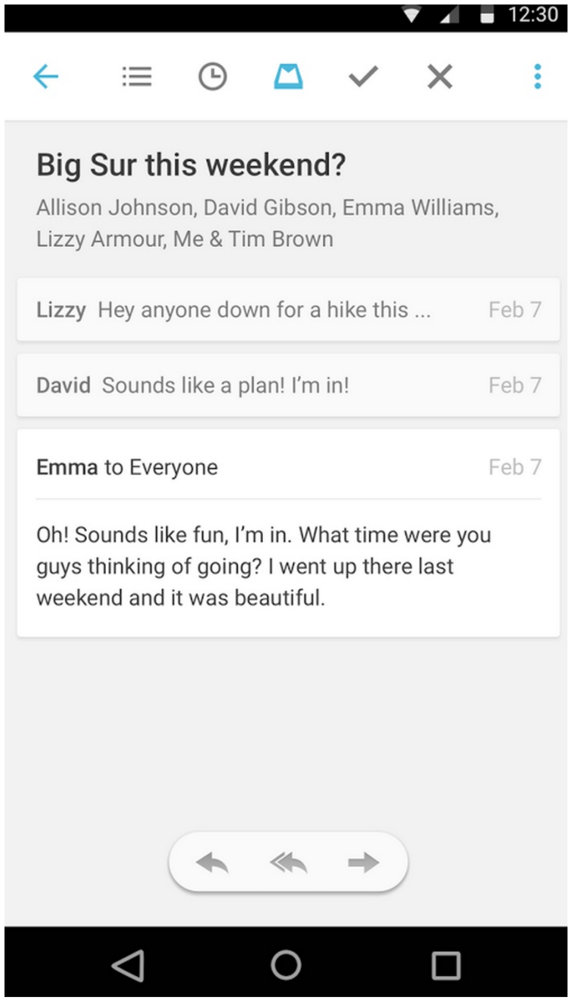
Thrillist 10:46 AM
SF: This. Looks. Fun.
Extra: Thrillist is counting down the top 50 sandwiches of all time, keep looking out for the...

This Week on TED.com
Starting a movement -- without
Email not displaying correctly? View browser

Me & Samantha Feb 7
Coffee Table? 3
Yes! Since he's local he offered a discount, so I think we're going to meet up later this week...

Lizzy Armour to Tim & 4 others
Big Sur this weekend?





Big Sur this weekend?
Allison Johnson, David Gibson, Emma Williams, Lizzy Armour, Me & Tim Brown

Lizzy Hey anyone down for a hike this ... Feb 7

David Sounds like a plan! I'm in! Feb 7

Emma to Everyone Feb 7

Oh! Sounds like fun, I'm in. What time were you guys thinking of going? I went up there last weekend and it was beautiful.

Example: Starting an Activity

- Activity sends and intent to the Android system which starts another activity

```
 //start the activity ActivityTwo
Intent i = new Intent(this, ActivityTwo.class);
startActivity(i);
```

context

Activity to be started

Intent Types

Explicit Intents

Explicitly define the **component** which should be called by the Android system

- Java class identifier

Implicit Intents

Specify the **action** which should be performed

- Can start any component that can handle the intended action

Explicit Intents

- Explicitly define the **component** which should be called by the Android system, by using the Java class as identifier

```
Intent i = new Intent (this, ActivityTwo.class);
i.putExtra("value1", "This value 1 for Activity
  Two");
i.putExtra("value2", "This value 2 for Activity
  Two");
```

Data: putExtra() – key/value pairs – key is always a string

Implicit Intents

- Specify the **action** which should be performed
- Optionally – provides data for the action
- Can start any component that can handle the intended action

Action to be performed

```
Intent i = new Intent(Intent.ACTION_VIEW);
i.setData(Uri.parse(“http://www.sfu.ca”));
startActivity(i);
```

- typically, web browser is registered to this intent, but other components could register as well

Implicit Intents

- The Android system searches for all components which are registered for the specific action and the data type
  - If only one component is found, it is started directly
  - Several components: the user will get a selection dialog and can decide which component should be used for the intent

Using Intents to Start Other Components

- If you send an Intent to the Android system, it needs to be specified which type of component the Intent should be sent

- Activities: `startActivity(Intent);`
- Services: `startService(Intent);`
  - We will explore services later in the course

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Sending the User to Another App

- Implicit intents
- Based on the **action**
  - – E.g., view, edit, send, get something, …
- Intents often include data - might be a Uri
  - – E.g., address, email message,…

```
Uri number = Uri.parse(“tel: 5551234”);
Intent callIntent = new Intent(Intent.ACTION_DIAL,
  number);
```

More Examples

View a map:

```
// Map point based on address
Uri location = Uri.parse("geo:0,0?q=1600+Amphitheatre
  +Parkway,+Mountain+View,+California");
      // Or map point based on latitude/longitude//
Uri location = Uri.parse("geo:37.422219,-122.08364?
  z=14"); // z param is zoom level
Intent mapIntent = new Intent(Intent.ACTION_VIEW,
  location);
```

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

More Examples

- View a webpage:

```
Uri webpage = Uri.parse("http://www.android.com");
Intent webIntent = new Intent(Intent.ACTION_VIEW,
  webpage);
```

Extra Data

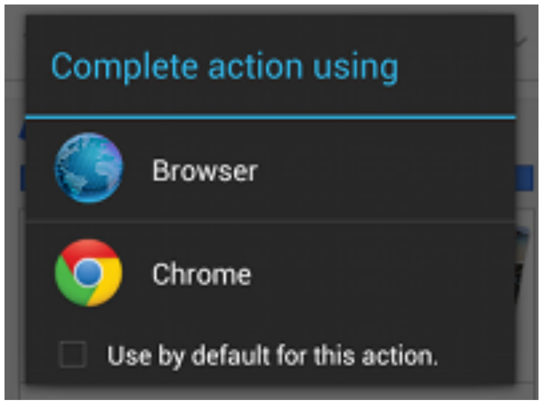
- `putExtra()`
- Example: send an email with attachment:

```
Intent emailIntent = new Intent(Intent.ACTION_SEND);
// The intent does not have a URI, so declare the "text/plain”
   type
emailIntent.setType(HTTP.PLAIN_TEXT_TYPE);
emailIntent.putExtra(Intent.EXTRA_EMAIL, new String[]
   {"jon@example.com"}); // recipients
emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Email subject");
emailIntent.putExtra(Intent.EXTRA_TEXT, "Email message text");
emailIntent.putExtra(Intent.EXTRA_STREAM, Uri.parse("content://
   path/to/email/attachment"));// You can also attach multiple
   items by passing an ArrayList of Uris
```

Verification Step

- Verify that there is an app to receive your intent
- `queryIntentActivities()`
  - Returns a list of activities capable of handling the intent
  - If the list is not empty, intent can be safely used

- This check should be performed when the activity first starts



Chooser Dialog

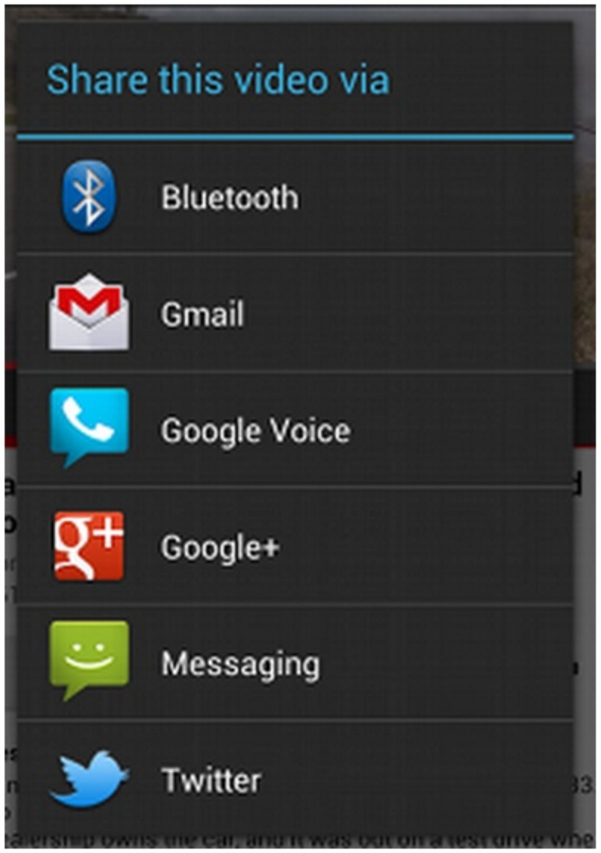


Figure 2. A chooser dialog.

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Example

```
// Build the intent
Uri location = Uri.parse("geo:0,0?q=1600+Amphitheatre+Parkway,+Mountain
   +View,+California");
Intent mapIntent = new Intent(Intent.ACTION_VIEW, location);

// Verify it resolves
PackageManager packageManager = getPackageManager();
List<ResolveInfo> activities =
   packageManager.queryIntentActivities(mapIntent, 0);
boolean isIntentSafe = activities.size() > 0;

// Start an activity if it's safe
if (isIntentSafe) {
    startActivity(mapIntent);
}
```

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Retrieving Result Data from the Called Activity

- Send the intent: `startActivityforResult()` method

- Once the called activity ends, the `onActivityResult()` method on the sending activity is called
  - The result from the triggered activity can be processed

Result from Started Activity

- Start a camera app and receive the captured photo as result
- Start the Contacts app in order to select a contact; receive the contact details as result

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Trigger an Activity for Result

```
public void onClick (View View)
{
  Intent i = new Intent(this, ActivityTwo.class);
  i.putExtra("Value1", "This Value1 for
  ActivityTwo");
  i.putExtra("Value2", "This Value2 for
  ActivityTwo");
  //request code
  startActivityforResult(i, REQUEST_CODE);
}
```

REQUEST CODE

- This is the argument that identifies the request

- When the result is received by the initial app, the same request code is provided
  - Initial (sending) app identifies the result based on the request code and determines how to handle this result

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Send Data (Result) to the Initial Activity

```
@Override
public void finish()
{
// Prepare data intent
   Intent data = new Intent();
   data.putExtra("returnKey1", ”This is the result");
   data.putExtra("returnKey2", ”We are sending data back");
   // Activity finished ok, return the data
setResult(RESULT_OK, data);
   super.finish();
}
```

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Receiving the result

```
@Override
protected void onActivityResult(int requestCode, int
   resultCode, Intent data) {
                if (resultCode == RESULT_OK && requestCode ==
   REQUEST_CODE) {
                if (data.hasExtra("returnKey1"))
                 {
                Toast.makeText(this,
   data.getExtras().getString("returnKey1"),Toast.LENGTH_SHORT
   ).show();     }
}}
```

`onActivityResult()` method

- This method has three arguments:
  - The request code that was passed to `startActivityforResult()`
  - A result code specified by the second activity
    - RESULT_OK: if operation was successful
    - RESULT_CANCELED: operation failed
  - The result data

T

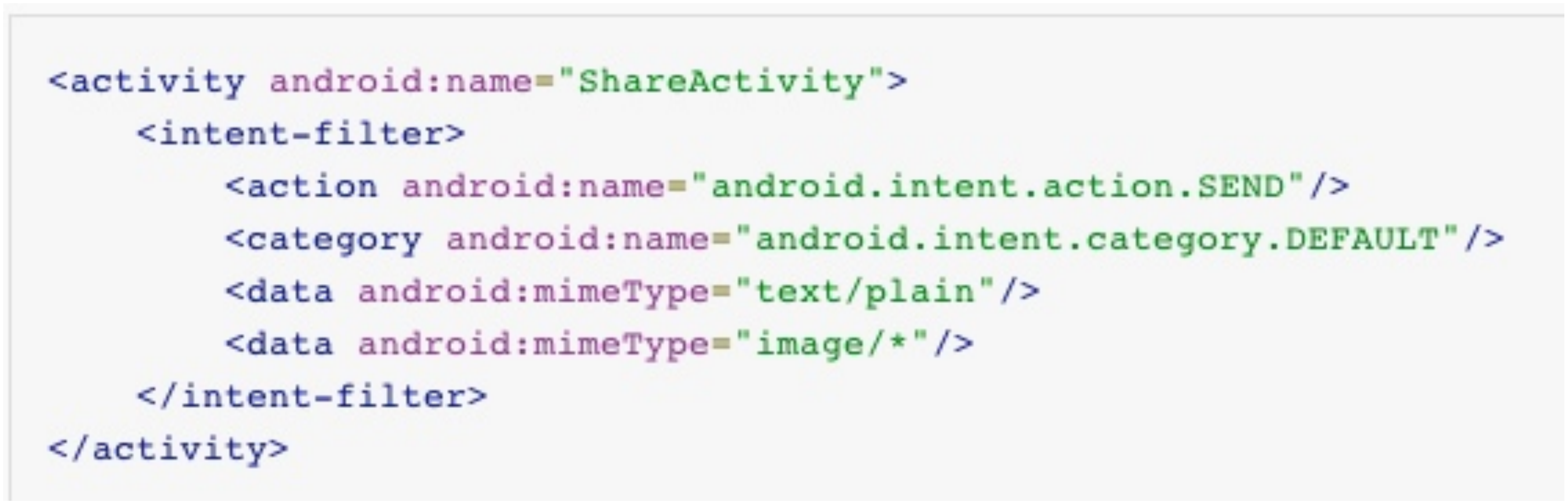
Intent Filters

- If your app can perform an action that might be useful for another app, it should be prepared to respond to action requests from other apps.
- Intent filter
- The system identifies all intent filters and adds the information to an internal catalog of intents supported by all installed apps.

Add an Intent Filter

- Has to be specific
  - Type of action
  - Data the activity accepts
- Action: string naming the action to perform
  - ACTION_SEND, ACTION_VIEW
- Data: description of data associated with the intent
- Category: additional way to characterize the activity handling the intent
  - User gesture, location, etc
  - CATEGORY_DEFAULT (default for all intents)

Example of Intent Filter



```
<activity android:name="ShareActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
        <data android:mimeType="image/*"/>
    </intent-filter>
</activity>
```

Device Hardware

Built-in sensors

- Motion, orientation, environment conditions

Data from sensors

- Raw data, high precision and accuracy

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Types of Sensors

Motion sensors
- Acceleration, forces, rotational forces along three axes
- Accerometers, gravity sensors, rotational vector sensors, gyroscopes

Environmental Sensors
- Ambient air temperature sensors, pressure, illumination, humidity

Position sensors
- Physical position of the device
- Orientation sensors, magnetometers,

Guidelines when Accessing Hardware

- No assumptions
  - Hardware may or may not exist
- Always check and verify optional features
- Exception handling and errors
- Return value checking
- Hardware features = device resources
  - Acquire late
  - Release as soon as done with them
  - Do not drain the device battery by misusing hardware resources

SFU

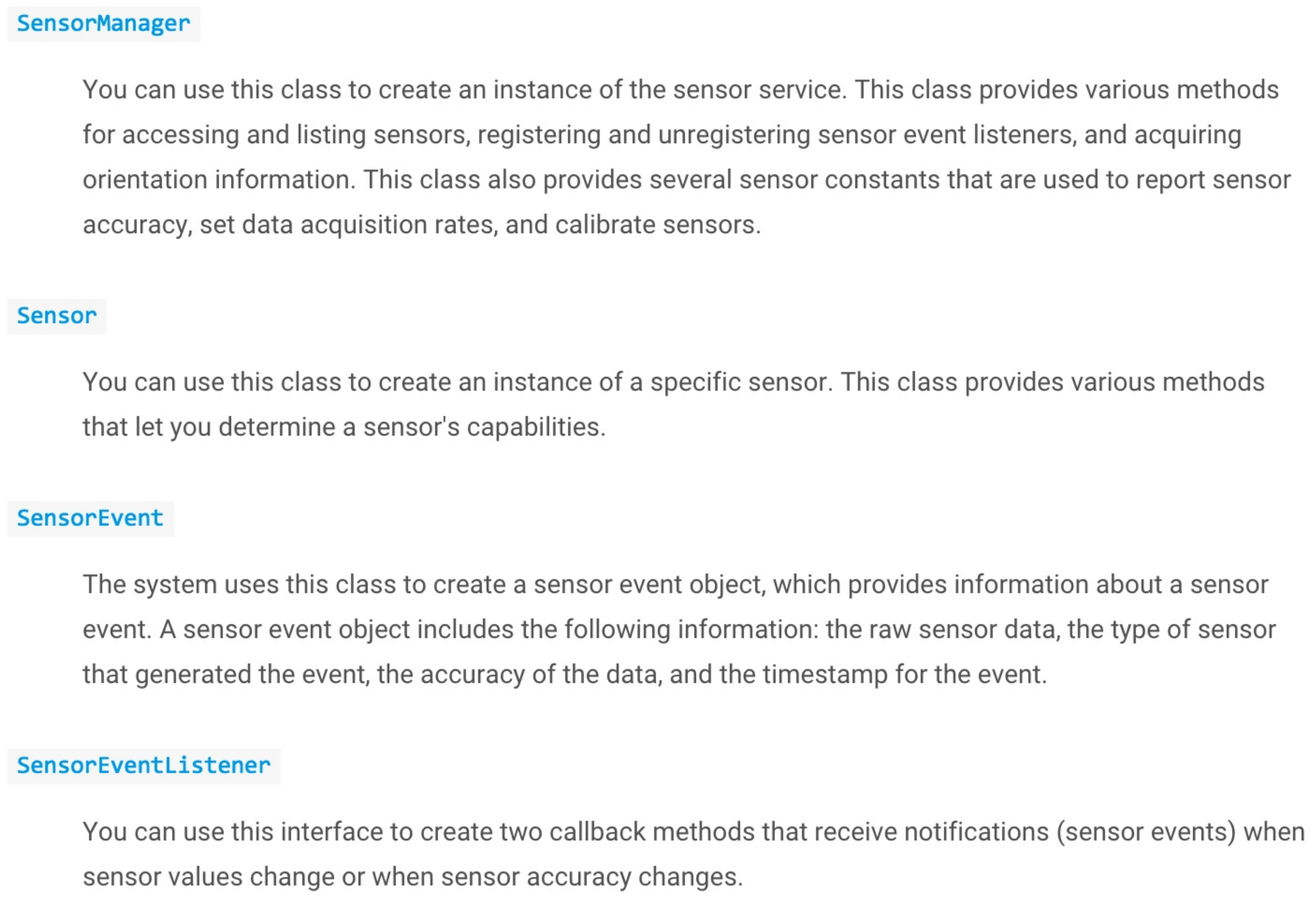
SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Android Emulator

- Limited / no support for simulating hardware sensors and the device battery
  - Testing on real device
  - **!!! Bring a physical device to the lab**

Android Sensors Framework

- Part of the android.hardware package



`SensorManager`

You can use this class to create an instance of the sensor service. This class provides various methods for accessing and listing sensors, registering and unregistering sensor event listeners, and acquiring orientation information. This class also provides several sensor constants that are used to report sensor accuracy, set data acquisition rates, and calibrate sensors.

`Sensor`

You can use this class to create an instance of a specific sensor. This class provides various methods that let you determine a sensor's capabilities.

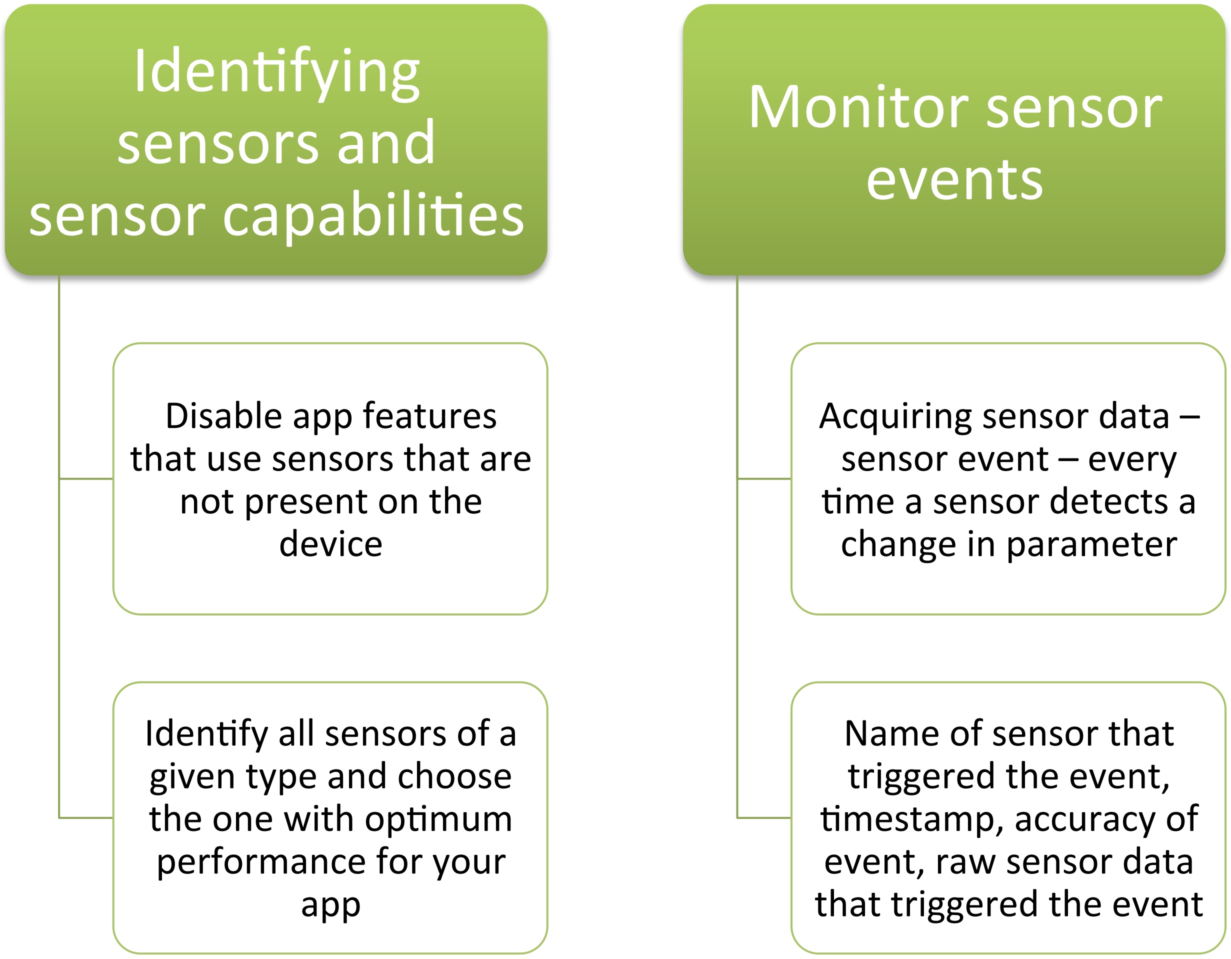
`SensorEvent`

The system uses this class to create a sensor event object, which provides information about a sensor event. A sensor event object includes the following information: the raw sensor data, the type of sensor that generated the event, the accuracy of the data, and the timestamp for the event.

`SensorEventListener`

You can use this interface to create two callback methods that receive notifications (sensor events) when sensor values change or when sensor accuracy changes.

Sensor-Related Tasks


Identifying sensors and sensor capabilities
Disable app features that use sensors that are not present on the device
Identify all sensors of a given type and choose the one with optimum performance for your app
Monitor sensor events
Acquiring sensor data – sensor event – every time a sensor detects a change in parameter
Name of sensor that triggered the event, timestamp, accuracy of event, raw sensor data that triggered the event

Process in Working With Sensors

Determine which sensors are available on a device.

Determine an individual sensor's capabilities, such as its maximum range, manufacturer, power requirements, and resolution.

Acquire raw sensor data and define the minimum rate at which you acquire sensor data.

Register and unregister sensor event listeners that monitor sensor changes.

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Identifying Sensors and Capabilities

- First: get a reference to the sensor device
  - Get an instance of the SensorManager class:

```java
private SensorManager mSensorManager;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

- Get a listing of every sensor on the device:

```java
List<Sensor> deviceSensors = mSensorManager.getSensorList(Sensor.TYPE_ALL);
```

Checking for a Certain Sensor

```
private SensorManager mSensorManager;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
if (mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD) != null){
  // Success! There's a magnetometer.
  }
else {
  // Failure! No magnetometer.
  }
```

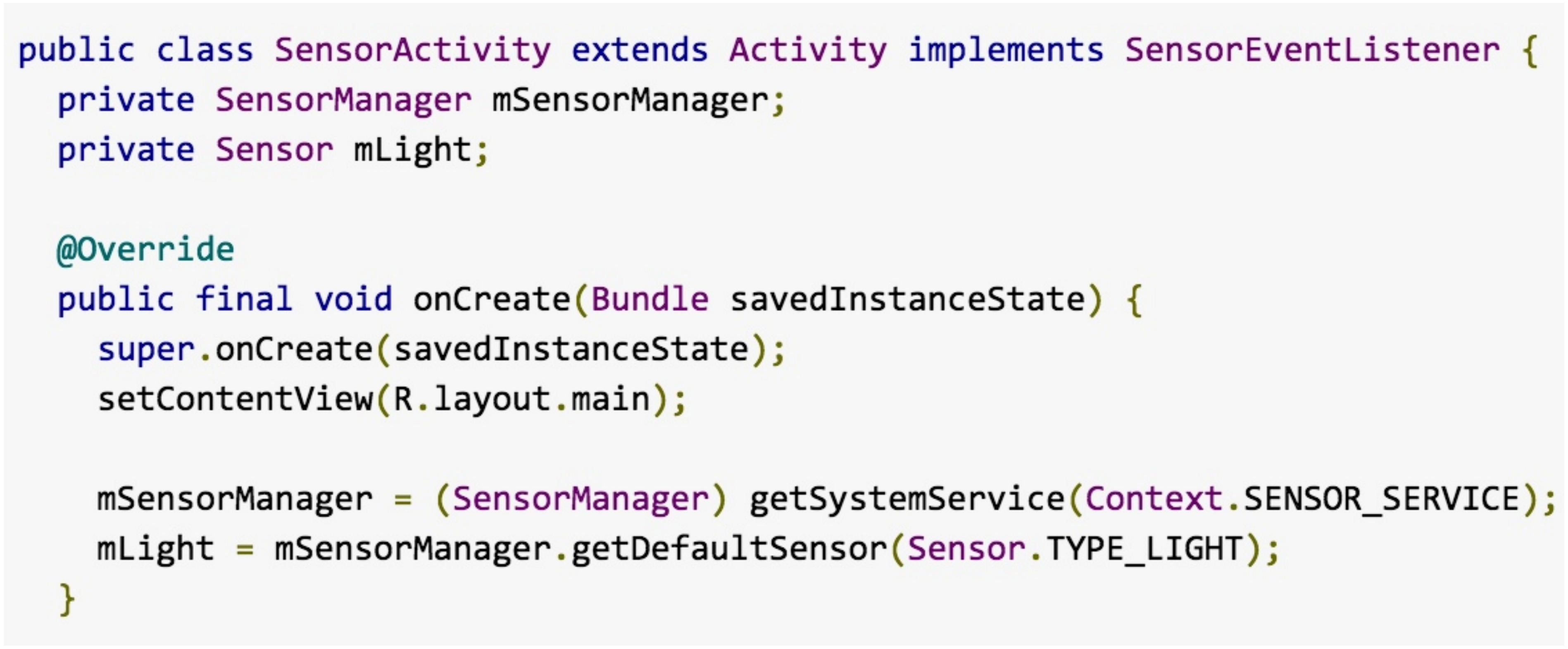
Sensors List

- public List<Sensor> getSensorList(int type);
- type is one of:
  - TYPE_ACCELEROMETER
  - TYPE_GYROSCOPE
  - TYPE_LIGHT
  - TYPE_MAGNETIC_FIELD
  - TYPE_ORIENTATION
  - TYPE_PRESSURE
  - TYPE_PROXIMITY
  - TYPE_TEMPERATURE
  - TYPE_ALL

Monitoring Sensor Events

- SensorEventListener interface: exposes two methods:
  - onAccuracyChanged(): accuracy changes, provides a reference to the Sensor object that changed, and the new accuracy value
  - onSensorChanged(): sensor reports a new value – new data, sensor that generated the data, timestamp, accuracy of data

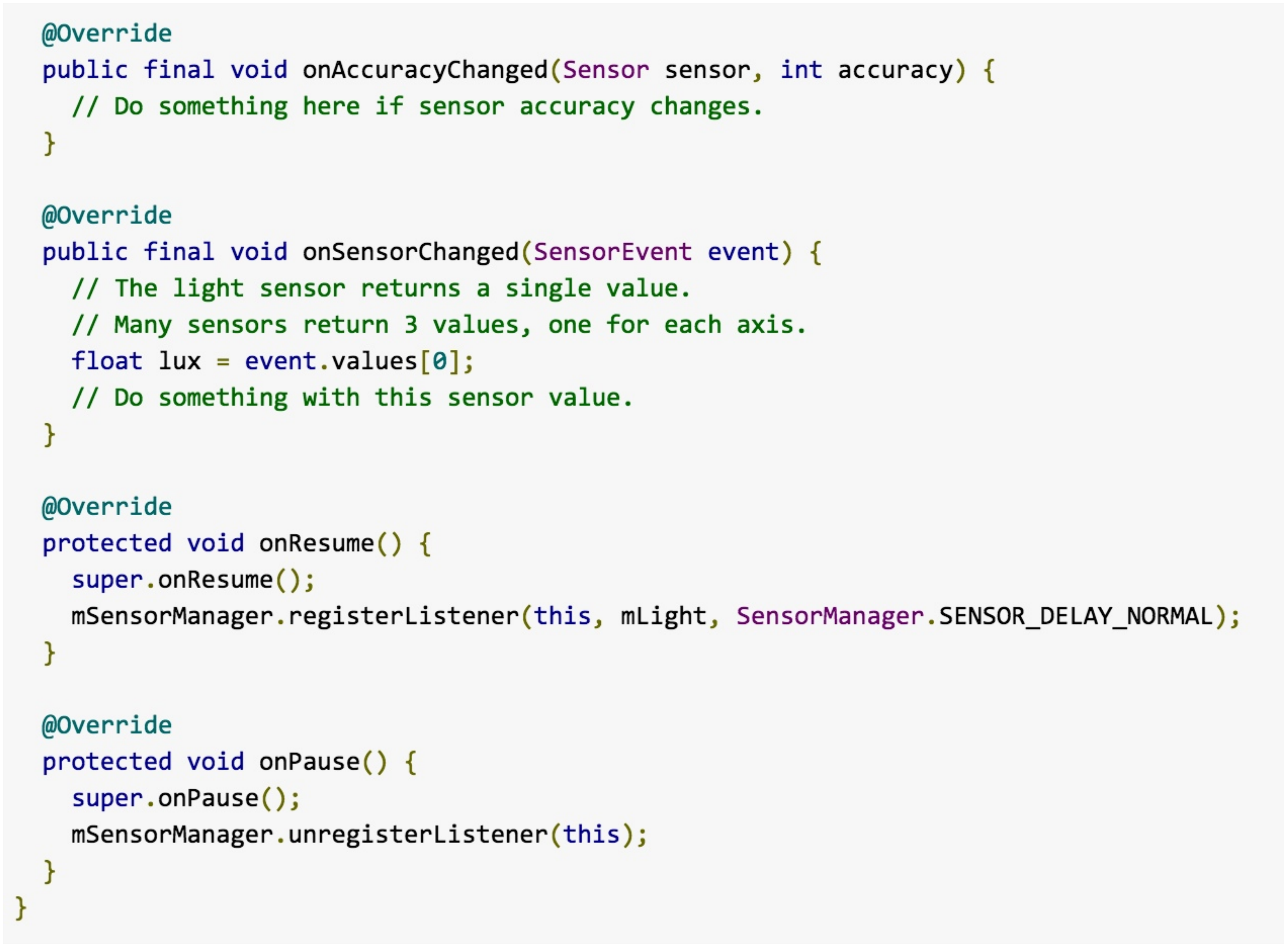
Sample Code



```java
public class SensorActivity extends Activity implements SensorEventListener {
  private SensorManager mSensorManager;
  private Sensor mLight;

  @Override
  public final void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    mLight = mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
  }
```



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

  @Override
  public final void onSensorChanged(SensorEvent event) {
    // The light sensor returns a single value.
    // Many sensors return 3 values, one for each axis.
    float lux = event.values[0];
    // Do something with this sensor value.
  }

  @Override
  protected void onResume() {
    super.onResume();
    mSensorManager.registerListener(this, mLight, SensorManager.SENSOR_DELAY_NORMAL);
  }

  @Override
  protected void onPause() {
    super.onPause();
    mSensorManager.unregisterListener(this);
  }
}
```

AndroidManifestFile - Sensors

- <uses-feature> tag – indicates which sensors are required by the application

```
<uses-feature android:name="android.hardware.sensor.barometer" />
<uses-feature
    android:name="android.hardware.sensor.gyroscope"
    android:required="false" />
```

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Acquiring a Reference to a Sensor

- getDefaultSensor() method of the SensorManager class

- This method takes a sensor type as parameter

```
Sensor accelSensor = sensors.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

Reading Sensor Data

- After we have a valid sensor object, we can read the sensor data periodically

- Sensor values: SensorEventListener object – send back the the sensor values to the app
  - The app must implement and register the SensorEventListener using the `registerListener()` method

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Methods to Implement

- SensorEventListener interface
  - `onAccuracyChanged()` – called whenever the accuracy of a given sensor changes
  - `onSensorChanged()` – called whenever the values of the sensor change
    - This method is used to inspect sensor information

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Sample: onSensorChanged()

```java
@Override
public void onSensorChanged(SensorEvent event) {
    StringBuilder sensorMessage =
        new StringBuilder(event.sensor.getName()).append(" new values: ");

    for (float value : event.values) {
        sensorMessage.append("[").append(value).append("]");
    }

    sensorMessage.append(" with accuracy ").append(event.accuracy);
    sensorMessage.append(" at timestamp ").append(event.timestamp);

    sensorMessage.append(".");

    Log.i(DEBUG_TAG, sensorMessage);
}
```

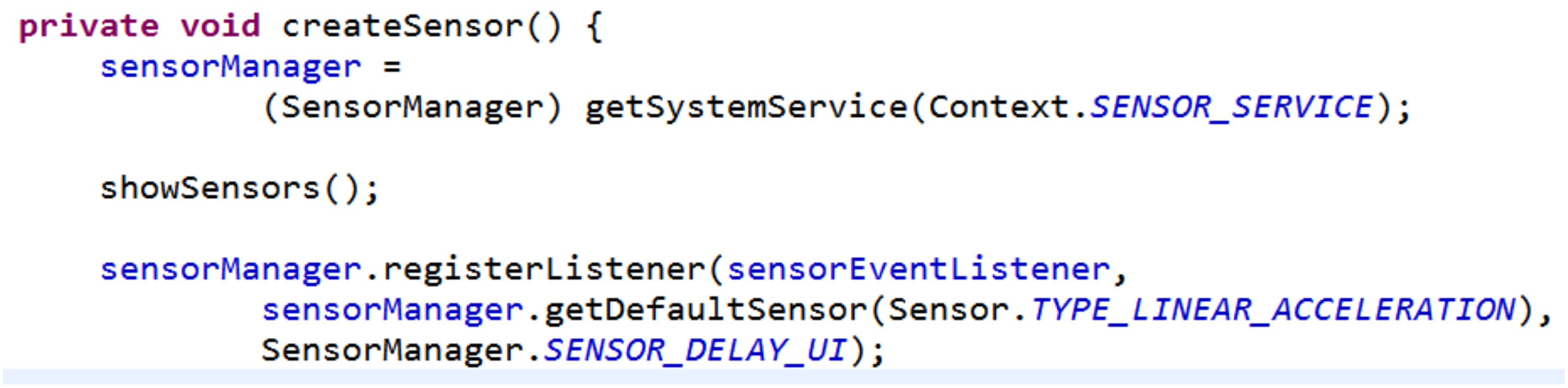
onSensorChanged() Method

- One single parameter: SensorEvent object
- The SensorEvent class:
  - Contains all the data about the sensor
    - Which sensor caused the event
    - The accuracy of the sensor
    - The sensor's current readings
    - Timestamp
  - What data to expect from each sensor? -> SensorEvent class documentation in Android SDK

Rate of Sensor Data

- Depending on the sensor in use
  - This rate might be high
- onSensorChanged() method – do as little as possible in this method
  - Processing data should not be done within this method

Example: Using Sensors

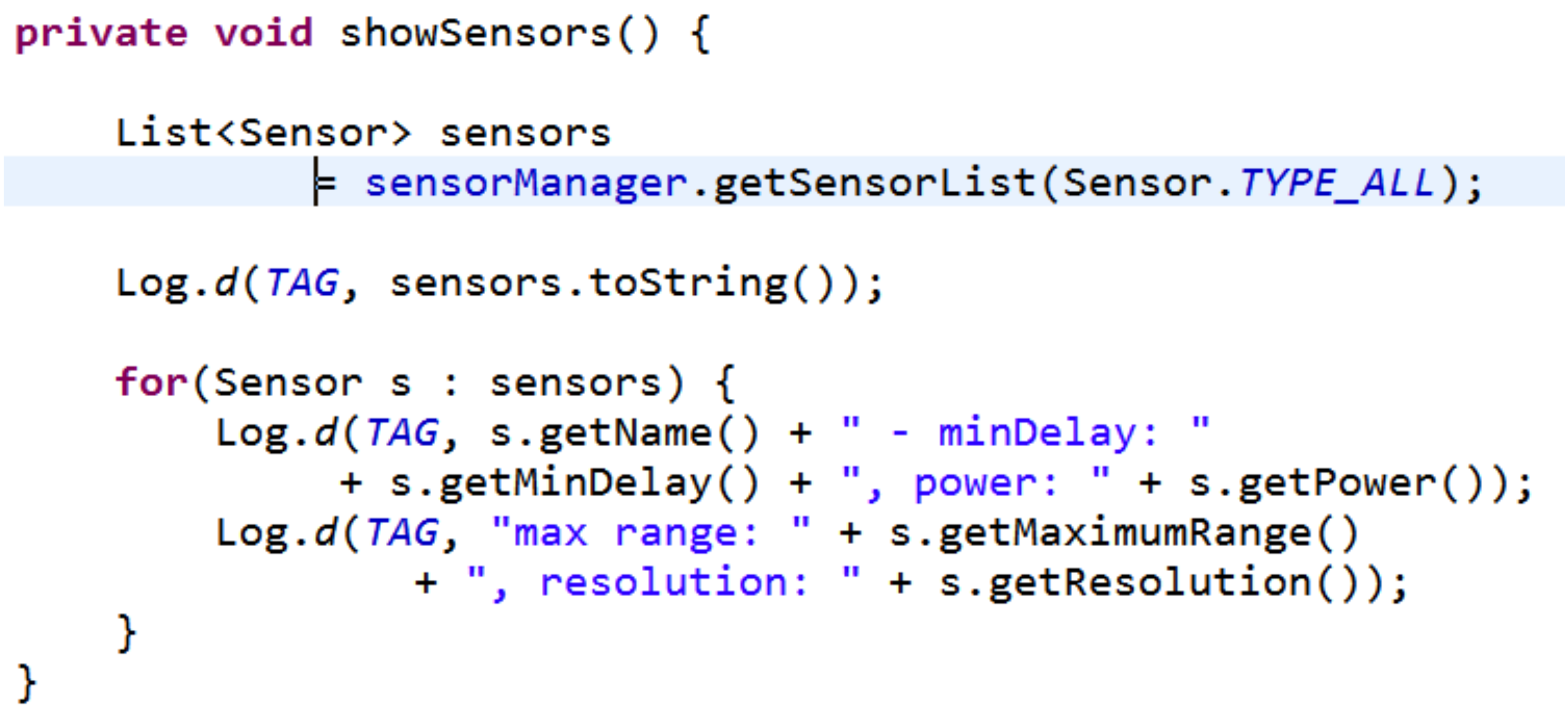


```
private void createSensor() {
    sensorManager =
            (SensorManager) getSystemService(Context.SENSOR_SERVICE);

    showSensors();

    sensorManager.registerListener(sensorEventListener,
            sensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION),
            SensorManager.SENSOR_DELAY_UI);
```

Listing Sensors on a Device to Log



```
private void showSensors() {

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

    Log.d(TAG, sensors.toString());

    for(Sensor s : sensors) {
        Log.d(TAG, s.getName() + " - minDelay: "
             + s.getMinDelay() + ", power: " + s.getPower());
        Log.d(TAG, "max range: " + s.getMaximumRange()
                + ", resolution: " + s.getResolution());
    }
}
```

Determining Device Orientation

- Can use the SensorManager class to determine the orientation of the device

- `getOrientation()` method
  - Two parameters:
    - Rotation matrix
    - An array of three float values (azimuth[z], pitch[x], roll[y])

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Android System Services

System Services

- There is a wide list of services available
  - Power Service
  - KeyGuard Service
  - Vibration Service
  - Alarm Service
  - Sensor Service
  - Audio Service
  - Telephony Service
  - Connectivity Service
  - Wi-Fi Service

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Power Service

- Android runs on limited capabilities devices
- It is crucial to use the battery wisely
- The power service gives us informations about the power of the system
- Get it with:

```
PowerManager pm = (PowerManager)
   getSystemService(Context.POWER_SERVICE);
```

Vibration Service

- Manages the vibration service
- Get it with:

```
Vibrator vibrator =
   (Vibrator)getSystemService(Context.VIBRATOR_S
   ERVICE);
```

- Some methods:
  - vibrate(long time);
  - cancel();
  - vibrate(long[] pattern, int repeat);
- Needs android.permission.VIBRATE

Alarm Service

- Fires an Intent in the future
- Get it with

```
AlarmManager as = (AlarmManager)
    getSystemService(Context.ALARM_SERVICE);
// set(int type, long triggerAtTime, PendingIntent operation);
```

- type is one of:
  - ELAPSED_REALTIME
  - ELAPSED_REALTIME_WAKEUP
  - RTC
  - RTC_WAKEUP

$\left.\begin{array}{l}\text{ELAPSED\_REALTIME}\\ \text{ELAPSED\_REALTIME\_WAKEUP}\end{array}\right\}$ `SystemClock.elapsedRealTime()`

$\left.\begin{array}{l}\text{RTC}\\ \text{RTC\_WAKEUP}\end{array}\right\}$ `System.currentTimeMillis()`

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Alarm Service

- More methods
  - setRepeating(int type, long triggerAtTime, long interval, PendingIntent operation);
    - Can use INTERVAL_HOUR, INTERVAL_HALF_DAY
  - cancel(PendingIntent operation);
    - Match with filterEquals(Intent anotherIntent);

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Telephony Service

- Interacts with calls
- Get it with

```
TelephonyManager tm = (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);
```

- Ask the device about call information
  - getCallState()
  - getDataState()
  - getDataActivity()
  - getNetworkType()
  - isNetworkRoaming()

Connectivity Service

- Check device network state
- Get it with

```
String serId = Context.CONNECTIVITY_SERVICE;
ConnectivityManager cm = (ConnectivityManager) Context.getSystemService(serId);
```

- Check WI-FI, GPRS
- Notify connection changes
- Needs
  - android.permission.ACCESS_NETWORK_STATE
  - android.permission.CHANGE_NETWORK_STATE

Wi-Fi Service

- Manages the Wi-Fi connection
- Get it with

```
WifiManager wfm = (WifiManager) getSystemService(Context.WIFI_SERVICE)
```

- Check Wi-Fi
  - getWifiState()
    - Returns WIFI_STATE_DISABLED, WIFI_STATE_DISABLING, WIFI_STATE_ENABLED, WIFI_STATE_ENABLING, WIFI_STATE_UNKNOWN
  - isWifiEnabled() / setWifiEnabled()
- Lists all the configured wifi connections
  - getConfiguredNetworks()

Wi-Fi Service

- Check/edit wi-fi connection
  - addNetwork(WifiConfiguration config)
  - updateNetwork(WifiConfiguration config)
  - removeNetwork(int netid)
- Scan for wi-fi networks
  - startScan()
- Be notified about wi-fi changes
  - Broadcast Intent: SCAN_RESULTS_AVAILABLE_ACTION
    - Call getScanResults()

Assignment 2

- Posted on Canvas

- Grading:
  - Submitted files
  - In-class demo on physical device
  - (teaching staff will upload your files to a device)

Readings

- Intents and intent filters: http://developer.android.com/guide/components/intents-filters.html
- Android Sensors Overview: http://developer.android.com/guide/topics/sensors/sensors_overview.html
- Android Sensors Framework: http://developer.android.com/guide/topics/sensors/sensors_overview.html#sensors-identify

Thank you

Questions?