# Bluetooth, Wi-Fi, Sensors

# Bluetooth

- **Bluetooth** is a way to exchange data with other devices wirelessly. Android provides Bluetooth API to perform several tasks such as:
- scan bluetooth devices
- connect and transfer data from and to other devices
- manage multiple connections etc.

Add these 2 permissions in the AndroidManifest file :

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

# BluetoothAdapter class

- By the help of **BluetoothAdapter** class, we can perform fundamental tasks such as initiate device discovery, query a list of paired (bonded) devices, create a BluetoothServerSocket instance to listen for connection requests etc.
- BluetoothAdapter class provides many constants. Some of them are as follows:

- String ACTION_REQUEST_ENABLE
- String ACTION_REQUEST_DISCOVERABLE
- String ACTION_DISCOVERY_STARTED
- String ACTION_DISCOVERY_FINISHED

# Methods of BluetoothAdapter class

Commonly used methods of BluetoothAdapter class are as follows:

- **static synchronized BluetoothAdapter getDefaultAdapter()** returns the instance of BluetoothAdapter.
- **boolean enable()** enables the bluetooth adapter if it is disabled.
- **boolean isEnabled()** returns true if the bluetooth adapter is enabled.
- **boolean disable()** disables the bluetooth adapter if it is enabled.
- **String getName()** returns the name of the bluetooth adapter.
- **boolean setName(String name)** changes the bluetooth name.
- **int getState()** returns the current state of the local bluetooth adapter.
- **Set<BluetoothDevice> getBondedDevices()** returns a set of paired (bonded) BluetoothDevice objects.
- **boolean startDiscovery()** starts the discovery process.

# Wi-Fi

- The **android.net.wifi.WifiManager** class can be used to manage the wifi connectivity. It can be used to add network, disable network, scan for access points, disconnect etc.
- Android allows applications to access the state of the wireless connections at very low level. Application can access almost all the information of a wifi connection.
- Add these 2 permissions in the AndroidManifest file :

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

# Wi-Fi

- We can instantiate this class by calling getSystemService method. Its syntax is given below :

```
WifiManager mainWifiObj;
mainWifiObj = (WifiManager) getSystemService(Context.WIFI_SERVICE);
```

- The wifi scan can be started by calling the startScan method of the WifiManager class. This method returns a list of ScanResult objects. You can access any object by calling the get method of list. Its syntax is given below −

```
List<ScanResult> wifiScanList = mainWifiObj.getScanResults();
String data = wifiScanList.get(0).toString();
```

# Sensors

- You all must have played some Android games that includes the supports of sensors i.e. by tilting the phone some actions might happen in the game. For example, in the Temple Run game, by tilting the phone to left or right, the position of the runner changes.
- So, all these games are using the sensors present in your Android device. Other examples can be shaking your phone to lock the screen, finding the direction with the help of a compass, etc. All these are examples of Android sensors
- Sensors can be used to monitor the three-dimensional device movement or change in the environment of the device.
- Android provides sensor api to work with different types of sensors.

# Sensors

Android supports three types of sensors:

**1) Motion Sensors**

These are used to measure acceleration forces and rotational forces along with three axes. This category includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors.

**2) Position Sensors**

These are used to measure the physical position of device. This category includes orientation sensors and magnetometers.

**3) Environmental Sensors**

These are used to measure the environmental changes such as temperature, humidity etc. This category includes barometers, photometers, and thermometers.

# Android Sensor API

**1) SensorManager class**

The android.hardware.SensorManager class provides methods to get sensor instance, to access and list sensors,to register and unregister sensor listeners etc.

You can get the instance of SensorManager by calling the method getSystemService() and passing the SENSOR_SERVICE constant in it.

**SensorManager sm = (SensorManager)getSystemService(SENSOR_SERVICE);**

The next thing you need to do is to instantiate the object of Sensor class by calling the getDefaultSensor() method of the SensorManager class. Its syntax is given below −

**SensorManager sMgr;**

**sMgr = (SensorManager)this.getSystemService(Context.SENSOR_SERVICE);**

**Sensor light;**

**light = sMgr.getDefaultSensor(Sensor.TYPE_LIGHT);**

# Android Sensor API

**2) Sensor class**

The android.hardware.Sensor class provides methods to get information of the sensor such as sensor name, sensor type, sensor resolution, sensor type etc.

**3) SensorEvent class**

Its instance is created by the system. It provides information about the sensor.

**4) SensorEventListener interface**

It provides two call back methods to get information when sensor values (x,y and z) change or sensor accuracy changes.

**void onAccuracyChanged(Sensor sensor, int accuracy)** - it is called when sensor accuracy is changed.

**void onSensorChanged(SensorEvent event)** - it is called when sensor values are changed.

# Getting list of sensors supported

You can get a list of sensors supported by your device by calling the getSensorList method, which will return a list of sensors containing their name and version number and much more information. You can then iterate the list to get the information. Its syntax is given below −

```
sMgr = (SensorManager)this.getSystemService(SENSOR_SERVICE);
List<Sensor> list = sMgr.getSensorList(Sensor.TYPE_ALL);

for(Sensor sensor: list){

}
```

# Android Sensors Types

- **Photometer** is used to sense and control the brightness. Besides, there are sensors for pressure, humidity, and temperature.
- For movements, **Accelerometer** is used to detect shakes/tilt gestures.
- **Proximity Sensors** are used to detect how close the object is to the device. It's commonly present in Call Applications. As you bring the phone close to the ear, the screen goes black, thanks to this sensor. Though the maximum range of proximity is 5 cms.
- **Gyroscope** is used to measure rotation/spin. Gravity sensors are used to measure the force of gravity.
- **Magneto Meter** is used to get the device position.
- **Pedometer** is used to detect the number of steps the user takes.

https://developer.android.com/guide/topics/sensors/sensors_overview