Permissions and Sensors

Chapter 8: Exploring Android App Permissions:

Permissions

- **Permissions** on a device control what your app can do and what it can’t.
- They keep user data safe and stop apps from doing things they shouldn’t.
- Some permissions allow access to data and actions because they pose very little risk to the user's privacy or the operation of other apps.
  - Such as **INTERNET**, **VIBRATE**
- To get permissions, you need to declared in AndroidManifest.xml and granted automatically.

Permissions

- While some other permissions grant access to user data or control over the device that could affect the user's privacy or the operation of other apps.
  - Such as **ACCESS_FINE_LOCATION**, **READ_CONTACTS**, **CAMERA**
- To get such permissions, you also need to declare in AndroidManifest.xml and must be requested at runtime.

1: Checking if the device is connected to the WIFI

- **1. Add the necessary permissions** to your AndroidManifest.xml file:

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

- **2. Create a new Java class** named WifiManagerUtil.java:

```
import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;

public class WifiManagerUtil {
// Method to check if the device is connected to the internet via WiFi
public static boolean isConnectedToWifi(Context context) {
ConnectivityManager connectivityManager = (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo wifiNetwork = connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
return wifiNetwork != null && wifiNetwork.isConnected();
}
}
```

Is a general-purpose method that provides access to various system-level services in Android

constant that specifies which service we want to retrieve. specify that you want the connectivity service.

Is class responsible for managing network connections, (Wi-Fi, mobile data, VPN), It contains several methods to check the network status, details about active networks, or manage network connections.

Note: the type Context, we normally used to access system services and resources.

1: Checking if the device is connected to the WIFI

- **3. Modify your MainActivity.java file**:

```
import android.os.Bundle;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

// Check if device is connected to WiFi
if (WifiManagerUtil.isConnectedToWifi(this)) {
showToast("Connected to WiFi");
} else {
showToast("Not connected to WiFi");
}
}

// Method to show toast message
private void showToast(String message) {
Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
}
}
```

```
package com.example.wifistatuscheck;

import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;

1 usage
public class WifiManagerUtil {
    // Method to check if the device is connected to the internet via WiFi
    1 usage
    public static boolean isConnectedToWifi(Context context) {
        ConnectivityManager connectivityManager = (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo wifiNetwork = connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
        return wifiNetwork != null && wifiNetwork.isConnected();
    }
}
```

1: The details of the above code

Importing necessary classes from the Android framew

```
import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
```

Declares a new Java class this case ‘WifiManagerUt

```
public class WifiManagerUtil {
```

Declare a method ‘isConnectedToWifi’ which takes a Context object as a parameter and returns a boolean value (true if connected to Wi-Fi, false otherwise).

```
f the device is connected to the internet via WiFi
1 usage
public static boolean isConnectedToWifi(Context context) {
```

This retrieves the system service for connectivity (ConnectivityManager) using the provided Contex

```
ConnectivityManager connectivityManager = (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
```

This gets information about the network connected to the device. Specifically, it retrieves information about the Wi-Fi net

```
NetworkInfo wifiNetwork = connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
```

This checks if wifiNetwork is not null and if it's connected. It then returns true if connected, false otherwise

```
return wifiNetwork != null && wifiNetwork.isConnected();
```

2: Check if internet available or not

AndroidManifest.xml This informs that the app requires permission to access the internet.

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

MainActivity.java add this inside a setOnClickListener method

```
if (isNetworkAvailable()) {
    Toast.makeText(MainActivity.this, "Internet Connection Available", Toast.LENGTH_SHORT).show();
} else {
    Toast.makeText(MainActivity.this, "No Internet Connection", Toast.LENGTH_SHORT).show();
}
```

Declare a method '**isNetworkAvailable**()' which returns a boolean value.

```
// method for network avilability
private boolean isNetworkAvailable() {
```

initialize the ConnectivityManager object, allowing the app to check the device's network connectivity.

```
    ConnectivityManager connectivityManager = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
```

retrieves information about the device's active network connection.

```
    NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo();

    return activeNetworkInfo != null && activeNetworkInfo.isConnected();
}
```

The variable **activeNetworkInfo** contains an object of type **NetworkInfo**. It (object) represents information about the current active network connection on the device.
It can include details such as the type of network (e.g., WIFI, mobile data), whether the device is connected to a network.

If both conditions are true, the method returns true, indicating that the network is available. Otherwise, it returns false.

check if there is any active network info available.

Is network is actually connected

3: Bluetooth permissions

AndroidManifest.xml

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

Declares the permission to access Bluetooth functionality. It allows the app to communicate with Bluetooth devices.

Allows to perform Bluetooth administrative tasks, such as enabling/disabling Bluetooth, pairing devices, etc.

Allows the permission to perform Bluetooth device discovery.

Allows the permission to connect to Bluetooth devices.

MainActivity.java add this inside a setOnClickListener method

```
// Check if Bluetooth Permission is Granted
boolean isBluetoothPermissionGranted = ContextCompat.checkSelfPermission(MainActivity.this, Manifest.permission.BLUETOOTH) == PackageManager.PERMISSION_GRANTED;

// Show Toast based on Permission Status
if (isBluetoothPermissionGranted) {
    Toast.makeText(MainActivity.this, "Bluetooth Permission Granted", Toast.LENGTH_SHORT).show();
} else {
    Toast.makeText(MainActivity.this, "Bluetooth Permission Denied", Toast.LENGTH_SHORT).show();
}
```

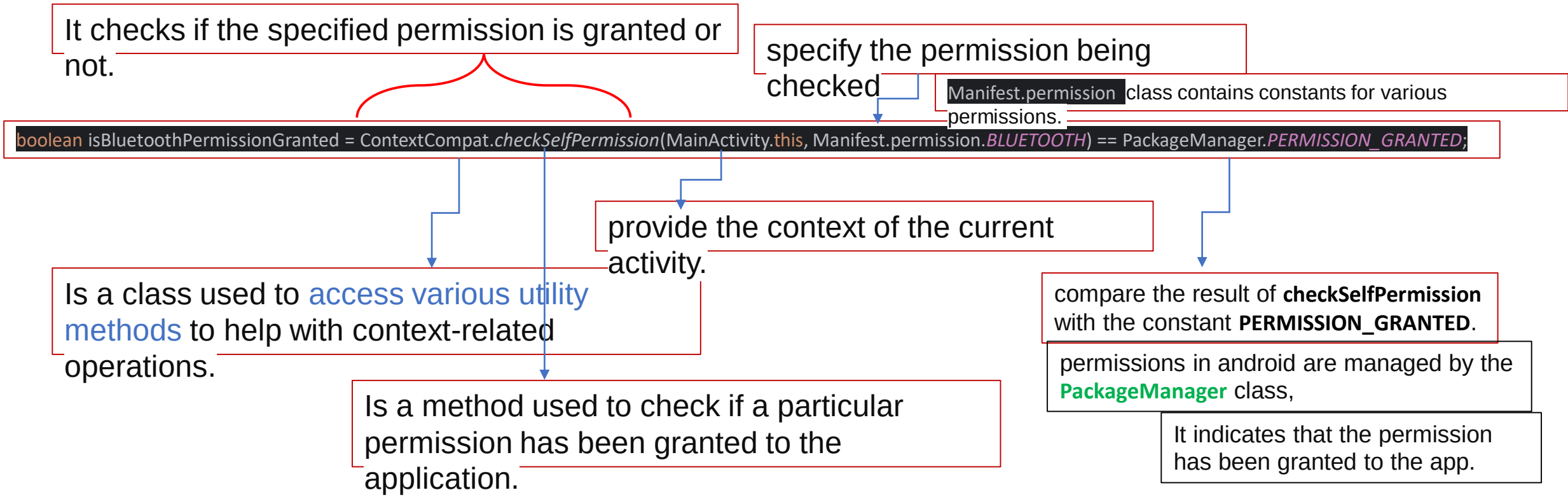
permissions in android are managed by the **PackageManager** class,

It indicates that the permission has been granted to the app.

PackageManager.PERMISSION_DENIED;

This checks if the app has been granted the Bluetooth permission (**android.permission.BLUETOOTH**) by the user.
It uses **ContextCompat.checkSelfPermission**() method to **check the permission status**. If the permission is granted, **isBluetoothPermissionGranted** will be true; otherwise, it will be false.

3: Bluetooth permissions



4: Check if the device has a vibrator

AndroidManifest.xml

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

•To access the device's vibrator service, we need this to declare, so that the app gives us permission to use the device's vibration functionality.
•Without this permission, the app cannot access the device's vibrator.

MainActivity.java add this inside a setOnClickListener method

object can be used to control the device's vibration functionality.

This is a method that returns a system-level service by name, in this case, the vibrator service.

request the vibrator service.

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
```

Is a method that returns true if the device has a vibrator, otherwise, it returns false.

```
if (vibrator.hasVibrator()) {
    Toast.makeText(MainActivity.this, "Vibrator Available", Toast.LENGTH_SHORT).show();
    // Vibrate for 500 milliseconds
    vibrator.vibrate(500);

} else {
    Toast.makeText(MainActivity.this, "Vibrator Not Available", Toast.LENGTH_SHORT).show();
}
```

Roles of Context and getSystemService

- The **getSystemService** method and the **Context** class play different roles but somehow interrelated.
- **Context roles** is to provides access to application-specific resources and classes, as well as information about the application environment.
- **Some of the services and resources he can access include**:
  - **Access Application Resources**: Retrieve resources, databases, and preferences.
  - **Start Activities and Services**: Launch activities, start and stop services.
  - **Send Broadcasts**: Send and receive broadcast messages across the system.
  - **Access System Services**: Obtain references to system-level services.

- The **getSystemService** method: is a method provided by the Context class that allows you to retrieve references to various system-level services.
  - **Retrieve System Services** such as connectivity, location, sensor

5: Checking for Accelerometer Availability

Giving the app the necessary permissions to access sensor data

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

Accelerometer, Gyroscope, Pedometer, Biometric Sensors and many more are sensors that are part of the broader category of **Body Sensors**.

We use this method to get the SensorManager system service, which allows access to the device's sensors

This **constant** is used to retrieve a reference to the SensorManager service.

gets the default accelerometer sensor.

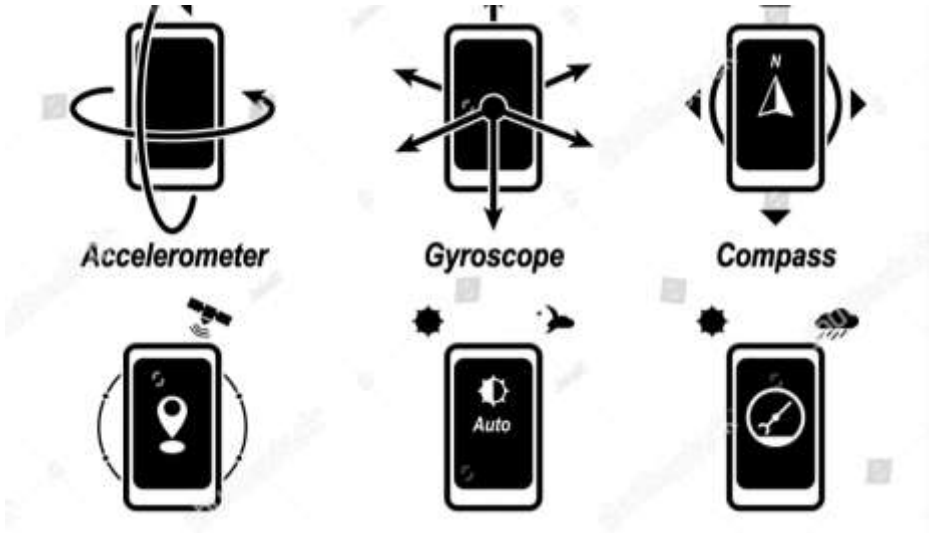
```
// Get the SensorManager service
SensorManager sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);

// Check if the accelerometer is available
if (sensorManager != null) {
    Sensor accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

    if (accelerometer != null) {
        Toast.makeText(MainActivity.this, "Accelerometer Available", Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(MainActivity.this, "Accelerometer Not Available", Toast.LENGTH_SHORT).show();
    }
}
```

Other Service types you can check

TYPE_GYROSCOPE
TYPE_MAGNETIC_FIELD
TYPE_LIGHT
TYPE_PRESSURE

Sensors

- **TYPE_ACCELEROMETER**
  - **Accelerometer** measures the acceleration applied to the device. Accelerometer helps confirm how you're holding your phone.
    - **Detecting the orientation of the device** (e.g., **landscape** or **portrait** mode).
    - **Motion detection** (e.g., shaking the device).
    - **Step counting in fitness applications**.
    - **Detecting free-fall scenarios**.
- **TYPE_GYROSCOPE**
  - A gyroscope measures the rate of rotation around the device's three primary axes (x, y, and z).
    - **Enhancing the accuracy of the accelerometer** by providing rotational motion data.
    - **Advanced gesture recognition**.
    - **Improving the orientation tracking for augmented reality applications**.
    - **Stabilizing the camera during video recording**.

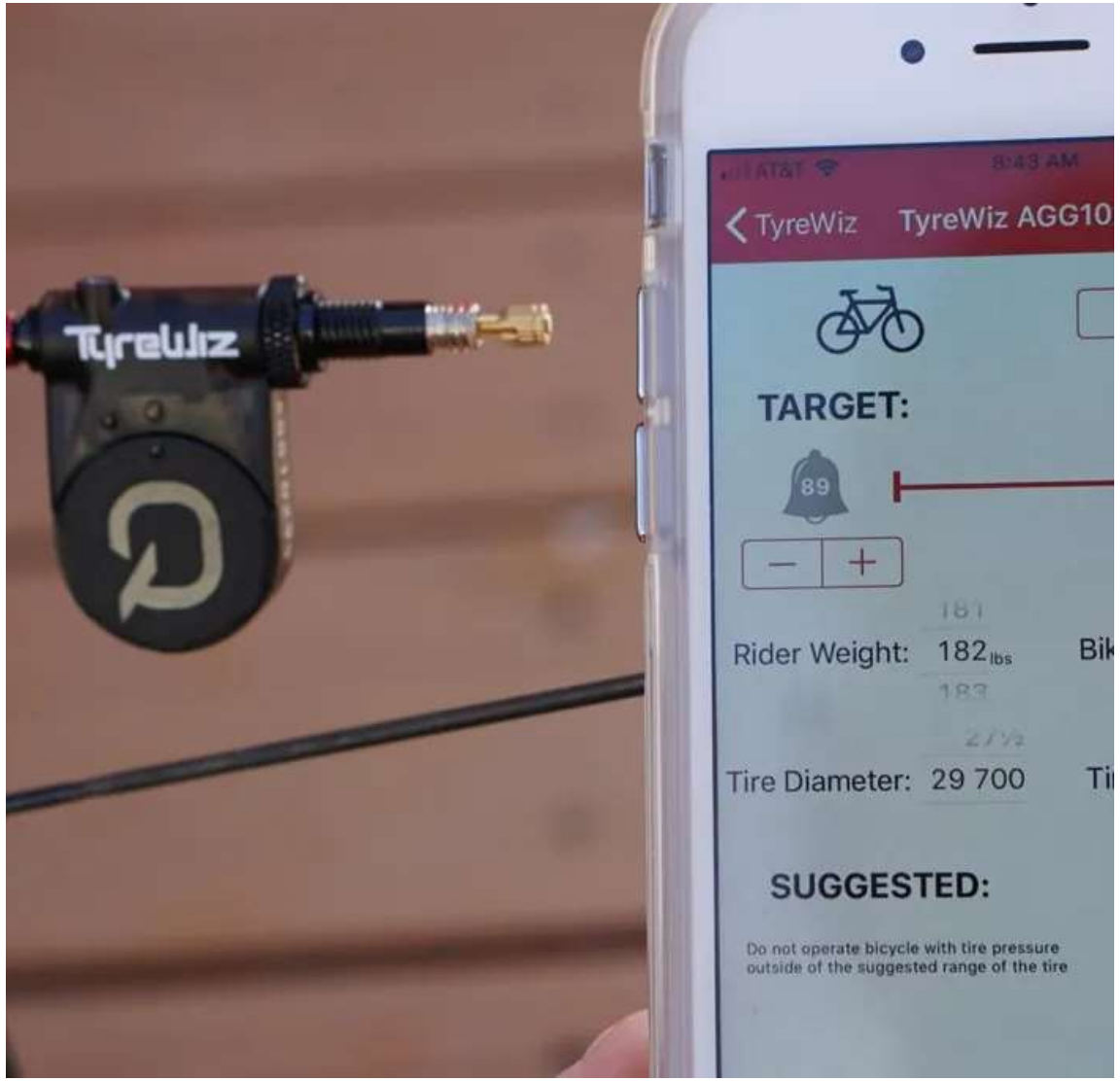


Sensors

- **TYPE_MAGNETIC_FIELD**
  - A magnetic field sensor and it helps your phone act like a compass, showing you which way is north.
    - Implementing a **digital compass**.
    - Providing **orientation data** when combined with **data from the accelerometer**.
    - **Augmented reality applications** that require precise orientation information.
    - **Navigation applications** for determining direction.
- **TYPE_LIGHT**
  - A **light sensor** measures the ambient light level (illumination) in lux.
    - **Adjusting the screen brightness automatically**.
    - **Conserving battery life** by dimming the screen in low light conditions.
    - **Enhancing user experience** by adapting to the lighting environment.
    - **Detecting if the device is in a pocket** or bag.

Sensors

- **TYPE_PRESSURE**
  - This sensor **measures the air pressure** around your device. It reports the pressure in units called hectopascals (hPa) or millibars (mBar).
    - Determining **altitude or elevation changes** for location-based applications (e.g., GPS navigation).
    - **Weather forecasting applications:** By measuring changes in air pressure, it help weather apps predict changes in the weather, like **when a storm is coming**.
    - **Fitness applications:** Measure how much you go up and down during activities like hiking or climbing.
    - **Enhancing indoor navigation** by identifying different floors in a building.



Any Question?