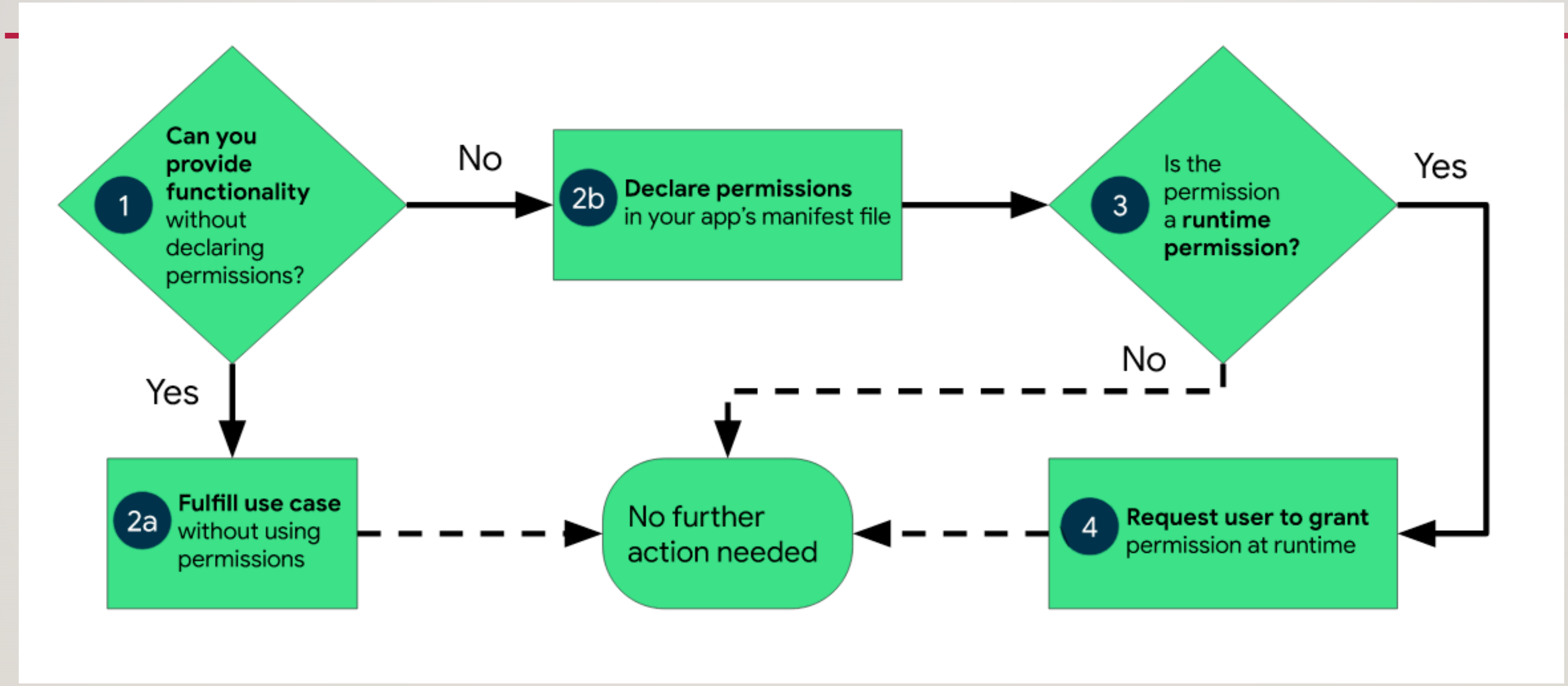
PERMISSIONS ON ANDROID

APP PERMISSIONS

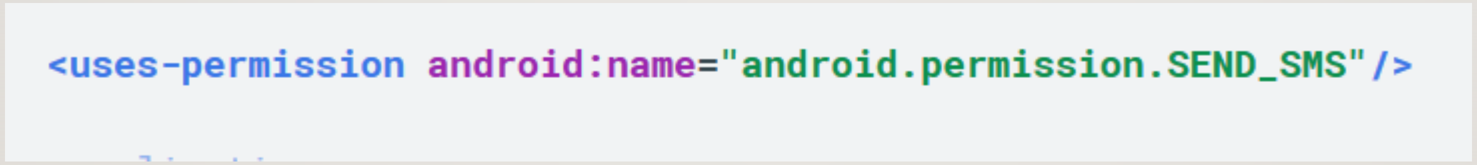
- help support user privacy by protecting access to the following:
  - **Restricted data**, such as system state and a user's contact information.
  - **Restricted actions**, such as connecting to a paired device and recording audio.

WORKFLOW FOR USING APP PERMISSIONS:


1
Can you provide functionality without declaring permissions?
No
2b
Declare permissions in your app's manifest file
3
Is the permission a runtime permission?
Yes
Yes
No
2a
Fulfill use case without using permissions
No further action needed
4
Request user to grant permission at runtime

Src: https://developer.android.com/guide/topics/permissions/overview

- to protect the privacy of an Android user.
- No permissions granted by default (the app must request them)
- Using manifest file, the app must request permission to access sensitive user data and system features e.g. Contacts, SMS, camera ,internet etc.



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

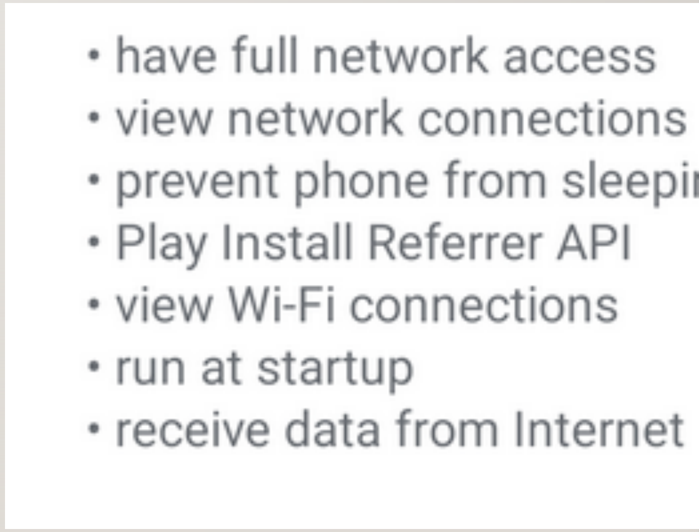
- Before API 23 the requested permissions are presented to the user before installing the application.
- Starting from Android 6.0(Marshmallow API 23 or higher)
  - Protection levels and run time permissions are interdicted
  - Depending on the feature, the system might grant the permission automatically or might prompt the user to approve the request.

TYPES OF PERMISSIONS

- Android categorizes permissions into different types,
- Install-time permissions,
- Runtime permissions
- Special permissions.

INSTALL-TIME PERMISSIONS

- Install-time permissions give your app limited access to restricted data, and they allow your app to perform restricted actions that minimally affect the system
- install-time permissions, include
  - normal permissions
  - signature permission



- have full network access
- view network connections
- prevent phone from sleepi
- Play Install Referrer API
- view Wi-Fi connections
- run at startup
- receive data from Internet

RUNTIME PERMISSIONS,

- Runtime permissions access *private user data*, a special type of restricted data that includes potentially sensitive information.
- Also known as dangerous permissions, give the app additional access to restricted data, allow your app to perform restricted actions that more substantially affect the system

- Android Permissions are divided into several protection levels
- **Normal permissions**

  INTERNET ,VIBRATE, SET_ALARM ,ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE

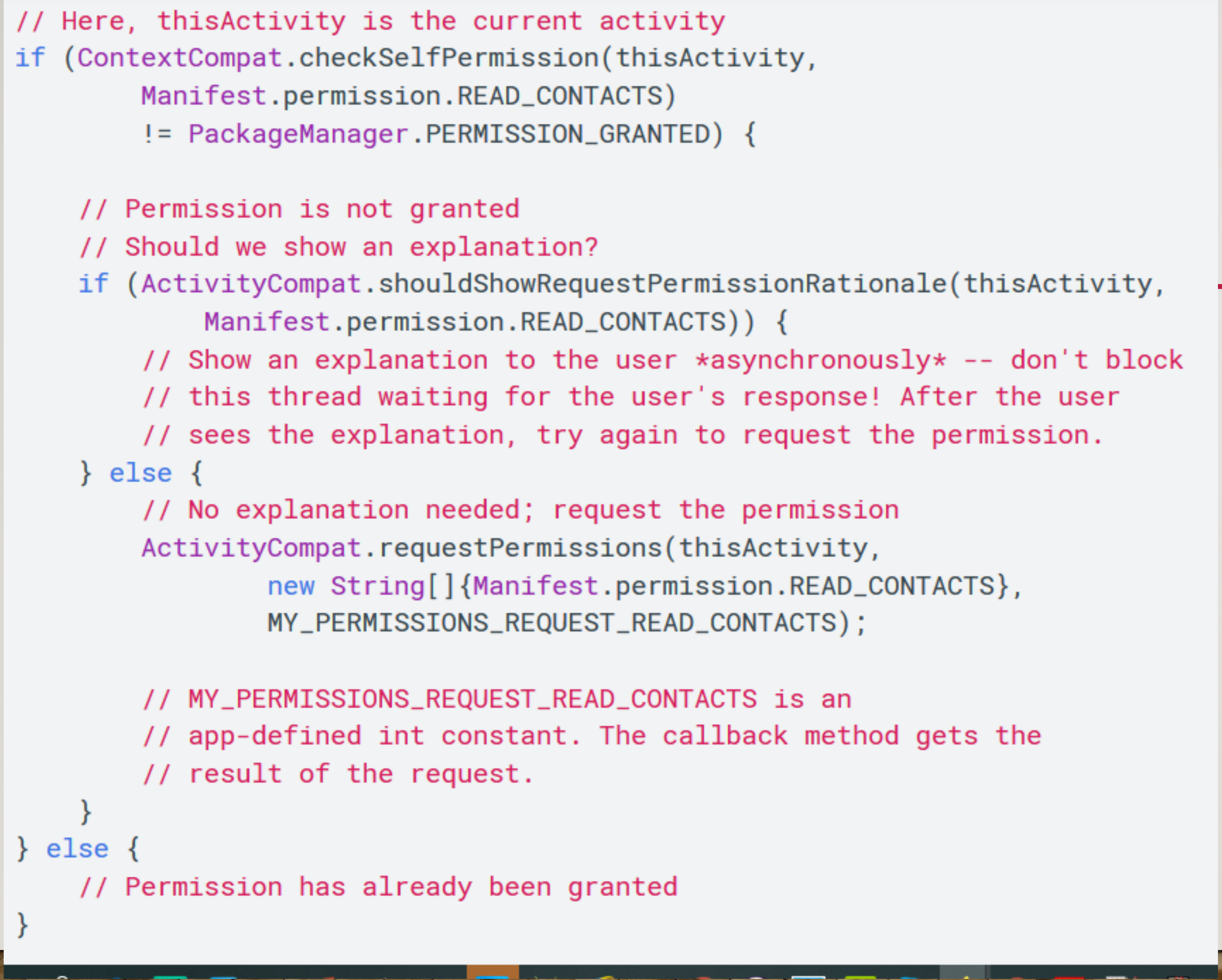
  The system automatically grants the app that permission at install time.
- **Signature permissions**
  - The system grants these app permissions at install time
  - The app that attempts to use a permission is signed by the same certificate as the app that defines the permission.
- **Dangerous permissions**
  - Access to data or resources that involve the user's private information contacts , camera , location
  - the user has to explicitly grant the permission to the app
  - pp must prompt the user to grant permission at runtime

BEST PRACTICES

- **Control:** The user has control over the data that they share with apps.
- **Transparency:** The user understands what data an app uses
- **Data minimization:** An app accesses and uses only the data that's required

REQUEST APP PERMISSIONS

- Declare the permission in the app manifest
- if the permission need user explicit approval (dangerous permission) ask the user to approve each permission at runtime (on Android 6.0 and higher)



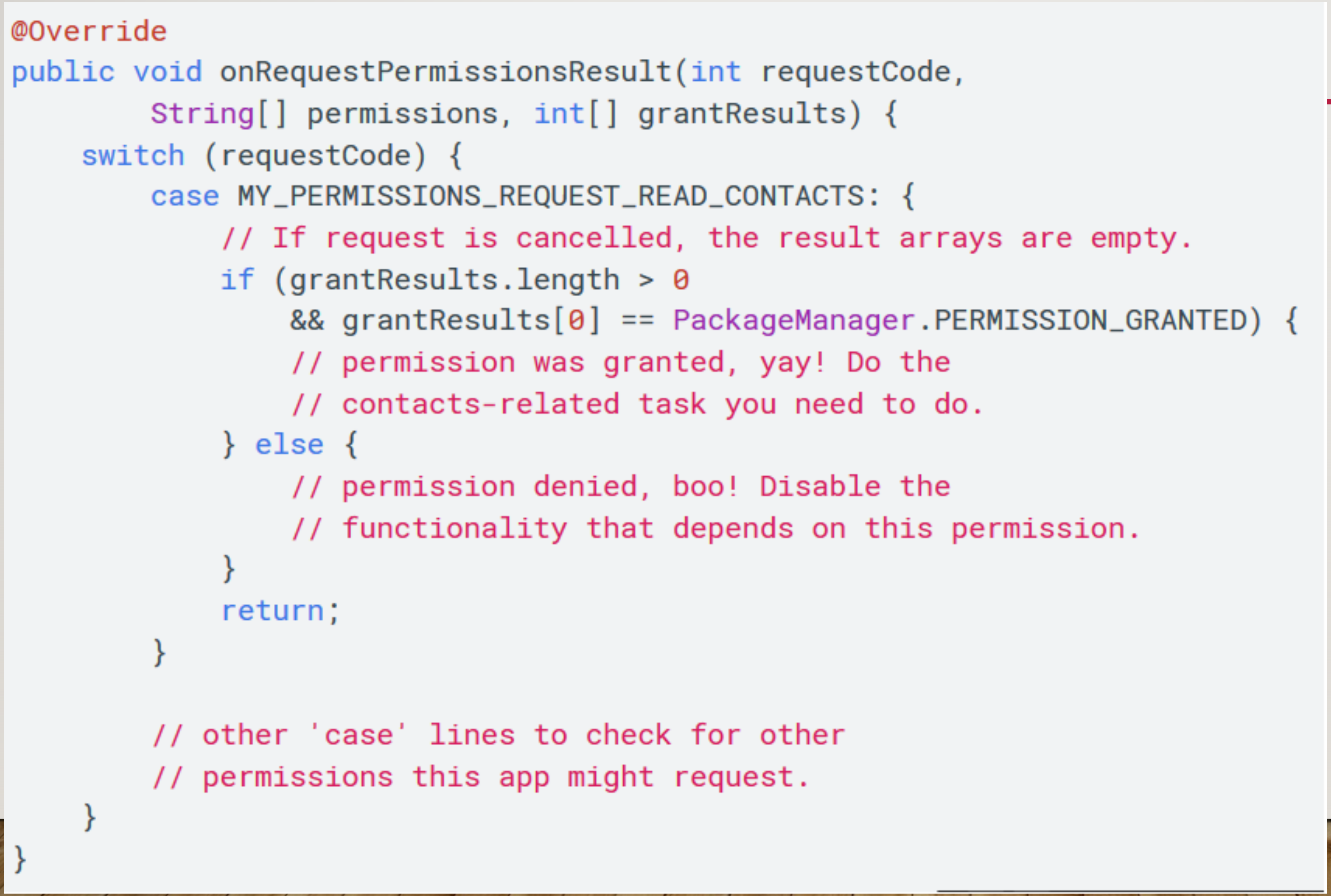
```
// Here, thisActivity is the current activity
if (ContextCompat.checkSelfPermission(thisActivity,
        Manifest.permission.READ_CONTACTS)
        != PackageManager.PERMISSION_GRANTED) {

    // Permission is not granted
    // Should we show an explanation?
    if (ActivityCompat.shouldShowRequestPermissionRationale(thisActivity,
            Manifest.permission.READ_CONTACTS)) {
        // Show an explanation to the user *asynchronously* -- don't block
        // this thread waiting for the user's response! After the user
        // sees the explanation, try again to request the permission.
    } else {
        // No explanation needed; request the permission
        ActivityCompat.requestPermissions(thisActivity,
                new String[]{Manifest.permission.READ_CONTACTS},
                MY_PERMISSIONS_REQUEST_READ_CONTACTS);

        // MY_PERMISSIONS_REQUEST_READ_CONTACTS is an
        // app-defined int constant. The callback method gets the
        // result of the request.
    }
} else {
    // Permission has already been granted
}
```

Src: https://developer.android.com/training/permissions/requesting

OVERRIDE THIS METHOD TO HANDLE THE PERMISSIONS REQUEST RESPONSE



```java
@Override
public void onRequestPermissionsResult(int requestCode,
        String[] permissions, int[] grantResults) {
    switch (requestCode) {
        case MY_PERMISSIONS_REQUEST_READ_CONTACTS: {
            // If request is cancelled, the result arrays are empty.
            if (grantResults.length > 0
                && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                // permission was granted, yay! Do the
                // contacts-related task you need to do.
            } else {
                // permission denied, boo! Disable the
                // functionality that depends on this permission.
            }
            return;
        }

        // other 'case' lines to check for other
        // permissions this app might request.
    }
}
```

Src: https://developer.android.com/training/permissions/requesting

EXAMPLE:

- Location Permissions
- android.permission.ACCESS_COARSE_LOCATION
- android.permission.ACCESS_FINE_LOCATION

LOCATION API IN ANDROID

- Android provides a location API which contain a number of important classes and interface.
  - **<u>LocationManager</u>** : – to get access to the location service of the system.
    <u>Location</u> :- A class that represents the geographic location
  - **<u>LocationListener</u>** :- listener which receives notification when the location changes or the location provider is disabled or enabled.

EXAMPLES REQUESTING PERMISSION AT RUNTIME

```
locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
    // Define the criteria how to select the locatioin provider -> use // default
    Criteria criteria = new Criteria();
    //criteria.
    String provider = locationManager.getBestProvider(criteria, false);

    if (checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION) !=
            PackageManager.PERMISSION_GRANTED
            && checkSelfPermission(Manifest.permission.ACCESS_COARSE_LOCATION) !=
                    PackageManager.PERMISSION_GRANTED) {

        Log.d("Loaction" , " permission required");
        requestPermission();
        return;
    }else {

        locationManager.requestLocationUpdates(provider, 400, 1, this);
    }
}
```

Min Time msec

Listener

Miin Distance (m)

- public class MapsActivity extends Activity implements LocationListener {

```
@Override  // call back method
public void onLocationChanged(Location location) {

  // new location

}

@Override
public void onStatusChanged(String s, int i, Bundle bundle) {

}

@Override
public void onProviderEnabled(String s) {

// GPS or Network Provider enabled

}
```

REFERENCES

- https://developer.android.com/guide/topics/permissions/overview
- https://developer.android.com/guide/topics/permissions/overview#normal
- https://developer.android.com/guide/topics/permissions/overview#runtime
- https://developer.android.com/reference/android/location/Location