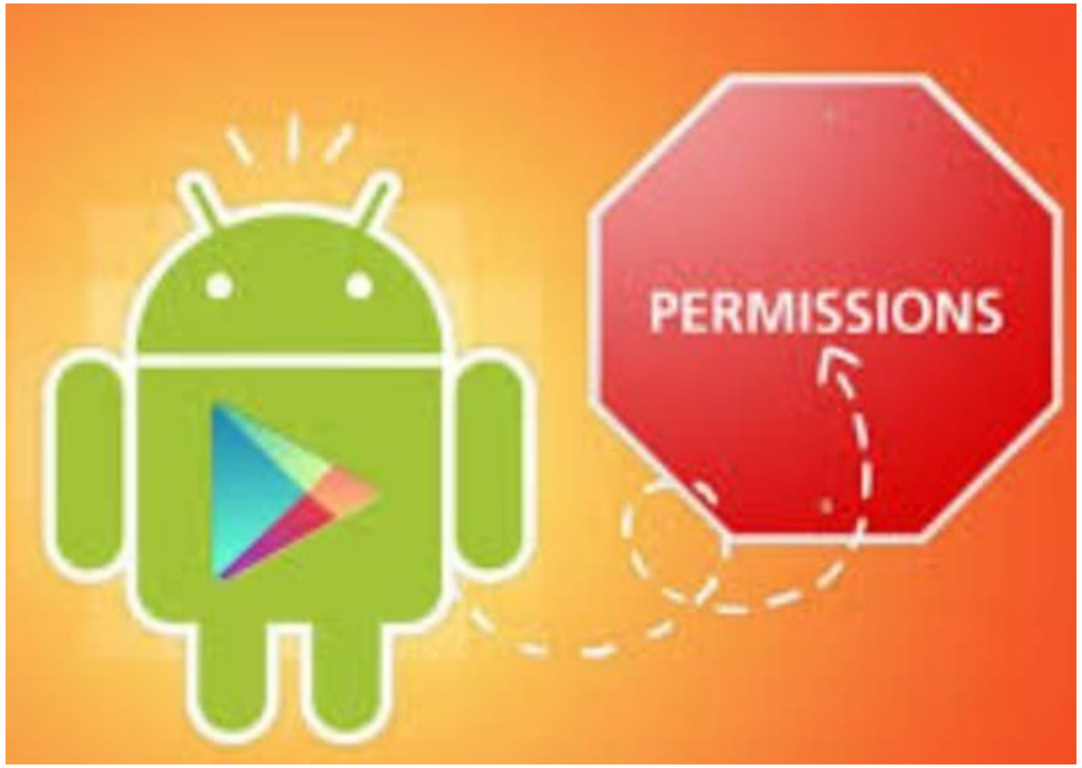
ANDROID PERMISSIONS

Part 1

INTRODUCTION

- In an attempt to maintain security (system integrity, user privacy) on Android, devices run each app in a limited sandbox .
- Android's app permissions build upon a central design point of the Android security architecture.
- By default, no app has permission to perform any operations that would adversely impact other apps, the operating system, or the user.
- If app wants to use resources (e.g. camera, storage, network) or information (e.g. contacts info) outside of sandbox, must request permission.



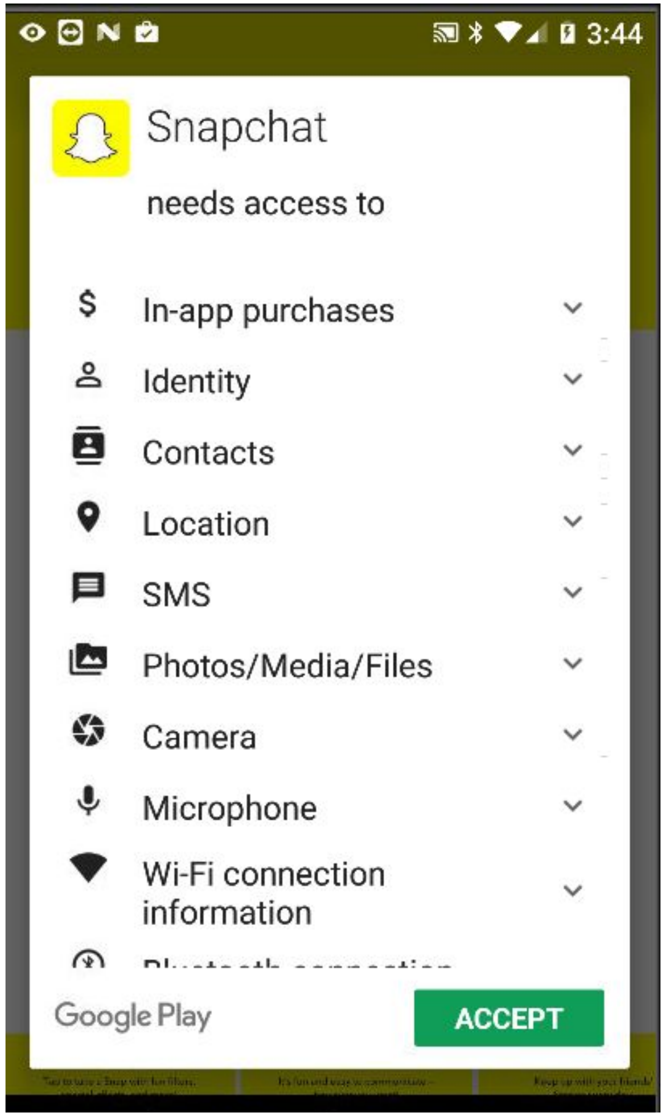
HISTORY OF PERMISSIONS

- Prior to Android version 6.0, Marshmallow (API 23)
  - Apps requested permission at **install time** (during installing the app from app store)
  - Permissions were Listed in the AndroidManifest.xml File

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

Install Time Permissions

- Pre 6.0 / Marshmallow user granted permission at *install time.*
- Google Play listed permissions for app when install was selected



3:44

Snapchat

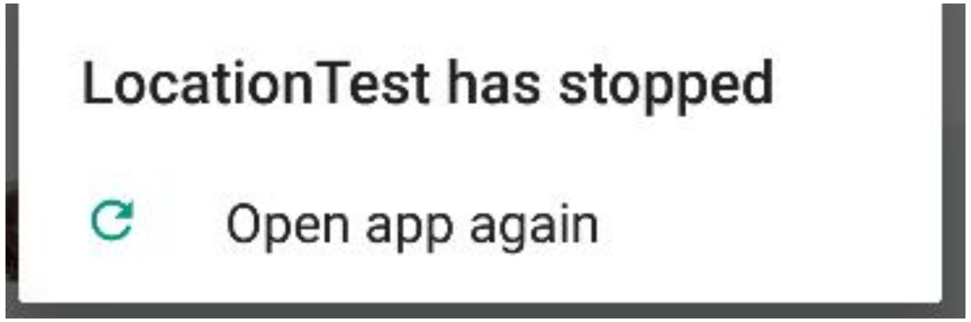
needs access to

- In-app purchases
- Identity
- Contacts
- Location
- SMS
- Photos/Media/Files
- Camera
- Microphone
- Wi-Fi connection information

Lacks Permissions (Install time)

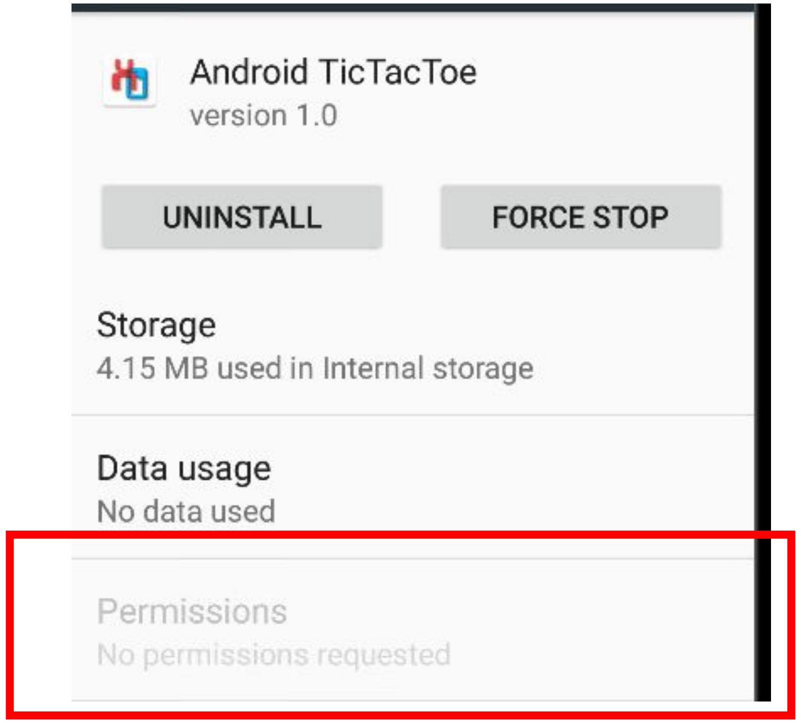
- App ***stopped*** by system as soon as it tries to perform operation it doesn't have permission for…
- Exception:

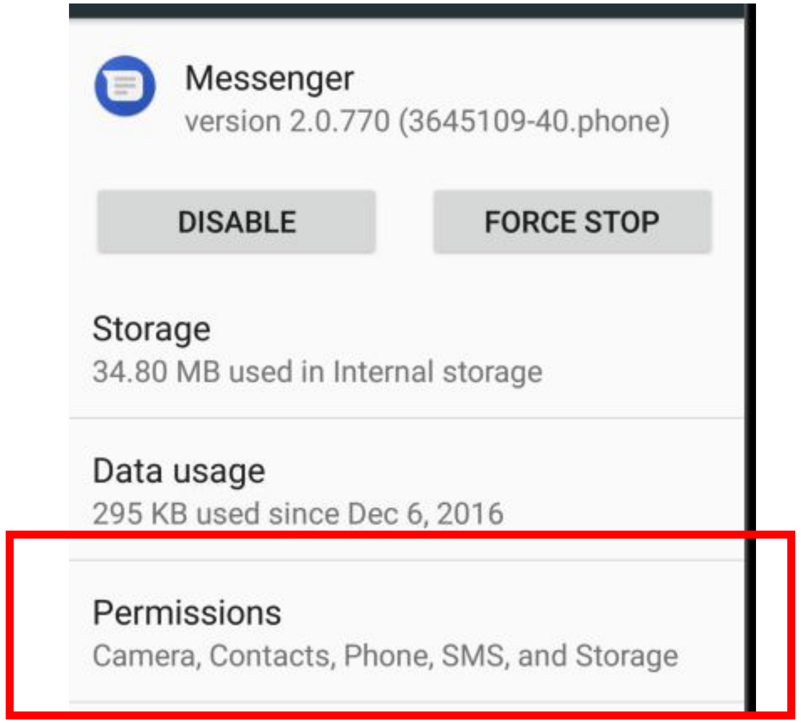
```
02-09 16:03:39.857 26846-26846/org.example.locationtest
E/AndroidRuntime: FATAL EXCEPTION: main
Process: org.example.locationtest, PID: 26846
java.lang.RuntimeException: Unable to start activity
ComponentInfo
{org.example.locationtest/org.example.locationtest.LocationTest}:
java.lang.SecurityException: "passive" location provider requires
ACCESS_FINE_LOCATION permission.
at android.location.LocationManager.getProvider(LocationManager.java:383)
at
org.example.locationtest.LocationTest.dumpProvider(LocationTest.java:372)
at
org.example.locationtest.LocationTest.dumpProviders(LocationTest.java:364)
at org.example.locationtest.LocationTest.onCreate(LocationTest.java:80)
```


LocationTest has stopped

VIEWING PERMISSIONS

- A user can see what permissions an app has in Settings -> Apps


Android TicTacToe
version 1.0
UNINSTALL
FORCE STOP
Storage
4.15 MB used in Internal storage
Data usage
No data used
Permissions
No permissions requested

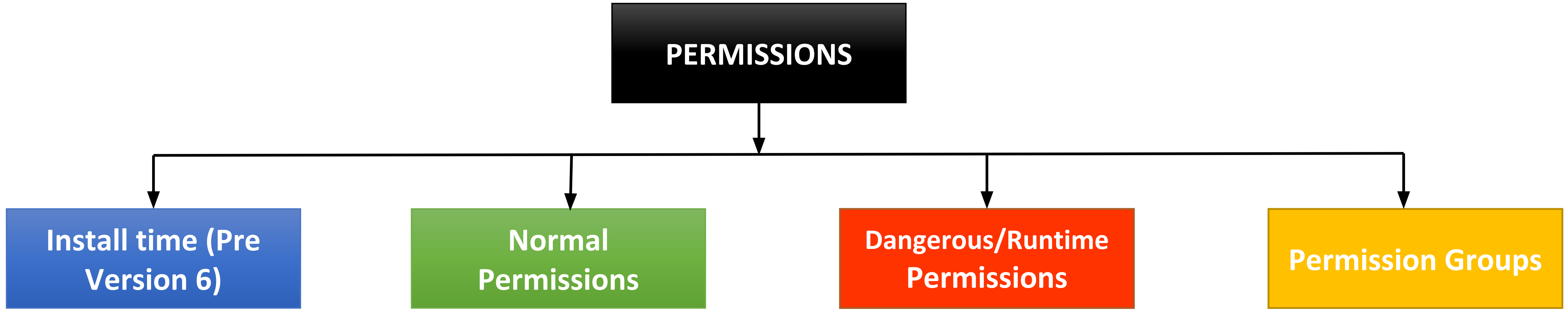

Messenger
version 2.0.770 (3645109-40.phone)
DISABLE
FORCE STOP
Storage
34.80 MB used in Internal storage
Data usage
295 KB used since Dec 6, 2016
Permissions
Camera, Contacts, Phone, SMS, and Storage

- Developers can see device permissions via command line:

```
$ adb shell pm list permissions -s
```

Changes to Permissions

- Android 6.0 / Marshmallow introduced changes to permissions
- Types of Permissions



Normal Permissions

- **Normal Permissions:** These permissions allow access to data and actions that extend beyond your app's sandbox. However, the data and actions present very little risk to the user's privacy, and the operation of other apps. They fall under Install time Permissions.

- ACCESS_LOCATION_EXTRA_COMMANDS
- ACCESS_NETWORK_STATE
- ACCESS_NOTIFICATION_POLICY
- ACCESS_WIFI_STATE
- **BLUETOOTH**
- BLUETOOTH_ADMIN
- BROADCAST_STICKY
- CHANGE_NETWORK_STATE
- CHANGE_WIFI_MULTICAST_STATE
- CHANGE_WIFI_STATE
- DISABLE_KEYGUARD
- EXPAND_STATUS_BAR
- GET_PACKAGE_SIZE
- INSTALL_SHORTCUT
- **INTERNET**
- KILL_BACKGROUND_PROCESSES
- MODIFY_AUDIO_SETTINGS
- NFC

- READ_SYNC_SETTINGS
- READ_SYNC_STATS
- RECEIVE_BOOT_COMPLETED
- REORDER_TASKS
- REQUEST_IGNORE_BATTERY_OPTIMIZATIONS
- REQUEST_INSTALL_PACKAGES
- **SET_ALARM**
- SET_TIME_ZONE
- SET_WALLPAPER
- SET_WALLPAPER_HINTS
- TRANSMIT_IR
- UNINSTALL_SHORTCUT
- USE_FINGERPRINT
- **VIBRATE**
- **WAKE_LOCK**
- WRITE_SYNC_SETTINGS

Dangerous/Runtime Permissions

- **Dangerous/Runtime Permissions:** Runtime permissions, also known as dangerous permissions, give your app additional access to restricted data, and they allow your app to perform restricted actions that more substantially affect the system and other apps. Therefore, you need to request runtime permissions in your app before you can access the restricted data or perform restricted actions.

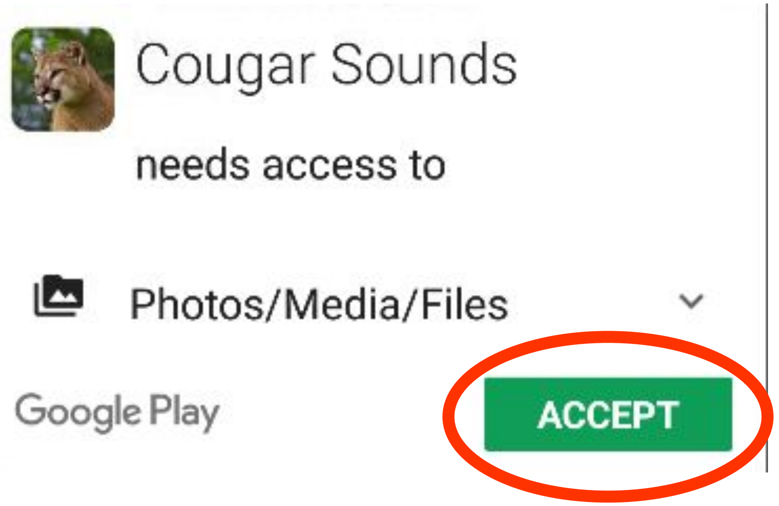
- Dangerous Permissions are listed below:
  • **CALENDAR**: READ_CALENDAR, WRITE_CALENDAR
  • **CAMERA**: CAMERA
  • **CONTACTS**: READ_CONTACTS, WRITE_CONTACTS, GET_ACCOUNTS
  • **LOCATION**: ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION
  • **MICROPHONE**: RECORD_AUDIO
  • **PHONE**: READ_PHONE_STATE, CALL_PHONE, READ_CALL_LOG, WRITE_CALL_LOG, ADD_VOICEMAIL, USE_SIP PROCESS_OUTGOING_CALLS
  • **SENSORS**: BODY_SENSORS
  • **SMS**: SEND_SMS, RECEIVE_SMS, READ_SMS, RECEIVE_WAP_PUSH, RECEIVE_MMS
  • **STORAGE**: READ_EXTERNAL_STORAGE, WRITE_EXTERNAL_STORAGE

Permissions Groups

- If several permissions relate to a similar capability or feature of an app or device, the system might represent these permissions as a *permissions group*.
- These permission groups help the system minimize the number of system dialogs that are presented to the user when an app requests closely-related permissions.
- Dangerous Permissions Organized into Permission Groups:
  • **CALENDAR**: READ_CALENDAR, WRITE_CALENDAR
  • **CONTACTS**: READ_CONTACTS, WRITE_CONTACTS, GET_ACCOUNTS
  • **LOCATION**: ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION
  • **PHONE**: READ_PHONE_STATE, CALL_PHONE, READ_CALL_LOG, WRITE_CALL_LOG, ADD_VOICEMAIL, USE_SIP, PROCESS_OUTGOING_CALLS
  • **SMS**: SEND_SMS, RECEIVE_SMS, READ_SMS, RECEIVE_WAP_PUSH, RECEIVE_MMS
  • **STORAGE**: READ_EXTERNAL_STORAGE, WRITE_EXTERNAL_STORAGE

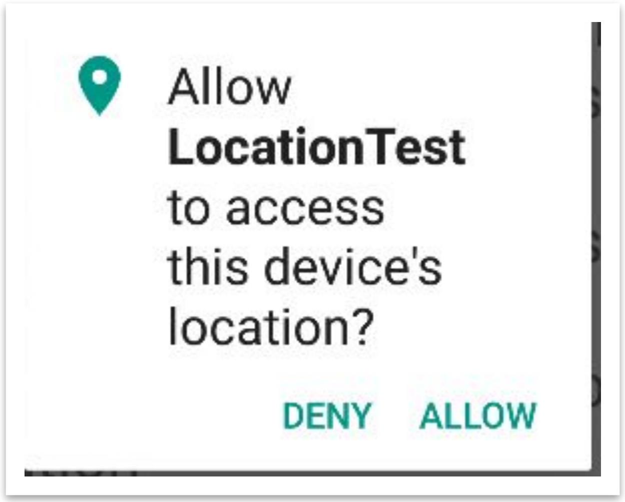
Permissions Post Android 6.0

- Post Android 6.0 / Marshmallow all Normal and Dangerous Permissions **still** placed in app Manifest
- If Device is Android 5.1 or lower OR app targets API level 22 or lower then
  - User must grant Dangerous Permissions at install time
  - Or app doesn't install
- If device is Android 6.0 or higher AND app targets API level 23 or higher then
  - Must still list all permissions in manifest
  - App must request each dangerous permission in needs **while the app is running**


Cougar Sounds

needs access to

Photos/Media/Files


Allow **LocationTest** to access this device's location?

ANDROID PERMISSIONS

Part 2

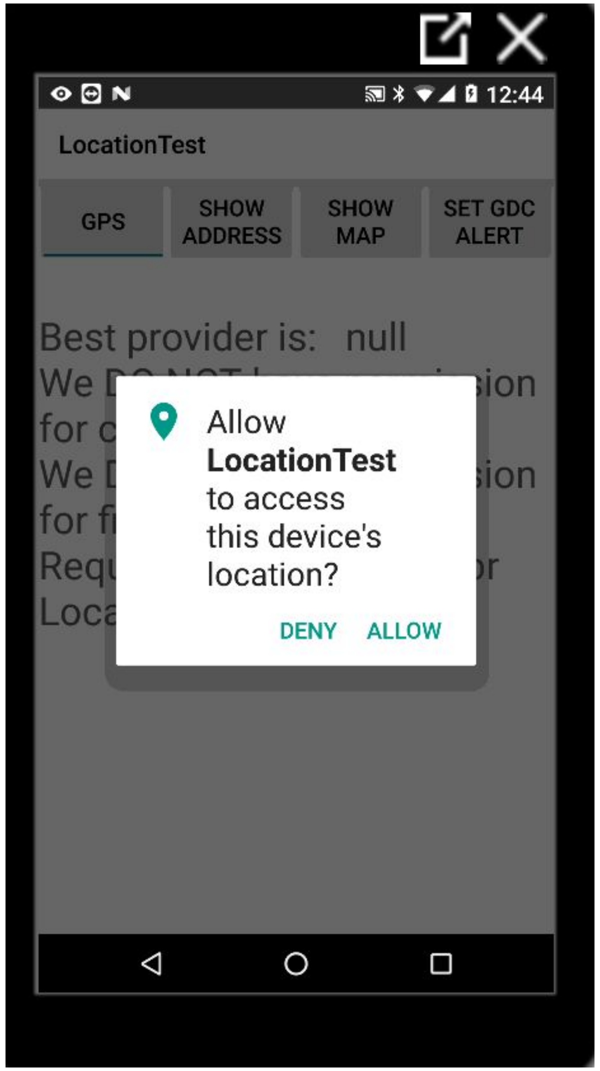
Requesting Permissions

- Requesting permission can be done in two ways.

1. Requesting Permission ***by yourself***: Traditional Approach
2. Requesting Permission ***by System***: Recommended Approach

Requesting Permissions: General format

- When app needs dangerous permission
  - Check to see if you already have permission
  - If not call one of the requestPermission methods with desired permissions and request code
  - requestPermission shows a standard, non modifiable dialog box to the user



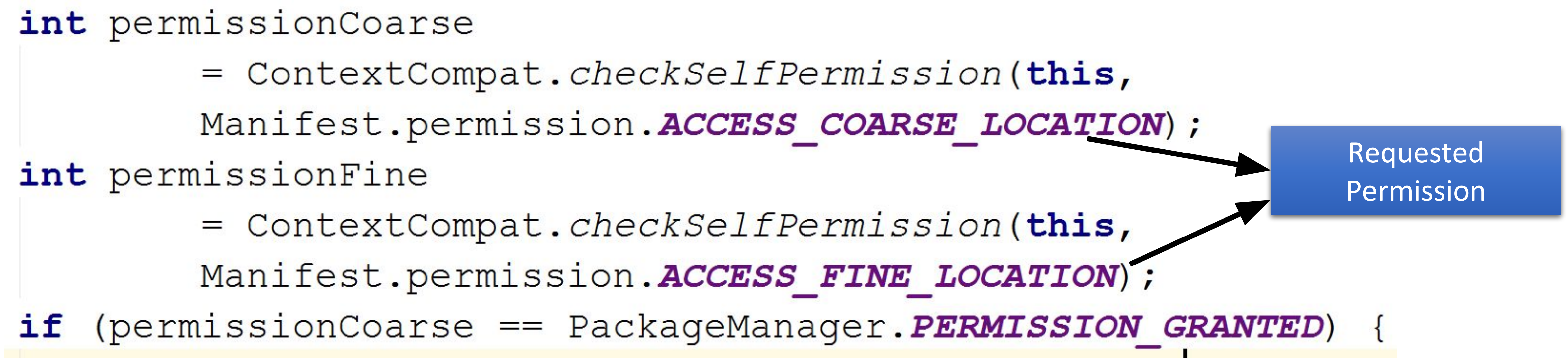
LocationTest

Best provider is: null

Allow **LocationTest** to access this device's location?

Requesting Permissions: By yourself

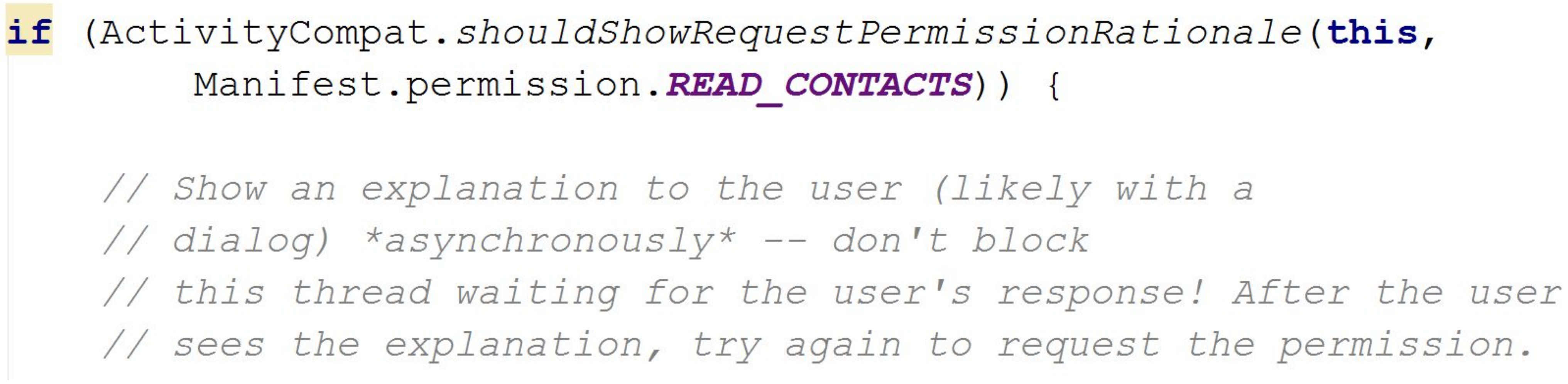
- **Checking Permissions:** Before Requesting a Permission, check to see if you already have it.
- To check if the user has already granted your app a particular permission, pass that permission into the ContextCompat.checkSelfPermission() method. This method returns either PERMISSION_GRANTED or PERMISSION_DENIED, depending on whether your app has the permission.



```java
int permissionCoarse
        = ContextCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_COARSE_LOCATION);
int permissionFine
        = ContextCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_FINE_LOCATION);
if (permissionCoarse == PackageManager.PERMISSION_GRANTED) {
```

Requesting Permissions: By yourself

- **Explain why your app needs permission:** If the ContextCompat.checkSelfPermission() method returns PERMISSION_DENIED, call shouldShowRequestPermissionRationale().
- If this method returns true, show an educational UI to the user. In this UI, describe why the feature, which the user wants to enable, needs a particular permission.

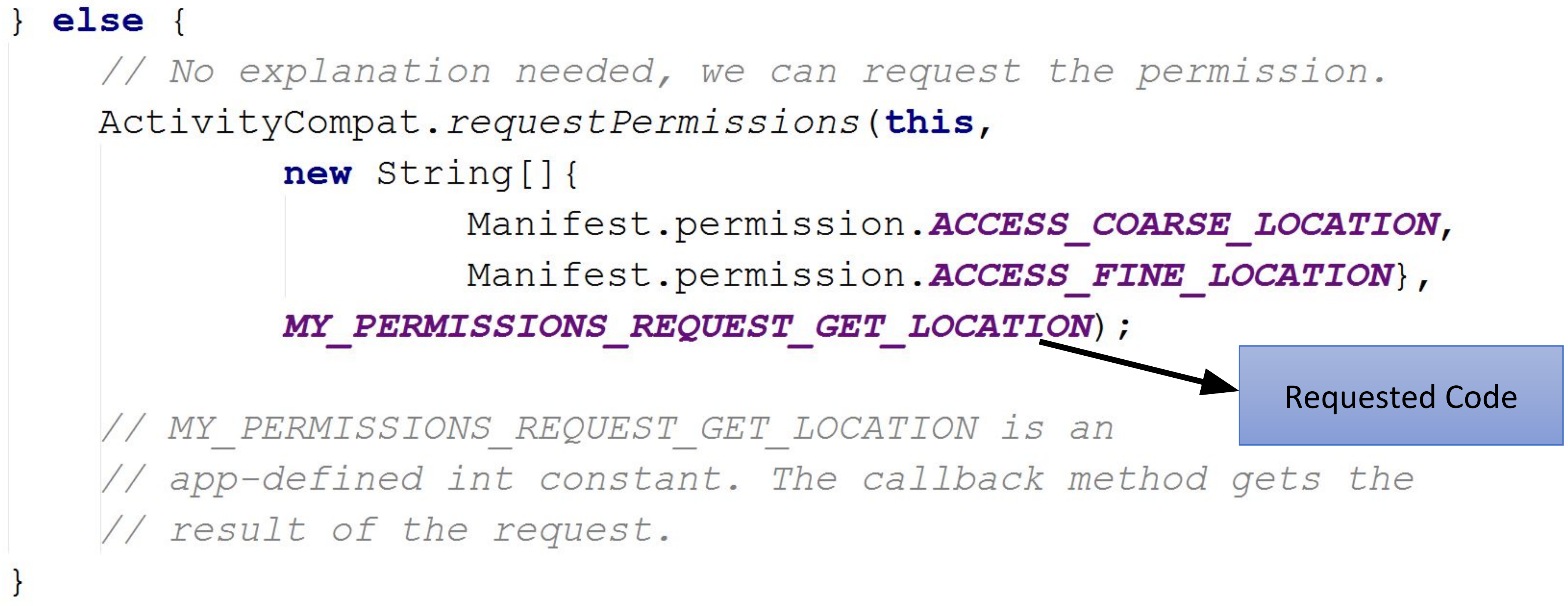


```
if (ActivityCompat.shouldShowRequestPermissionRationale(this,
        Manifest.permission.READ_CONTACTS)) {

    // Show an explanation to the user (likely with a
    // dialog) *asynchronously* -- don't block
    // this thread waiting for the user's response! After the user
    // sees the explanation, try again to request the permission.
```

- shouldShowRequestPermissionRationale returns true if the app previously requested the permission and the user denied it

Requesting Permissions: By yourself

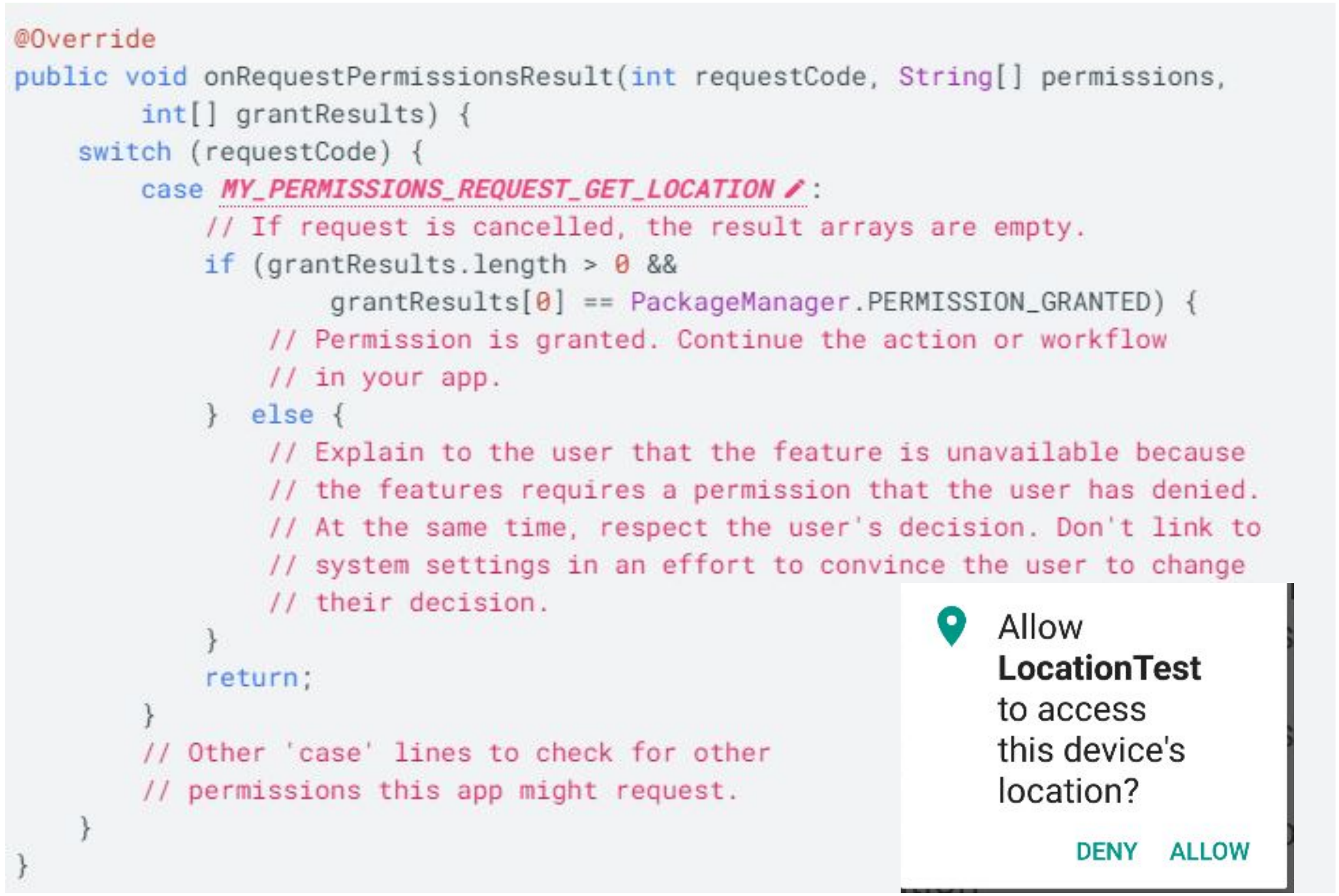
- **Requesting permission:** If user has not previously denied permission then request the permission. Users see a system permission dialog, where they can choose whether to grant a particular permission to your app.



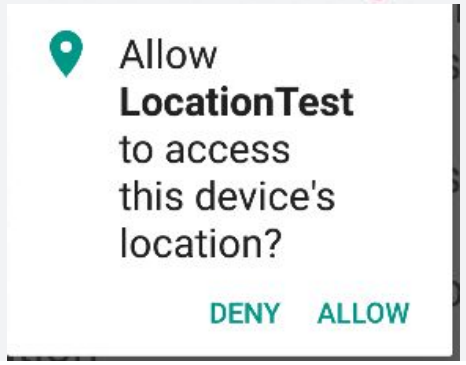
```
} else {
    // No explanation needed, we can request the permission.
    ActivityCompat.requestPermissions(this,
            new String[]{
                    Manifest.permission.ACCESS_COARSE_LOCATION,
                    Manifest.permission.ACCESS_FINE_LOCATION},
            MY_PERMISSIONS_REQUEST_GET_LOCATION);

    // MY_PERMISSIONS_REQUEST_GET_LOCATION is an
    // app-defined int constant. The callback method gets the
    // result of the request.
}
```

Requesting Permissions: By yourself

- **Dialog box:** Dialog displayed to user. Lists permission group, not individual permissions.
- After the user responds to the system permissions dialog, the system then invokes your app's implementation of onRequestPermissionsResult().
- The system passes in the user response to the permission dialog, as well as the request code that you defined.



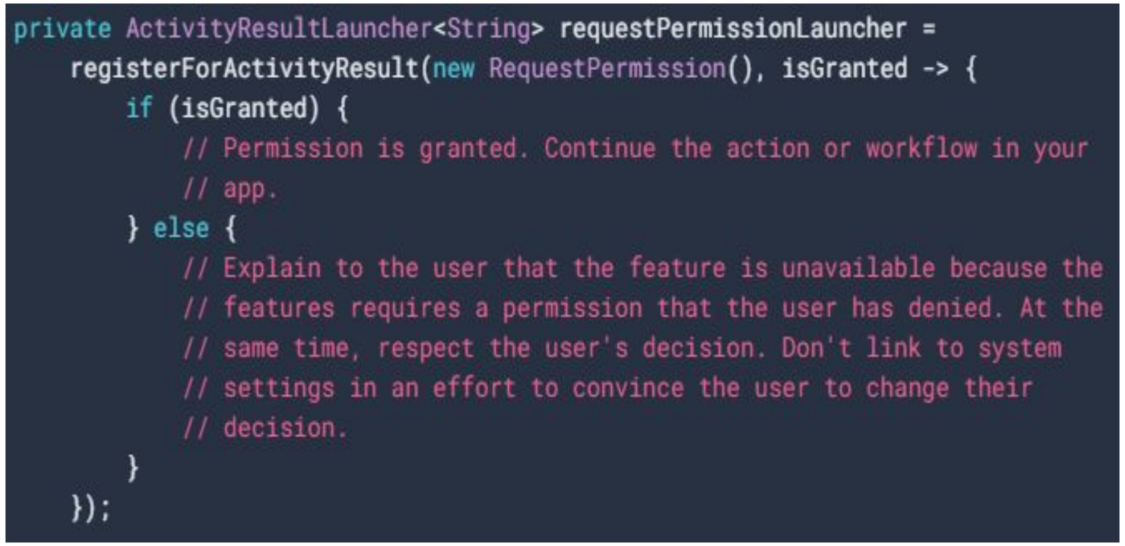
```
@Override
public void onRequestPermissionsResult(int requestCode, String[] permissions,
        int[] grantResults) {
    switch (requestCode) {
        case MY_PERMISSIONS_REQUEST_GET_LOCATION:
            // If request is cancelled, the result arrays are empty.
            if (grantResults.length > 0 &&
                    grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                // Permission is granted. Continue the action or workflow
                // in your app.
            }  else {
                // Explain to the user that the feature is unavailable because
                // the features requires a permission that the user has denied.
                // At the same time, respect the user's decision. Don't link to
                // system settings in an effort to convince the user to change
                // their decision.
            }
            return;
        }
        // Other 'case' lines to check for other
        // permissions this app might request.
    }
}
```



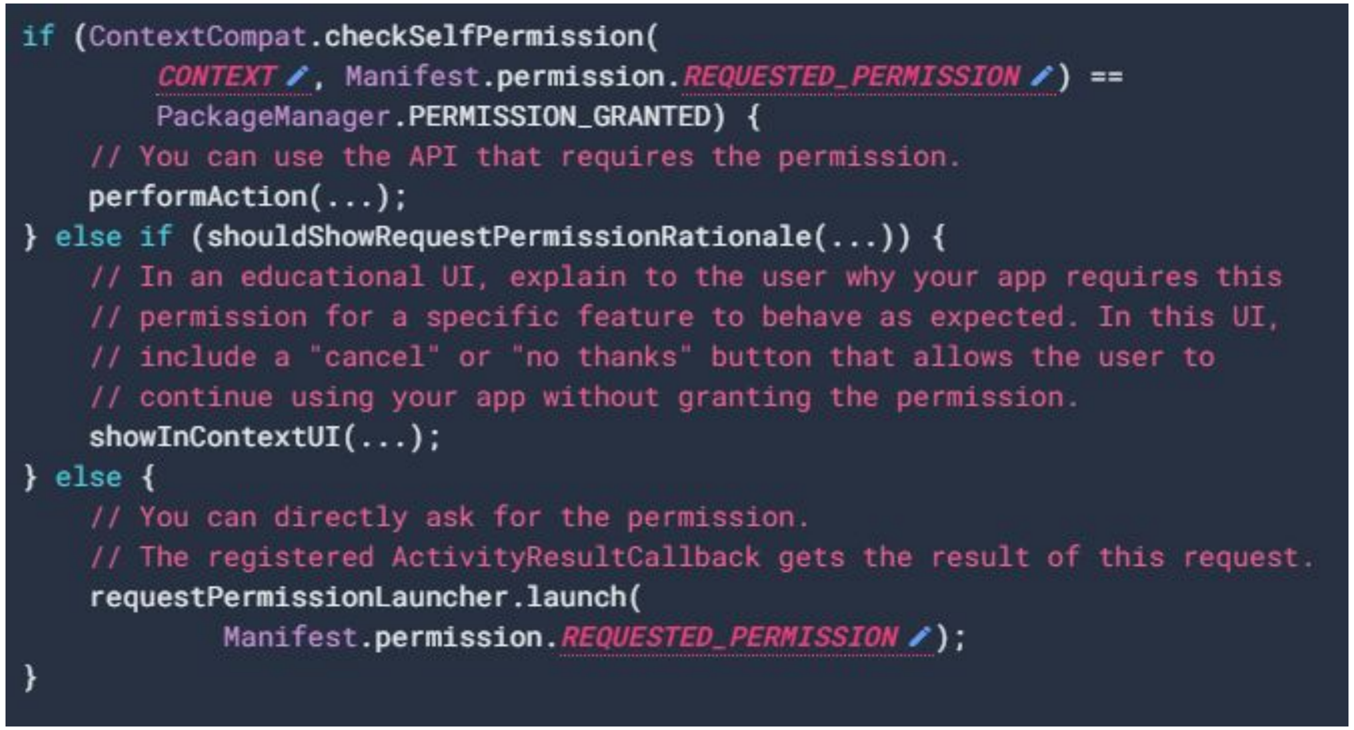
Allow **LocationTest** to access this device's location?

Requesting Permissions: By System

- To allow the system to manage the request code that's associated with a permissions request, add a dependency on the androidx.activity library in your module's build.gradle file.
- Here the System controls the response of the user through the registerForActivityResult() instead of the onRequestPermissionsResult().



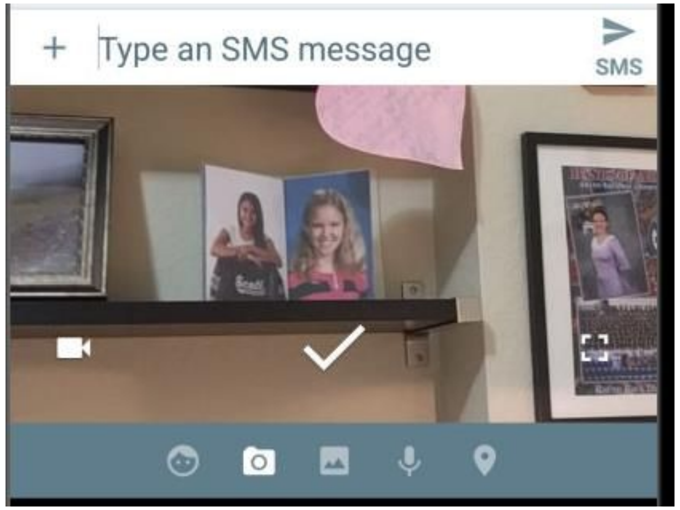
```
private ActivityResultLauncher<String> requestPermissionLauncher =
    registerForActivityResult(new RequestPermission(), isGranted -> {
        if (isGranted) {
            // Permission is granted. Continue the action or workflow in your
            // app.
        } else {
            // Explain to the user that the feature is unavailable because the
            // features requires a permission that the user has denied. At the
            // same time, respect the user's decision. Don't link to system
            // settings in an effort to convince the user to change their
            // decision.
        }
    });
```

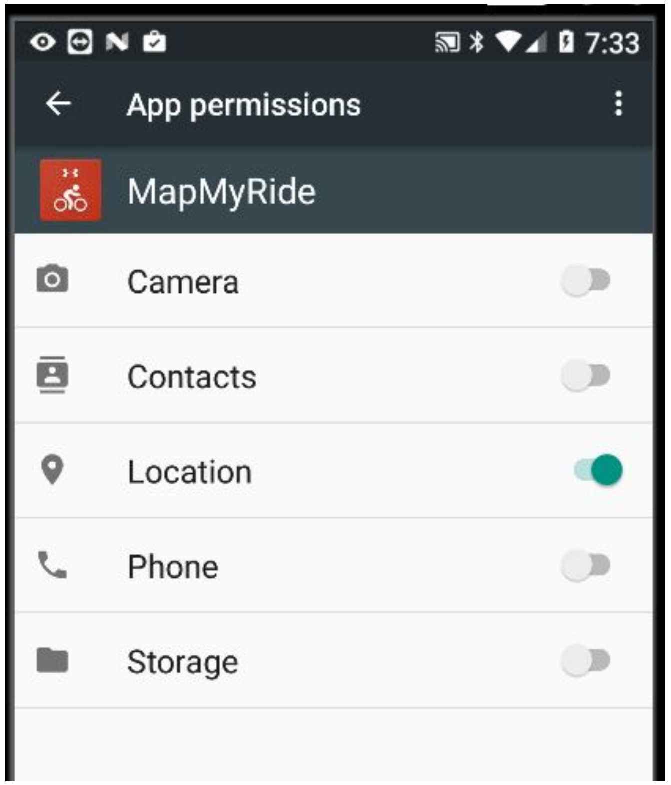


```
if (ContextCompat.checkSelfPermission(
        CONTEXT, Manifest.permission.REQUESTED_PERMISSION) ==
        PackageManager.PERMISSION_GRANTED) {
    // You can use the API that requires the permission.
    performAction(...);
} else if (shouldShowRequestPermissionRationale(...)) {
    // In an educational UI, explain to the user why your app requires this
    // permission for a specific feature to behave as expected. In this UI,
    // include a "cancel" or "no thanks" button that allows the user to
    // continue using your app without granting the permission.
    showInContextUI(...);
} else {
    // You can directly ask for the permission.
    // The registered ActivityResultCallback gets the result of this request.
    requestPermissionLauncher.launch(
            Manifest.permission.REQUESTED_PERMISSION);
}
```

App Permissions Notes

- If you use an intent to accomplish something, then your app does not usually have to request permission
- For example, if you use an Intent to take picture, you DO NOT need CAMERA permission.
  - Only if you want to access camera directly in your app.
  - Example, messenger ->
- A user can also alter permissions in the Settings app
  - Settings -> app
  - Toggle Switches to grant and deny permissions




7:33
App permissions
MapMyRide

Camera	off
Contacts	off
Location	on
Phone	off
Storage	off

App Permissions Best Practices

- ***Only use the permissions necessary for your app to work***. Depending on how you are using the permissions, there may be another way to do what you need (system intents, identifiers, backgrounding for phone calls) without relying on access to sensitive information.
- ***Pay attention to permissions required by libraries.*** When you include a library, you also inherit its permission requirements. You should be aware of what you're including, the permissions they require, and what those permissions are used for.
- ***Be transparent.*** When you make a permissions request, be clear about what you're accessing, and why, so users can make informed decisions..
- ***Make system accesses explicit.*** Providing continuous indications when you access sensitive capabilities (for example, the camera or microphone) makes it clear to users when you're collecting data and avoids the perception that you're collecting data surreptitiously.

- https://developer.android.com/training/permissions/usage-notes: Alternatives for App Permissions

THANK YOU