

# Send Notification

Dr. Charles Yu

- Send a notification to phone user
- Here are the frequently seen use cases
  - For the perspective on the system's level
    - When we get the system Broadcast intents from the Broadcast Receiver
      - Receiving a phone call
      - Receiving a SMS/MMS
  - For the perspective on end user's application's level
    - When something is going wrong
      - Data collection
      - Data computation

# Send a notification to phone user

- [Demo1] SendNotification
  - In this application, we only have one Java file --- MainActivity
    - And one button
    - Click the button and send the Notification
    - Here Is an example (See the next page)

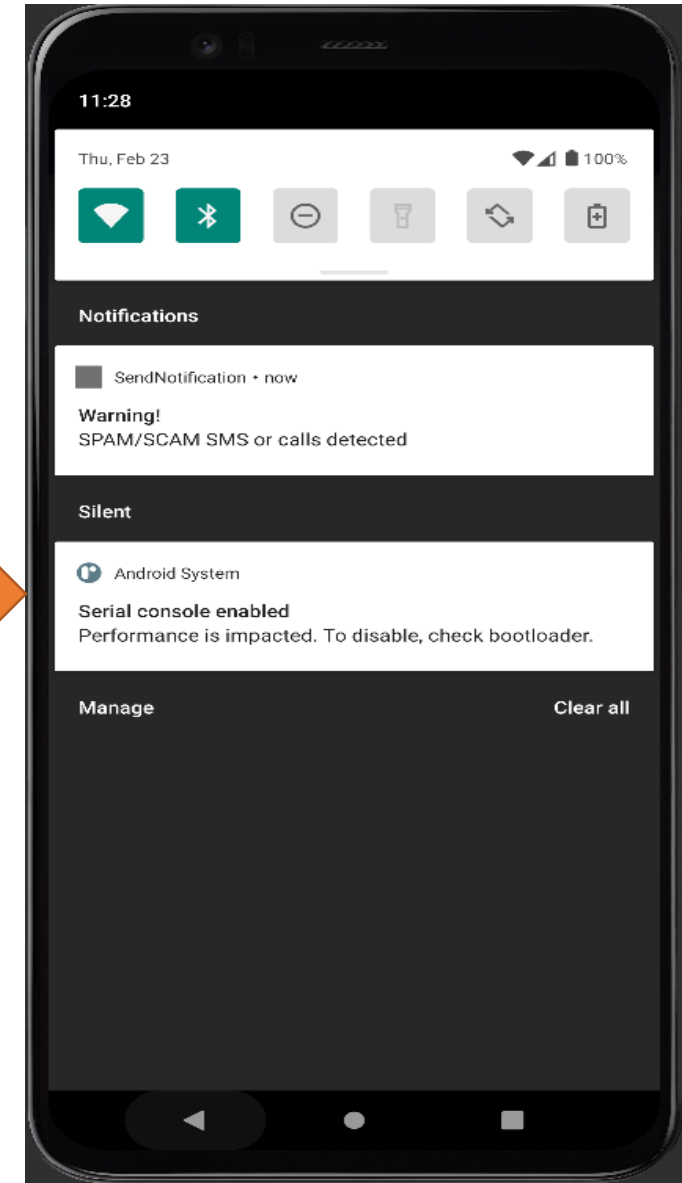
# Send a notification to phone user



Click the  
Button

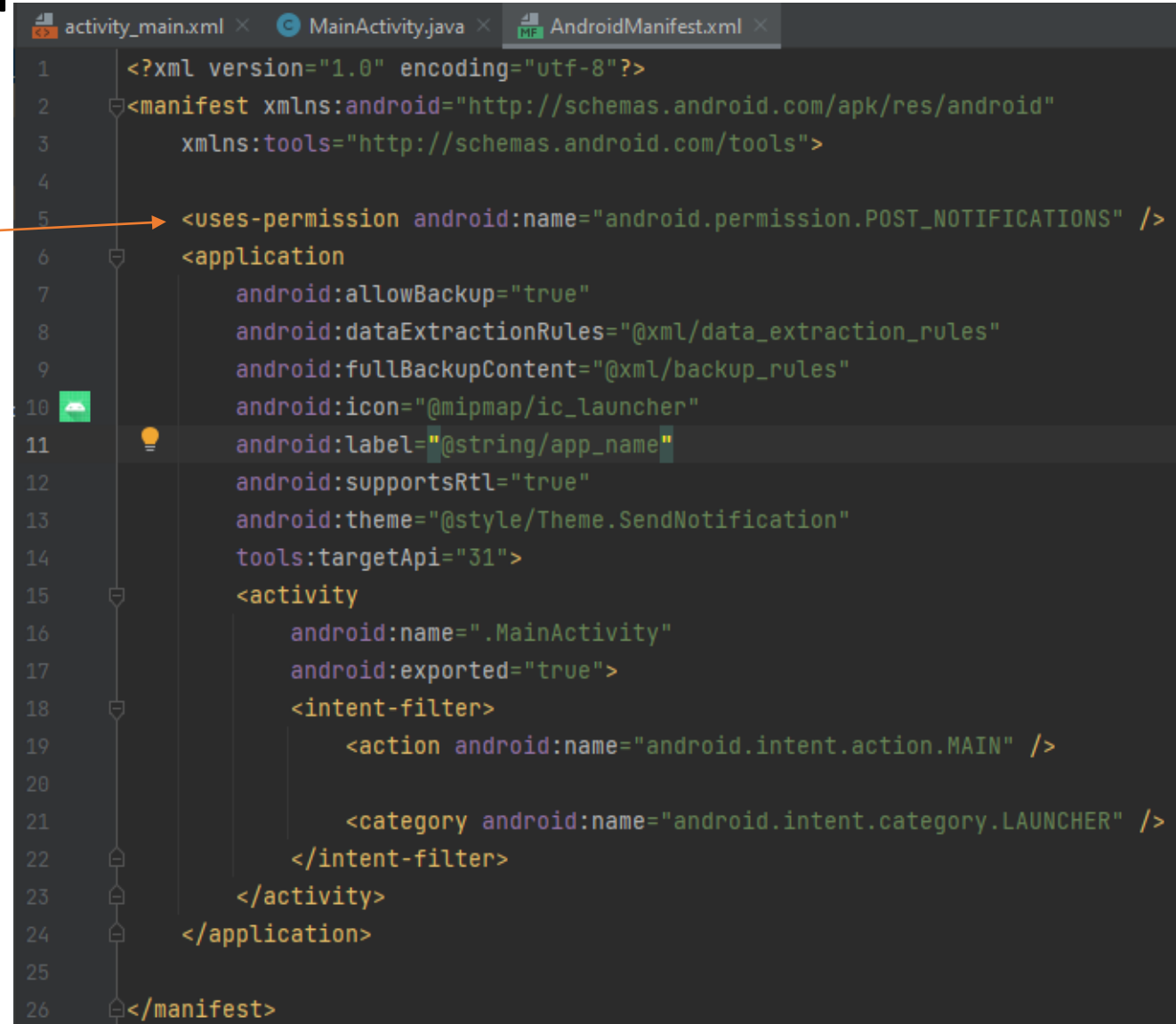


Swipe it  
Down



# Send a notification to phone user

- Our AndroidManifest.xml
  - The installation access right
    - This is required
    - POST\_NOTIFICATION

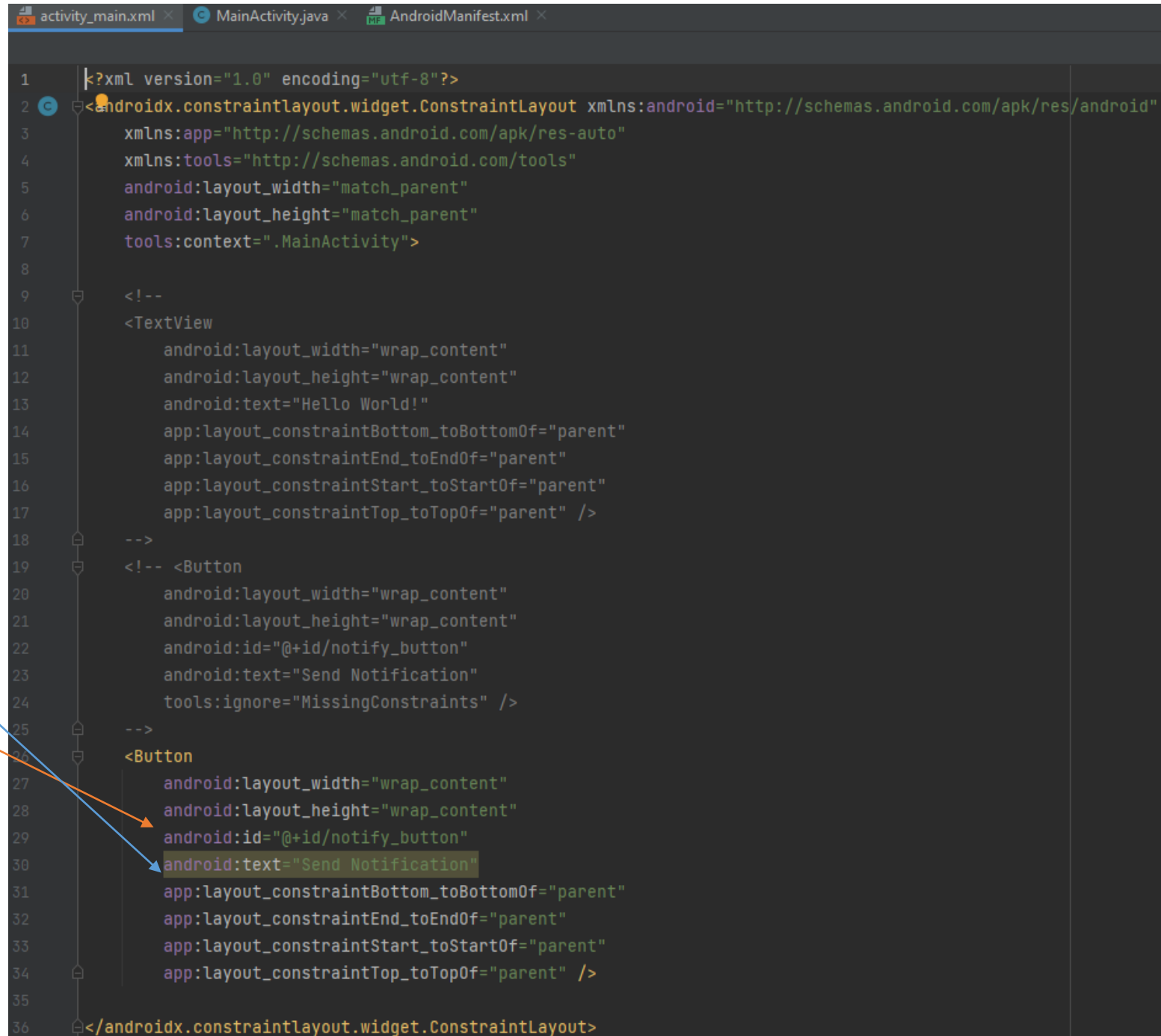


```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3          xmlns:tools="http://schemas.android.com/tools">
4
5      <uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
6
7      <application
8          android:allowBackup="true"
9          android:dataExtractionRules="@xml/data_extraction_rules"
10         android:fullBackupContent="@xml/backup_rules"
11         android:icon="@mipmap/ic_launcher"
12         android:label="@string/app_name"
13         android:supportRtl="true"
14         android:theme="@style/Theme.SendNotification"
15         tools:targetApi="31">
16
17         <activity
18             android:name=".MainActivity"
19             android:exported="true">
20
21             <intent-filter>
22                 <action android:name="android.intent.action.MAIN" />
23
24                 <category android:name="android.intent.category.LAUNCHER" />
25             </intent-filter>
26         </activity>
27     </application>
28 </manifest>
```

The screenshot shows the AndroidManifest.xml file in an IDE. An orange arrow points from the text 'The installation access right' in the list to the line of code: `<uses-permission android:name="android.permission.POST_NOTIFICATIONS" />`. The IDE tabs at the top show 'activity\_main.xml', 'MainActivity.java', and 'AndroidManifest.xml'.

# Send a notification to phone user

- activity\_main.xml
- I comment out the default “TextView” and put a “Button” over there.
- This is the id
- This is the displaying text



```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".MainActivity">
8
9      <!--
10      <TextView
11          android:layout_width="wrap_content"
12          android:layout_height="wrap_content"
13          android:text="Hello World!"
14          app:layout_constraintBottom_toBottomOf="parent"
15          app:layout_constraintEnd_toEndOf="parent"
16          app:layout_constraintStart_toStartOf="parent"
17          app:layout_constraintTop_toTopOf="parent" />
18      -->
19      <!-- <Button
20          android:layout_width="wrap_content"
21          android:layout_height="wrap_content"
22          android:id="@+id/notify_button"
23          android:text="Send Notification"
24          tools:ignore="MissingConstraints" />
25      -->
26      <Button
27          android:layout_width="wrap_content"
28          android:layout_height="wrap_content"
29          android:id="@+id/notify_button"
30          android:text="Send Notification"
31          app:layout_constraintBottom_toBottomOf="parent"
32          app:layout_constraintEnd_toEndOf="parent"
33          app:layout_constraintStart_toStartOf="parent"
34          app:layout_constraintTop_toTopOf="parent" />
35
36  </androidx.constraintlayout.widget.ConstraintLayout>
```

# Send a notification to phone user

- We still need to use the request code to send runtime permission
- The Context object is Instantiated
- The button, which we will use it later to send the notification

```
20 public class MainActivity extends AppCompatActivity {
21
22     2 usages
23     Button notifyBtn;
24     4 usages
25     Context mContext;
26
27     2 usages
28     private static final int MY_PERMISSIONS_POST_NOTIFICATIONS = 0; // This is the request code
29
30     @Override
31     protected void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33         setContentView(R.layout.activity_main);
34
35         mContext = this.getApplicationContext();
36
37         notifyBtn = findViewById(R.id.notify_button);
38
39         /*
40          * This means if the version is newer
41          * 0: Oreo, API level 26, Android v8.0
42          * https://en.wikipedia.org/wiki/Android\_version\_history
43          */
44
45         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
46             NotificationChannel channel = new NotificationChannel( id: "MyNotificationChannelId",
47                 name: "MyNotificationChannelName",
48                 NotificationManager.IMPORTANCE_DEFAULT);
49
50             NotificationManager manager = getSystemService(NotificationManager.class);
51             manager.createNotificationChannel(channel);
52         }
53     }
54 }
```

# Send a notification to phone user

- If the version of Android OS (emulator) is greater than v8.0 (**O**reo), we need to setup the **notification channel** by passing channel **id**, channel **name**, and **importance** level into the constructor

```
20 public class MainActivity extends AppCompatActivity {
21
22     2 usages
23     Button notifyBtn;
24     4 usages
25     Context mContext;
26
27     2 usages
28     private static final int MY_PERMISSIONS_POST_NOTIFICATIONS = 0; // This is the request code
29
30     @Override
31     protected void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33         setContentView(R.layout.activity_main);
34
35         mContext = this.getApplicationContext();
36
37         notifyBtn = findViewById(R.id.notify_button);
38
39         /*
40          * This means if the version is newer
41          * 0: Oreo, API level 26, Android v8.0
42          * https://en.wikipedia.org/wiki/Android\_version\_history
43          */
44         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
45             NotificationChannel channel = new NotificationChannel( id: "MyNotificationChannelId",
46                 name: "MyNotificationChannelName",
47                 NotificationManager.IMPORTANCE_DEFAULT);
48
49             NotificationManager manager = getSystemService(NotificationManager.class);
50             manager.createNotificationChannel(channel);
51         }
52     }
53 }
```



# Send a notification to phone user

- The notification manager is instantiated by calling `getSystemService()`
- Then, we use the notification manager to create the notification channel by passing the channel we just build.

```
20 public class MainActivity extends AppCompatActivity {
21
22     2 usages
23     Button notifyBtn;
24     4 usages
25     Context mContext;
26
27     2 usages
28     private static final int MY_PERMISSIONS_POST_NOTIFICATIONS = 0; // This is the request code
29
30     @Override
31     protected void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33         setContentView(R.layout.activity_main);
34
35         mContext = this.getApplicationContext();
36
37         notifyBtn = findViewById(R.id.notify_button);
38
39         /*
40          * This means if the version is newer
41          * 0: Oreo, API level 26, Android v8.0
42          * https://en.wikipedia.org/wiki/Android\_version\_history
43          */
44
45         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
46             NotificationChannel channel = new NotificationChannel(id: "MyNotificationChannelId",
47                 name: "MyNotificationChannelName",
48                 NotificationManager.IMPORTANCE_DEFAULT);
49
50             NotificationManager manager = getSystemService(NotificationManager.class);
51             manager.createNotificationChannel(channel);
52         }
53     }
54 }
```

# Send a notification to phone user

- Now, it's about the time to deal with the button
- Get the notification builder instance first

```
notifyBtn.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        // Notification code goes here  
  
        NotificationCompat.Builder builder = new NotificationCompat.Builder(mContext, channelId: "MyNotificationChannelId");  
        builder.setContentTitle("Warning!");  
        builder.setContentText("SPAM/SCAM SMS or calls detected");  
        builder.setSmallIcon(R.drawable.ic_launcher_background);  
        builder.setAutoCancel(true);  
  
        NotificationManagerCompat managerCompat = NotificationManagerCompat.from(mContext);  
  
        if (ActivityCompat.checkSelfPermission(mContext, android.Manifest.permission.POST_NOTIFICATIONS) != PackageManager.PERMISSION_GRANTED) {  
            // TODO: Consider calling  
            //     ActivityCompat#requestPermissions  
            // here to request the missing permissions, and then overriding  
            // public void onRequestPermissionsResult(int requestCode, String[] permissions,  
            //                                     int[] grantResults)  
            // to handle the case where the user grants the permission. See the documentation  
            // for ActivityCompat#requestPermissions for more details.  
  
            ActivityCompat.requestPermissions(activity: MainActivity.this,  
                new String[]{ android.Manifest.permission.RECEIVE_SMS},  
                MY_PERMISSIONS_POST_NOTIFICATIONS);  
        }  
  
        // As for the id, you can pass any unique id.  
        // In this way, I'm just saying the notification id starts from zero  
        managerCompat.notify(id: 0, builder.build());  
    }  
});
```

# Send a notification to phone user

- Setup the notification, its title and text content
- Setup the representing **icon** and **AutoCancel** properties

```
notifyBtn.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        // Notification code goes here  
  
        NotificationCompat.Builder builder = new NotificationCompat.Builder(mContext, channelId: "MyNotificationChannelId");  
        builder.setContentTitle("Warning!");  
        builder.setContentText("SPAM/SCAM SMS or calls detected");  
        builder.setSmallIcon(R.drawable.ic_launcher_background);  
        builder.setAutoCancel(true);  
  
        NotificationManagerCompat managerCompat = NotificationManagerCompat.from(mContext);  
  
        if (ActivityCompat.checkSelfPermission(mContext, android.Manifest.permission.POST_NOTIFICATIONS) != PackageManager.PERMISSION_GRANTED) {  
            // TODO: Consider calling  
            //     ActivityCompat#requestPermissions  
            // here to request the missing permissions, and then overriding  
            //     public void onRequestPermissionsResult(int requestCode, String[] permissions,  
            //                                           int[] grantResults)  
            // to handle the case where the user grants the permission. See the documentation  
            // for ActivityCompat#requestPermissions for more details.  
  
            ActivityCompat.requestPermissions(activity: MainActivity.this,  
                new String[]{ android.Manifest.permission.RECEIVE_SMS},  
                MY_PERMISSIONS_POST_NOTIFICATIONS);  
        }  
  
        // As for the id, you can pass any unique id.  
        // In this way, I'm just saying the notification id starts from zero  
        managerCompat.notify(id: 0, builder.build());  
    }  
});
```

# Send a notification to phone user

- Get the NotificationManagerCompat Object
- We are going to use this object to **send out the notification**
- We still need to ask for **run-time permissions** from user
- Check the slide #5, it is the **only one permission** we need

```
notifyBtn.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        // Notification code goes here  
  
        NotificationCompat.Builder builder = new NotificationCompat.Builder(mContext, channelId: "MyNotificationChannelId");  
        builder.setContentTitle("Warning!");  
        builder.setContentText("SPAM/SCAM SMS or calls detected");  
        builder.setSmallIcon(R.drawable.ic_launcher_background);  
        builder.setAutoCancel(true);  
  
        NotificationManagerCompat managerCompat = NotificationManagerCompat.from(mContext);  
  
        if (ActivityCompat.checkSelfPermission(mContext, android.Manifest.permission.POST_NOTIFICATIONS) != PackageManager.PERMISSION_GRANTED) {  
            // TODO: Consider calling  
            // ActivityCompat#requestPermissions  
            // here to request the missing permissions, and then overriding  
            // public void onRequestPermissionsResult(int requestCode, String[] permissions,  
            //                                     int[] grantResults)  
            // to handle the case where the user grants the permission. See the documentation  
            // for ActivityCompat#requestPermissions for more details.  
  
            ActivityCompat.requestPermissions(activity: MainActivity.this,  
                new String[]{ android.Manifest.permission.RECEIVE_SMS},  
                MY_PERMISSIONS_POST_NOTIFICATIONS);  
        }  
  
        // As for the id, you can pass any unique id.  
        // In this way, I'm just saying the notification id starts from zero  
        managerCompat.notify(id: 0, builder.build());  
    }  
});
```

# Send a notification to phone user

- There is a thing.

(weird)

If I don't put this

**"check the permission"**

I will get a compilation error.

- However, if I put this, it doesn't show the run-time requesting dialog, like we have seen earlier

```
notifyBtn.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        // Notification code goes here  
  
        NotificationCompat.Builder builder = new NotificationCompat.Builder(mContext, channelId: "MyNotificationChannelId");  
        builder.setContentTitle("Warning!");  
        builder.setContentText("SPAM/SCAM SMS or calls detected");  
        builder.setSmallIcon(R.drawable.ic_launcher_background);  
        builder.setAutoCancel(true);  
  
        NotificationManagerCompat managerCompat = NotificationManagerCompat.from(mContext);  
  
        if (ActivityCompat.checkSelfPermission(mContext, android.Manifest.permission.POST_NOTIFICATIONS) != PackageManager.PERMISSION_GRANTED) {  
            // TODO: Consider calling  
            // ActivityCompat#requestPermissions  
            // here to request the missing permissions, and then overriding  
            // public void onRequestPermissionsResult(int requestCode, String[] permissions,  
            //                                     int[] grantResults)  
            // to handle the case where the user grants the permission. See the documentation  
            // for ActivityCompat#requestPermissions for more details.  
            ActivityCompat.requestPermissions(activity: MainActivity.this,  
                new String[]{ android.Manifest.permission.RECEIVE_SMS},  
                MY_PERMISSIONS_POST_NOTIFICATIONS);  
        }  
  
        // As for the id, you can pass any unique id.  
        // In this way, I'm just saying the notification id starts from zero  
        managerCompat.notify(id: 0, builder.build());  
    }  
});
```

# Send a notification to phone user

- The last piece in MainActivity. We put a call-back function to handle the permission status
- However, it seems that **it is NEVER get executed! Because I cannot see the Toast at all!**

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);

    switch (requestCode) {
        case MY_PERMISSIONS_POST_NOTIFICATIONS:
            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                Toast.makeText(context: this, text: "Permission Granted", Toast.LENGTH_LONG).show();
            } else {
                Toast.makeText(context: this, text: "Permission Not Granted!", Toast.LENGTH_LONG).show();
                this.finish();
            }
        }
    }
}
```

# Send a notification to phone user

- In conclusion. It seems like **we only** need to check the access-rights to **send the notification**. And, this is a **MUST**.
- However, we don't really need to request run-time permission **"officially"** for sending the notification
- It seems like the access rights for "sending the notification" **is not the security critical jobs?**