LECTURE 22: NOTIFICATIONS

BY LINA HAMMAD & AHMAD BARGHASH

In this lecture, we will learn how to create notifications in kotlin

NOTIFICATIONS IN KOTLIN

- **Notification** is a message that is used to display some short messages outside of our main application. Even if the app is not running, notifications will still work. Notifications have the following contents: an icon, title of notification and some text content.

- In this lecture we will be discussing how to produce notifications in Kotlin .

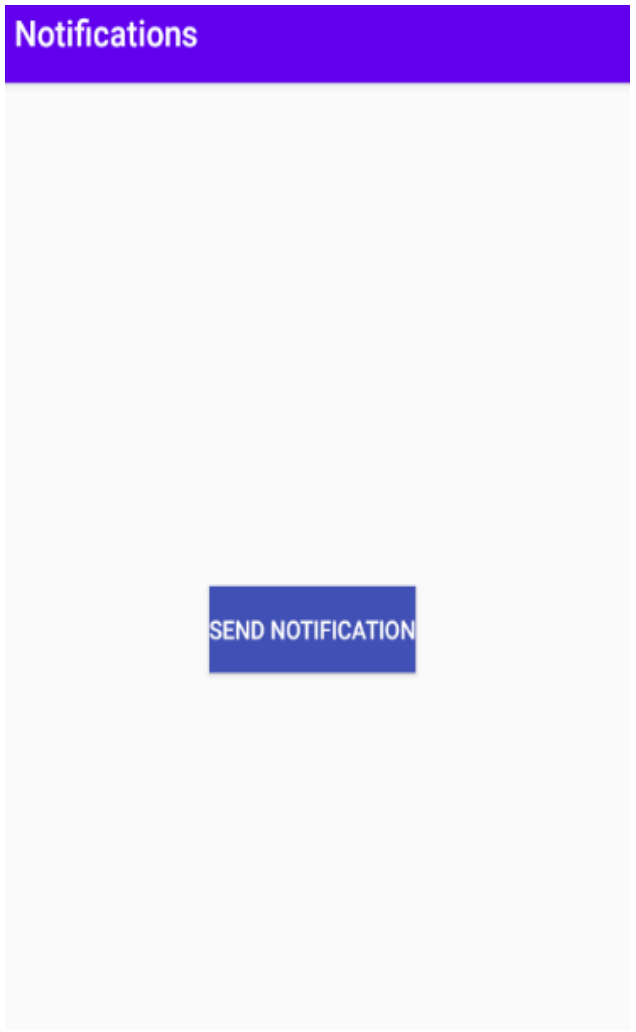
Step1: Crete new project

- Let's start by first creating a project in Android Studio. To do so, follow these instructions:
  - Click on File, then New and then New Project and give name whatever you like
  - Then, select Kotlin language Support and click next button
  - Select minimum SDK, whatever you need.
  - Select Empty activity and then click finish.

Step2: Modify activity_main.xml file

- Second step is to design our layout page.

```
<Button
    android:id="@+id/btn_1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:text="Send Notification“
    android:background="#3F51B5"
    android:textColor="#FFFFFF"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    />
```



Step3: Create a New Activity Named AfterNotification

- Create a new activity named AfterNotification. For this go to res/layout then right click and select new then activity, Empty Activity. When someone clicks on the notification, this activity will open up in our app that is the user will be redirected to this page.

- activity_after_notification.xml:

```
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:text="Activity After Notifications"
    android:textColor="#3F51B5"
    android:textSize="15sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

- AfterNotification.kt

```
class AfterNotification : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_after_notification)
    }
}
```

NOTE

- Without configuring Notification Channels, you cannot build notification for applications with Android API >=26. For them generating a notification channel is mandatory. Apps with API<26 doesn't need notification channel they just need notification builder. Each channel is supposed to have a particular behavior which will be applicable to all the notifications which are a part of it.

- Every channel will therefore have a Channel ID which basically will act as a unique identifier for this Channel which will be useful if user wants to differentiate a particular notification channel. In contrast, Notification builder provides a convenient way to set the various fields of a Notification and generate content views using the platform's notification layout template but is not able to target a particular notification channel.

Displaying Your First Notification

- In order to display a notification, you will have to do some setup work. You will need to:
  - Create a notification channel.
  - Pending Intents
  - Create a notification.
  - Send a notification using **NotificationManager**.

Create a Notification Channel

- Notification channels provide a common visual and auditory experience for notifications of a similar type. Since their introduction in API 26, you are now required to set a channel for a notification, otherwise they will not display on newer versions of Android.

```
//1: checking if android version is greater than oreo(API 26) or not
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    val channelId = "com.example.notifications"
    val description = "Test notification"

    //2: Create a unique name for the notification channel
    notificationChannel = NotificationChannel(
        channelId,
        description,
        NotificationManager.IMPORTANCE_DEFAULT
    )

    notificationChannel.enableLights(true)
    notificationChannel.lightColor = Color.GREEN
    notificationChannel.enableVibration(false)

    //3: Create the channel using the NotificationManager.
    notificationManager =
        getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager
    notificationManager.createNotificationChannel(notificationChannel)
}
```

Here you:

1. Safety checked the OS version for API 26 and greater.
2. Created a unique name for the notification channel. The name and description are displayed in the application's Notification settings.
3. Created the channel using the NotificationManager.

Pending Intents

- So far you have created the notification channel and began building a notification but you want to direct the user somewhere when they tap on the notification. This requires adding a PendingIntent to the notification calling setContentIntent when building the notification. A PendingIntent is roughly described as an action to be taken at a later point in time.

```
//4: pendingIntent is an intent for future use i.e after the notification is clicked, this intent will come into
action
val intent = Intent(this, AfterNotification::class.java)

//FLAG_UPDATE_CURRENT specifies that if a previous PendingIntent already exists, then the current one
// will update it with the latest intent.
// 0 is the request code, using it later with the same method again will get back the same pending
// intent for future reference.
//intent passed here is to our AfterNotification class

val pendingIntent = PendingIntent.getActivity(
    this,
    0, intent, PendingIntent.FLAG_UPDATE_CURRENT
)
```

CREATE A NOTIFICATION

- Here you:
  - Use Notification.Builder to begin building the notification.
  - Set a small icon to be display in the notification shade. This is the only required attribute.
  - Set a title for the notification.
  - Set content for the notification.
  - Set the unique channelId for the notification.

```
//5: create the notification: use Notification.Builder to begin building the notification.
builder = Notification.Builder(this, channelId)
    .setSmallIcon(android.R.drawable.ic_dialog_info) // Set a small icon to be display in the notification shade.
    .setContentTitle("Example Notification") // Set a title for the notification.
    .setContentText("This is an  example notification.")//Set content for the notification.
    .setContentIntent(pendingIntent)
    .setChannelId(channelId)
```

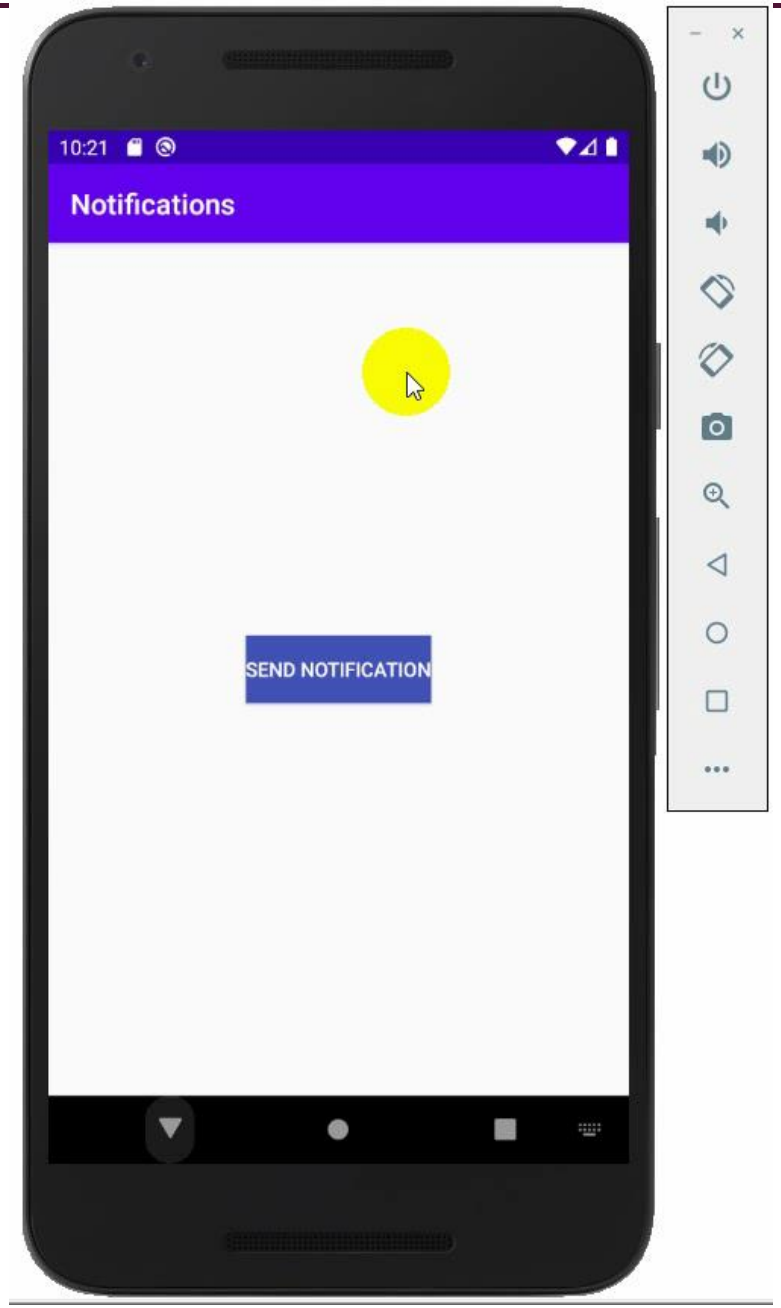
NOTIFYING THE MANAGER

- The last piece to issuing a notification is getting a reference to the NotificationManager system service and calling notify().

```
//6: Notifying the Manager: to notify the user of events that happen.
//1001: is an identifier
//builder.build(): the actual notification
notificationManager.notify(1001, builder.build())
```

```kotlin
//declaring variables
lateinit var notificationManager: NotificationManager
lateinit var notificationChannel: NotificationChannel
lateinit var builder: Notification.Builder

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    btn_1.setOnClickListener {
        //4: pendingIntent is an intent for future use i.e after the notification is clicked, this intent will come into action
        val intent = Intent(this, AfterNotification::class.java)
        val pendingIntent = PendingIntent.getActivity(this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT )

        //1: checking if android version is greater than oreo(API 26) or not
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val channelId = "i.apps.notifications"
            val description = "Test notification"

            //2: Create a unique name for the notification channel
            notificationChannel = NotificationChannel(channelId, description, NotificationManager.IMPORTANCE_DEFAULT)
            notificationChannel.enableLights(true)
            notificationChannel.lightColor = Color.GREEN
            notificationChannel.enableVibration(false)

            //3: Create the channel using the NotificationManager.
            notificationManager = getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager
            notificationManager.createNotificationChannel(notificationChannel)

            //5: create the notification: use Notification.Builder to begin building the notification.
            var builder = Notification.Builder(this, channelId)
                .setSmallIcon(android.R.drawable.ic_dialog_info) // Set a small icon to be display in the notification shade.
                .setContentTitle("Example Notification") // Set a title for the notification.
                .setContentText("This is an  example notification.")//Set content for the notification.
                .setContentIntent(pendingIntent)
                .setChannelId(channelId)
        } else {
            builder = Notification.Builder(this)
                .setSmallIcon(R.drawable.ic_launcher_background)
                .setContentIntent(pendingIntent)
                .setContentTitle("Example Notification")
                .setContentText("This is an  example notification.")
        }
        //6: Notifying the Manager: to notify the user of events that happen.
        notificationManager.notify(1001, builder.build())
    }
}
```

MainActivity.kt



FINAL RESULT