● 알람(notification): 앱의 각종 상황을 사용자에게 알릴 목적으로 이용하는 기능이다. 알람을 띄우기 위한 기본 클래스는 NotificationManager와 Notification이다. Notification 객채에 각종 정보를 담고 NotificationManager로 시스템에 등록하는 구조이다. Notification 객체는 직접 생성되지 않으며 NotificationCompat.Builder로 생성하는데, API Level 26부터는 NotificationChannel에 의해서 Builder가 생성된다. NotificationChannel은 일종의 알람에 대한 관리 단위로, 앱에서 띄우는 알림을 구분할 때 사용한다. NotificationChannel을 적용하면 하나의 앱에서 띄우는 알림을 채널 단위로 구분해서, 사용자가 각각 따로 설정할 수 있게 한다. NotificationChannel은 API Level 26부터 제공하므로, 하위 버전에서 실행되지 않게 Build.VERSION.SDK_INT을 이용하여 버전 분기 프로그램을 작성해야 한다.

NotificationChannel	알림의 관리 단위(Android Oreo에서 추가)
NotificationCompat.Builder	알림을 다양한 정보로 생성
Notification	알림 구성 정보를 가지는 객체
NotificationManager	알림을 시스템에 발생시키는 SystemService

```
// NotificationManager을 getSystemService() 함수를 이용해 얻는다.
NotificationManager manager = (NotificationManager) getSystemService(NOTIFICATION_SERVIC
```

```
NotificationCompat.Builder builder;

if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.0) {
    // 채널 생성
    String ChannelId = "one-channel";
    String channelName = "My Channel One";
    String channelDescription = "My Channel One Description";
    NotificationChannel channel = new NotificationChannel(channelId, channelName, Notifi
    channel.setDescription(channelDescription);
    // 각종 채널에 대한 설정
    channel.enableLights(true);
    channel.setLightColor(Color.RED);
    channel.setVibrationPattern(new long[] {100, 200, 300});
    // NotificationManager에 채널 등록
    manager.createNotificationChannel(channel);
    // 빌더 생성과 생성 시 채널 등록
    builder = new NotificationCompat.Builder(this, channelId);
} else {
    builder = new NotificationCompat.Builder(this);
}
```

▶ 기본적인 알림 구성: Builder의 각종 setter 함수를 이용하여 알림의 구성 정보를 명시해 준다.

setSmallIcon	작은 아이콘 이미지 지정
setWhen	시간
setcontentTitle	확장 내용의 타이틀 문자열
setContentText	확장 내용의 본문 문자열
setAutoCancel	터치 시 자동 삭제 여부, true 값이 지정되면 터치 시 삭제됨
setOngoing	진행표시 여부, true 값이 설정되면 사용자가 손가락으로 밀어서 삭제 불가

```
builder.setSmallIcon(android.R.drawable.ic_notification_overlay);
builder.setWhen(System.currentTimeMillis());
builder.setContentTitle("Content Title");
builder.setContentText("Content Message");
builder.setAutoCancel(true);
```

알림 내용을 구성한 후에는, NotificationManager로 만들어진 알림을 상태바에 등록하면 된다.

```
manager.notify(222, builder.build());
```

notify() 함수의 첫 번째 매개변수는 등록한 알림의 식별자 값으로, 개발자가 임의의 숫자로 지정하여 코드로 알림을 취소할 때 사용된다. 사용자가 알림을 클릭하거나 손으로 밀어서 사라지게 만들 수 있지만, 때로는 자바 코드에서 취소해야 할 때도 있는데, 이때는 cancel() 함수를 이용한다.

```
manager.cancel(222);
```

알람의 확장된 부분을 터치 했을 때, 사용자 이벤트로 처리하여 앱의 특정 액티비티를 실행해야 하는 경우가 많다. 이를 위해서는 사용자 이벤트를 등록해야 한다. 알림은 우리 앱에서 구성하지만, 알림이 보이는 곳은 우리 앱의 액티비티가 아니라 시스템이 제어하는 상태바이므로, 개발자가 자바 코드로 직접 이벤트를 처리할 수 없다. 즉, 알림의 사용자 터치 이벤트는 시스템 화면에서 발생하는 것이므로, 시스템에 이벤트가 발생하면 어떻게 처리해달라고 의뢰하는 방식이다. 이벤트가 발생하면 우리 앱의 엑티비티(또는 서비스, 브로드캐스트 리시버)가 실행되어야 한다. 결국, 이벤트 발생 시 시스템이 우리 앱의 액비티비를 실행해야 하며 이렇게 하는 방법은 인텐트밖에 없다. 알림이 터치될 때 발생시킬 인텐트를 준비하여 시스템에 등록하면, 실제 사용자의 터치 이벤트가 발생하는 순간에 시스템에서 인벤트를 실행하는 구조이다.

```
// 알림 터치 시 사용할 인텐트를 준비한다
Intent intent = new Intent(this. MainAcitivity.class);
```

보통은 인벤트 객체가 준비되면 개발자 코드에서 startActivity()등의 함수로 직접 발생시킴녀 되지만, 인텐트를 발생시키는 곳이 시스템이므로 준비된 인텐트 발생을 시스템에 의뢰해야 한다. 인텐트 발생 의뢰는 인텐트 객체 이외에 부가 정보가 더 설정되어야 하는데, 이 정보들을 표현하는 클래스가 PendingIntent이다. 이 클래스는 인텐트 발생을 누군가에게 의뢰해야 하는 여러 곳에서 사용된다.

```
PendingIntent ipIntent = PendingIntent.getAcitivty(this, 10, intent, PendingIntent.FLAG_
```

PendingIntent.getActivity()혹은 getBroadcast(), getService() 함수로 만들며, 함수의 매개변수로 의뢰 정보를 설정한다. 두 번째 매개변수는 requestCode 값으로 의뢰를 식별하기 위해 사용되며, 코드에서 의뢰를 취소할 때 사용된다. 세 번째 매개변수는 의뢰하고자 하는 인텐트 객체이며, 네 번째 매개변수는 flag값이다. Android 12 버전 이상에서는 FLAG_MUTABLE 혹은 FLAG_IMMUTABLE 상수 변수로 지정한다. PendingIntent로 사용자 터치 시 인텐트 발생 의뢰를 위한 객체가 준비되었다면, Builder의 setContentIntent() 함수의 매개변수에 지정해 주면 된다.

```
builder.setContentIntent(pIntent);
```

● 알림의 다양한 구성

▶ Action — 간단한 이벤트 처리: 알림 자체는 일종의 메시지이며 사용자가 알림을 터치할 때 상세보기 화면 등을 실행하여 간단한 서비스를 제공하는 구조이다. 그런데 때로는 화면 전환 없이 알림 내에서 간단한 터치 이벤트를 처리하기도 한다. 예를 들면 스크린 캡처 후 스크린샷을 공유, 편집, 삭제하는 기능을 갤러리 앱을 실행시키지 않고도 알림에서 간단하게 처리할 수 있다. 이처럼 사용자의 추가 이벤트를 처리하는 액션(action)을 알림에서 제공할 수 있다. 추가 이벤트는 addAction()함수로 설정하며, 내용 구성은 NotificationCompat.Action.Builder로 만든다. 아이콘과 문자열, 그리고 액션 클릭 시 인텐트 발생을 의뢰할 PendingIntent 정보로 구성된다. 추가 이벤트는 addAction() 함수로 설정하며, 내용 구성은 NotificationCompat.Action.Builder로 만든다. 아이콘과 문자열, 그리고 액션 클릭 시 인텐트 발생을 의뢰할 PendingIntent 정보로 구성된다.

```
builder.addAction(new NotificationCompat.Action.Builder(android.R.drawable.ic_menu_share
```

▶ setLageIcon — 큰 아이콘: 알림 화면에 보이는 작은 아이콘 이외에 setLargeIcon() 함수로 큰 아이콘을 설정할 수도 있다.

```
Bitmap largeIcon = BitmapFactory.decodeResource(getResources().R.drawable.noti_large);
builder.setLargeIcon(largeIcon);
```

▶ BigPictureStyle — 큰 이미지: 알림 화면에서 ContentTitle, ContentText을 이용한 문자열 이외에도 다양한 콘텐츠로 구성할 수 있다. 이를 지원하기 위한 것이 Style이며, 여러 스타일을 제공한다. 그 중 BigPictureStyle은 알림에 큰 크기의 이미지를 출력하기 위한 스타일이다. NotificationCompat.BigPictureStlye의 bigPicture() 함수의 매개변수로 준비한 Bitmap 객체를 설정하고, 다시 Builder의 setStyle() 함수로 준비한 스타일을 적용하는 구조이다.

```
NotificationCompat.BigPictureStyle bigStyle = new NotificationCompat.BigPictureStyle(bui
Bitmap bigPicture = BitmapFactory.decodeResource(getResources().R.drawable.big_picture);
bigStyle.bigPicture(bigPicture);
builder.setStyle(bigStyle);
```

▶ BigTextStyle — 긴 문자열: 알림에서 액티비티로 화면을 전환하지 않고도 긴 문자열을 출력할 수 있다. 이를 지원하기 위한 스타일이 BigTextStyle이다.

```
NotificationCompat.BigTextStyle bigTextStyle = new NotificationCmpat.BigTextStyle(builde
bigTextStyle.setSummartText("BigText Summary");
bigTextStyle.setBigContentTitle("BigText Title");
bigTextStyle.bigText(getResources().getText(R.string.big_text));
builder.setStyle(bigTextStyle);
```

▶ InboxStyle — 목록: 알림에 여러 데이터를 목록 형태로 제공해야 할 때도 있는데, 이를 지원하기 위한 스타일이다. InboxStyle에 addLine() 함수로 항목을 설정하여 사용한다.

```
NotificationCompat.InboxStyle style = new NotificationCompat.InboxStyle(builder);
style.addLine("Activity");
style.addLine("BroadcastReceiver");
style.addLine("Service");
style.addLine("ContentProvider");
style.setSummaryText("Android Component");
builder.setStyle(style);
```

▶ Progress — 프로그레스: 알림에 프로그레스바로 작업의 진행 사항을 표시해야 할 때 사용한다. 프로그레스바는 특별한 스타일로 제공되지 않고, Builder의 setProgress() 함수로 progress 값을 대입해 주면 자동으로 나타난다. 파일을 내려받거나 음악 재생 등을 표시할 때 자주 이용된다. 보통 프로그레스바를 알림에 표션하면 setAutoCancel() 함수를 false로, setOngoing() 함수를 true로 설정하여 사용자가 알림을 없애지 못하게 하며, 코드에서 cancel() 함수로 제거한다.

```
Thread t = new Thread(() -> {
    for (int i = 1; i <= 10; i++) {
        builder.setAutoCancel(false);
        builder.setOnging(true);
        builder.setProgress(10, i, false);
        manager.notify(222, builder.build());
        if (i >= 10) {
            manager.cancel(222);
        }
        SystemClock.sleep(1000);
    }
});
t.start();
```

▶ Message Style — 메시지 스타일: 여러 사람이 주고받은 메시지를 구분해서 출력할 때 사용한다. Message 객체로 생성되며, 3가지 정보를 보여준다.

```
Message(CharSequence text, long timestamp, Person person)
```

첫 번째 매개변수는 메시지 내용이며, 두 번째 매개변수는 메시지가 발생한 시각이고, 세 번째 매개변수는 Person 객체로 어떤 사람이 보냈는지에 대한 정보를 위한 것이다. Person은 메시지 스타일 알림에 출력될 한 사람의 정보를 담는 클래스이다. 각 Person 객체에 이름, 아이콘 등을 설정할 수 있다.

```
Person sender1 = new Person.Builder()
        .setName("kkang")
        .setIcon(IconCompat.createWithResource(this, R.drawable.person1))
        .build();
Person sender2 = new Person.Builder()
        .setName("kim")
        .setIcon(IconCompat.createWithResource(this, R.drawable.person2))
        .build();
```

이 Person 객체를 MessageStlye을 이용해 알림에 출력할 수 있다.

```
NotificationCompat.MessagingStyle.Message message = new NotificationCompat.MessagingStyl
NotificationCompat.MessagingStyle style = new NotificationCompat.MessagingStyle(sender1)

builder.setStyle(style)
```

▶ RemoteInput — 원격 입력: 알림에 글을 직접 입력할 수 있도록 하는 기능이다. 예를 들면 사용자가 답장을 할 때 알림에서 바로 글을 입력할 수 있도록 지원하는 것이 액션의 일종인 원격입력이다. 원격 입력을 구현하기 위해서는 RemoteInput을 이용하며, RemoteInput에 사용자에게 입력 받을 정보를 설정한 후에 액션을 추가하는 구조이다.

```
String replyKey = "message";
String replayLabel = "답장";
// RemoteInput.Builder에 정보를 설정한3다. 매개변수는 사용자 입력을 식별하는 값으로, 임의의 문
RemoteInput.Builder remoteBuilder = new RemoteInput.Builder(replyKey);
remoteBuilder.setLabel(replyLabel); // setLabel 함수로 설정하는 값은 입력 화면에 출력되는 힌
// RemoteInput 객체를 생성한다. 이때, 버전호환성을 위해 androidx.core.app.RemoteInput을 이용
RemoteInput remoteInput = remoteBuilder.build();

//RemoteInput도 액션의 일종이므로, 터치 이벤트를 처리하는 Pendingntent가 준비되어야 한다.
Intent remoteIntent = new Intent(this, MyReceiver.class);
// 이때 사용하는 PendingIntent 생성자의 세 번째 매개변수 값은 FLAG_MUTABLE로 지정한다.
// PendingIntent로 액티비티, 서비스, 브로드캐스트 리시버 등을 실행할 수 이씻만, 대부분 사용자
// 화면 전환 없이 처리할 때 원격입력을 사용하므로, 브로드캐스트 리시버에 많이 적용된다.
PendingIntent remotePendingIntent = PendingIntent.getBroadcast(this, 20, remoteIntent, F

// 액션에 RemoteInput을 적용한다.
NotificationCompat.Action.Builder remoteActionBuilder = new NotificationCompat.Action.Bu
remoteActionBuilder.addRemoteInput(remoteInput);
NotificationCompat.Action remoteAction = remoteActionBuilder.build();

builder.addAction(remoteAction);
```

RemoteInput에 의해 아래의 코드로 입력된 문자열을 획득할 수 있다. 아래의 코드는 브로드캐스트 리시버가 실행되었다는 가정 하에 작성되었다. getCharSequence() 함수의 매개변수는 RemoteInput을 등록할 때 지정했었던 식별자 문자열이다. 문자열을 획득한 후에는 NotificationManager로 cancel(), 혹은 notify() 함수를 호출해 신호를 잘 받았다고 알려주어야 한다. 이때 cancel()함수의 매개변수는 알림의 식별자로, RemoteInput을 이용한 곳에서 알림을 띄울 때 사용했던 식별자와 같아야 한다.

```
public class MyReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        String replyTxt = RemoteInput.getResultsFromIntent(intent).getCharSequence("mess
            Log.d("kkang", "receiver...." + replyTxt);
            NotificationManager manager = (NotificationManger)context.getSystemService(C
            manager.cancel(222);
    }
}
```

```
// 원격 입력을 테스트하기 위한 브로드캐스트 리시버 만들기
패키지 > New > Other > Broadcast Receiver > 생성
```

```
// activity_main.xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.androic
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/bigPictureButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="100dp"
        android:text="BigPicture"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/progressButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:text="Progress"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/bigPictureButton" />

    <Button
        android:id="@+id/messageButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:text="Message"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/progressButton" />

    <Button
        android:id="@+id/remoteButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:text="Remote"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/messageButton" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

```java
// NotiReceiver.java
package com.android.practicenotification;

import android.app.NotificationManager;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;

import androidx.core.app.RemoteInput;

public class NotiReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        String replyTxt = RemoteInput.getResultsFromIntent(intent).getCharSequence("mess
        Log.d("kkang", "receiver...." + replyTxt);
        NotificationManager manager = (NotificationManager) context.getSystemService(Cor
        manager.cancel(222);
    }
}
```

```java
// MainActivity.java
package com.android.practicenotification;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.NotificationCompat;
import androidx.core.app.Person;
import androidx.core.app.RemoteInput;
import androidx.core.graphics.drawable.IconCompat;

import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.os.Build;
import android.os.Bundle;
import android.os.SystemClock;
import android.view.View;

import com.android.practicenotification.databinding.ActivityMainBinding;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    ActivityMainBinding binding;
    NotificationManager manager;
    NotificationCompat.Builder builder;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        binding.bigPictureButton.setOnClickListener(this);
        binding.progressButton.setOnClickListener(this);
        binding.messageButton.setOnClickListener(this);
        binding.remoteButton.setOnClickListener(this);
    }
```

```java
@Override
    public void onClick(View v) {
        manager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            // 채널 생성
            String channelId = "one-channel";
            String channelName = "My Channel One";
            String ChannelDescription = "My Channel One Description";
            NotificationChannel channel = new NotificationChannel(channelId, channelName
            channel.setDescription(ChannelDescription);

            // 각종 채널에 대한 설정
            channel.enableLights(true);
            channel.setLightColor(Color.RED);
            channel.enableVibration(true);
```

```
        channel.setVibrationPattern(new long[] {100, 200, 300});
        manager.createNotificationChannel(channel);
        // 채널이 등록된 빌더
        builder = new NotificationCompat.Builder(this, channelId);
    } else {
        builder = new NotificationCompat.Builder(this);
    }


    builder.setSmallIcon(android.R.drawable.ic_notification_overlay);
    builder.setWhen(System.currentTimeMillis());
    builder.setContentTitle("메시지 도착");
    builder.setContentText("안녕하세요");
    builder.setAutoCancel(true);
    Bitmap largeIcon = BitmapFactory.decodeResource(getResources(), R.drawable.noti_
    builder.setLargeIcon(largeIcon);


    if (v == binding.bigPictureButton) {
        // BigPictureStyle
        Bitmap bigPicture = BitmapFactory.decodeResource(getResources(), R.drawable.
        NotificationCompat.BigPictureStyle bigStyle = new NotificationCompat.BigPict
        bigStyle.bigPicture(bigPicture);
        builder.setStyle(bigStyle);
    } else if (v == binding.progressButton) {
        // Progress 알림
        Thread t = new Thread(() -> {
            for (int i = 1; i <= 10; i++) {
                builder.setAutoCancel(false);
                builder.setOngoing(true);
                builder.setProgress(10, i, false);
                manager.notify(222, builder.build());
                if (i >= 10) {
                    manager.cancel(222);
                }
                SystemClock.sleep(1000);
            }
        });
        t.start();
    } else if (v == binding.messageButton) {
        // MessageStyle
        // androidx.core.app.Person 이용
        Person sender1 = new Person.Builder().setName("kkang").setIcon(IconCompat.cr
        Person sender2 = new Person.Builder().setName("kim").setIcon(IconCompat.crea

        NotificationCompat.MessagingStyle.Message message = new NotificationCompat.M
        NotificationCompat.MessagingStyle style = new NotificationCompat.MessagingSt
        builder.setStyle(style);
    } else if (v == binding.remoteButton) {
        // RemoteInput
        String replyKey = "message";
        String replyLabel = "답장";
        // android.core.app.RemoteInput 이용
        RemoteInput.Builder remoteBuilder = new RemoteInput.Builder(replyKey);
        remoteBuilder.setLabel(replyLabel);
        RemoteInput remoteInput = remoteBuilder.build();

        Intent remoteIntent = new Intent(this, NotiReceiver.class);
        PendingIntent remotePendingIntent = PendingIntent.getBroadcast(this, 20, rem
        NotificationCompat.Action.Builder remoteActionBuilder = new NotificationComp
```

```
            remoteActionBuilder.addRemoteInput(remoteInput);
            NotificationCompat.Action remoteAction = remoteActionBuilder.build();;
            builder.addAction(remoteAction);
        }
        manager.notify(222, builder.build());
    }

}
```

Resources: 깡쌤의 안드로이드 프로그래밍