Lecture 8: Context, Notification and Service

Jianjun Chen (Jianjun.Chen@xjtlu.edu.cn)

Context

- “Interface to global information about an application environment. This is an abstract class whose implementation is provided by the Android system. It allows access to application-specific resources and classes, as well as up-calls for application-level operations such as launching activities, broadcasting and receiving intents, etc.”

- We have seen this many times.

Intent intent = new Intent(MyActivity.this, …).

Why Having Context?

- Your app can be installed on different devices, they may provide different hardware (like sensors, GPS unit etc.).
- Users can control the permissions of your app, there must be a place to find out these permissions.
- Allow android components like Activities and Services to obtain shared information of the current app.
- Many more… So check the API while you are learning this module.

Context

- “It allows access to application-specific resources and classes”

`checkPermission(String permission, int pid, int uid)`	Determine whether the given permission is allowed for a particular process and user ID running in the system.
`checkSelfPermission(String permission)`	Determine whether *you* have been granted a particular permission.

`getPackageName()`	Return the name of this application's package.

`final <T> T`	`getSystemService(Class<T> serviceClass)` Return the handle to a system-level service by class.
`abstract Object`	`getSystemService(String name)` Return the handle to a system-level service by name.

`getSystemService(Class<T> servi`

Return the handle to a system-level serv

`getSystemService(String name)`

Return the handle to a system-level serv

Available in the official android API document

`NOTIFICATION_SERVICE` ("notification")

A `NotificationManager` for informing the user of background events.

`KEYGUARD_SERVICE` ("keyguard")

A `KeyguardManager` for controlling keyguard.

`LOCATION_SERVICE` ("location")

A `LocationManager` for controlling location (e.g., GPS) updates.

`SEARCH_SERVICE` ("search")

A `SearchManager` for handling search.

`VIBRATOR_SERVICE` ("vibrator")

A `Vibrator` for interacting with the vibrator hardware.

`CONNECTIVITY_SERVICE` ("connection")

A `ConnectivityManager` for handling management of network connections.

`WIFI_SERVICE` ("wifi")

A `WifiManager` for management of Wi-Fi connectivity.

`WIFI_P2P_SERVICE` ("wifip2p")

Context

- It allows …, as well as up-calls for application-level operations

`abstract boolean`	`bindService(Intent service, ServiceConnection conn, int flags)` Connect to an application service, creating it if needed.

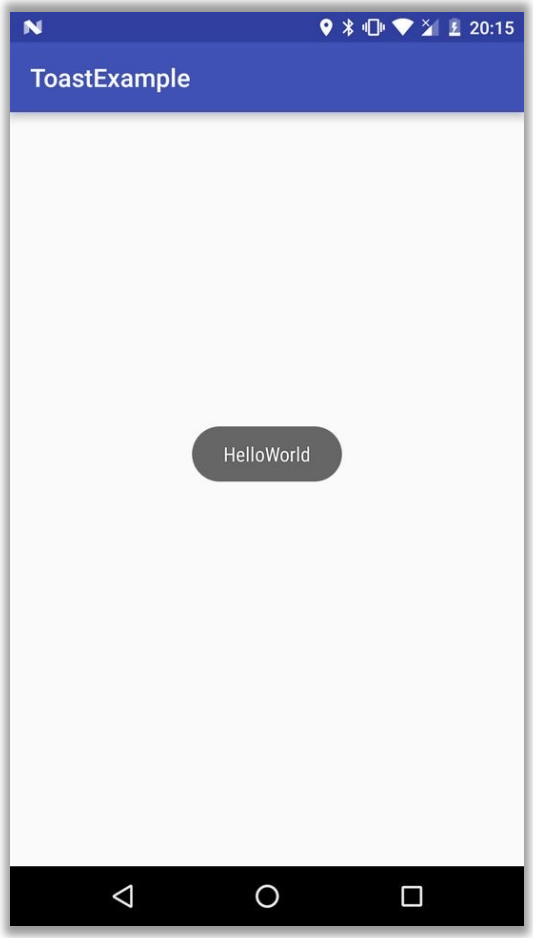
- We will learn about service in this lecture.

Notifications

Toasts, Notifications

Toast

- Toast is a lightweight notification that
  - Shows a simple message in the centre of the screen for a brief while.
  - Does not interrupt the running of the current app.
- Users cannot interact with a toast.



Showing a Toast

- Create an `Toast` object using `Toast.makeText(context, message, duration)`.
  - Duration: Toast.***LENGTH_LONG*** or Toast.***LENGTH_SHORT***
- Optionally, adjust the toast position with `setGravity()`
- Then, Call `show()` method.

```
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      Toast t = Toast.makeText(MainActivity.this, "Toasted", Toast.LENGTH_SHORT);
      t.setGravity(Gravity.FILL_HORIZONTAL,0,0);
      t.show();
   }
}
```

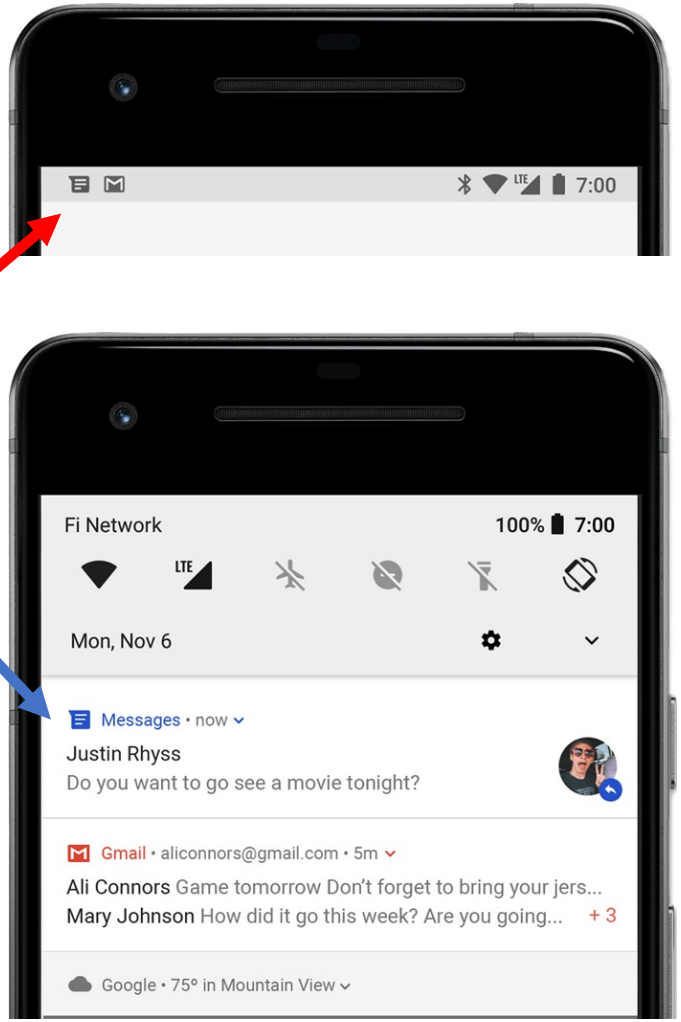
Gravity settings

- They are constants defined inside the `Android.view.Gravity` class.

int	BOTTOM Push object to the bottom of its container, not changing its size.
int	CENTER Place the object in the center of its container in both the vertical and horizontal axis, not changing its size
int	CENTER_HORIZONTAL Place object in the horizontal center of its container, not changing its size.
int	CENTER_VERTICAL Place object in the vertical center of its container, not changing its size.
int	CLIP_HORIZONTAL Flag to clip the edges of the object to its container along the horizontal axis.
int	CLIP_VERTICAL

Notification

- Notifications are shown:
  - In the status bar on the top of the screen.
  - In the notification drawer.



- A notification can be configured to allow users to tap it to go to a specified activity.

Creating a Notification

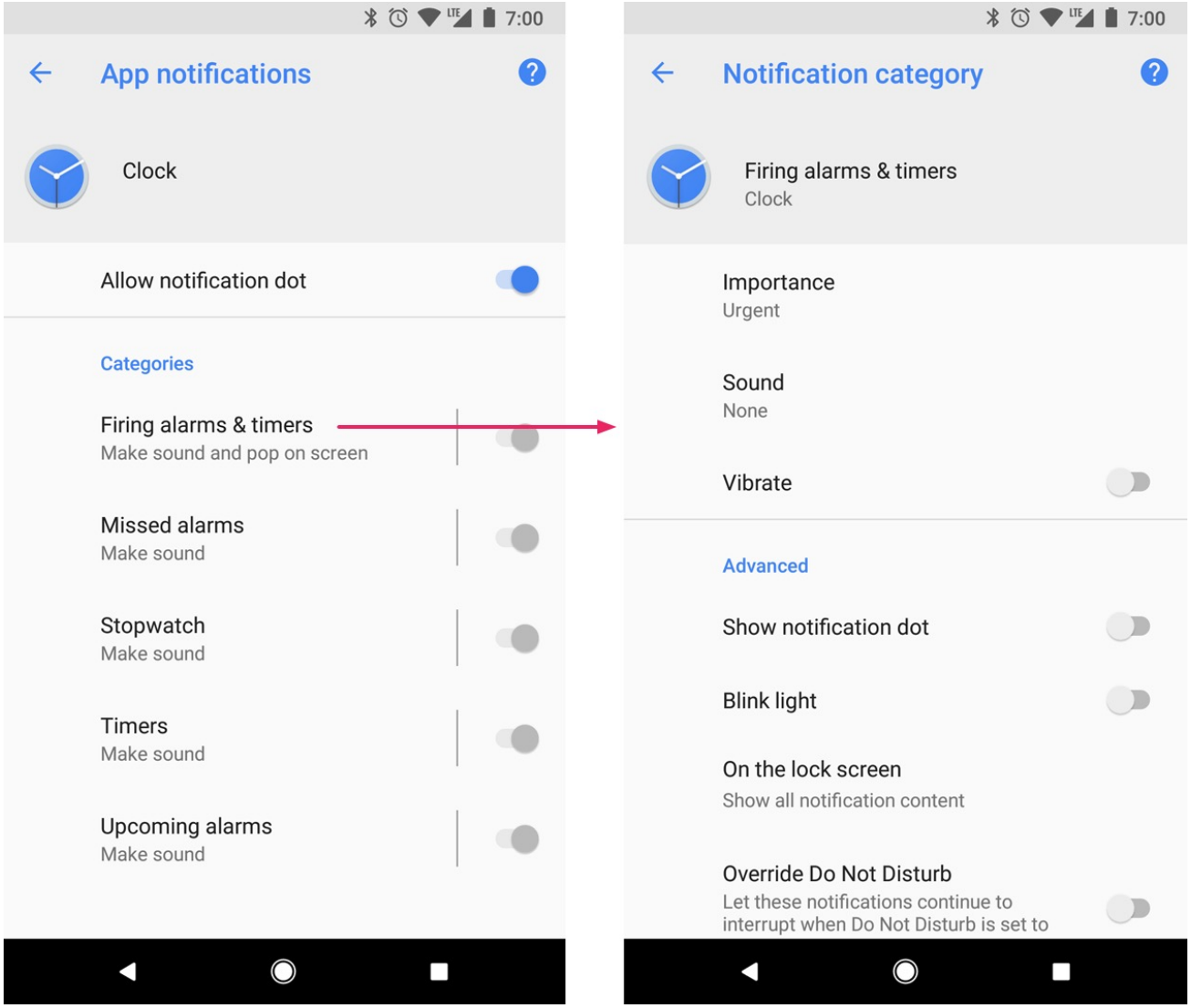
1. First create a notification channel if the target Android version is 8 (API level 26) or above.

2. Create and configure the notification content and style.

3. Optionally, set the notification's tap action.

What is a Notification Channel?

- Considering a message app that can send and receive messages from friends.
  - When a message fails to send, the app makes a sound and shows a notification.
  - When a message arrives, the app makes a different sound and shows a notification.
- Notification channel allows an app to have different styles of notifications, for different situations.


7:00
App notifications
Clock
Allow notification dot
Categories
Firing alarms & timers
Make sound and pop on screen
Missed alarms
Make sound
Stopwatch
Make sound
Timers
Make sound
Upcoming alarms
Make sound
7:00
Notification category
Firing alarms & timers
Clock
Importance
Urgent
Sound
None
Vibrate
Advanced
Show notification dot
Blink light
On the lock screen
Show all notification content
Override Do Not Disturb
Let these notifications continue to interrupt when Do Not Disturb is set to

Why Notification Channel?

- Why we need to create channels first? Why not letting the app to decide the settings of a notification every time it is fired?
  - All channels are registered in the system. Users can override the channel behaviours later.
  - Apps can no longer control the behaviours of a channel once created.
  - Prevents badly-designed apps from annoying users,

Call this function early in your app. Such as in `onCreate()`

This is “oh”, not zero. It means Android Oreo, Version 8, API level 26

```
private void createNotificationChannel() {
  if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    NotificationChannel channel1 = new NotificationChannel(CHANNEL_IDS[0],
                "Channel 1 (channel name)",
                NotificationManager.IMPORTANCE_HIGH);
```

https://developer.android.google.cn/training/notify-user/channels.html#importance

This is a String that needs to be defined by yourself. It is needed later.

```
        channel1.setDescription("description 1");

        // Register the channel, you can't change the importance
        // or other notification behaviors after this
        NotificationManager notificationManager =
                            getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel1);
    }
}
```

You CANNOT create a notification channel with the same ID again. Newer requests will just be ignored.

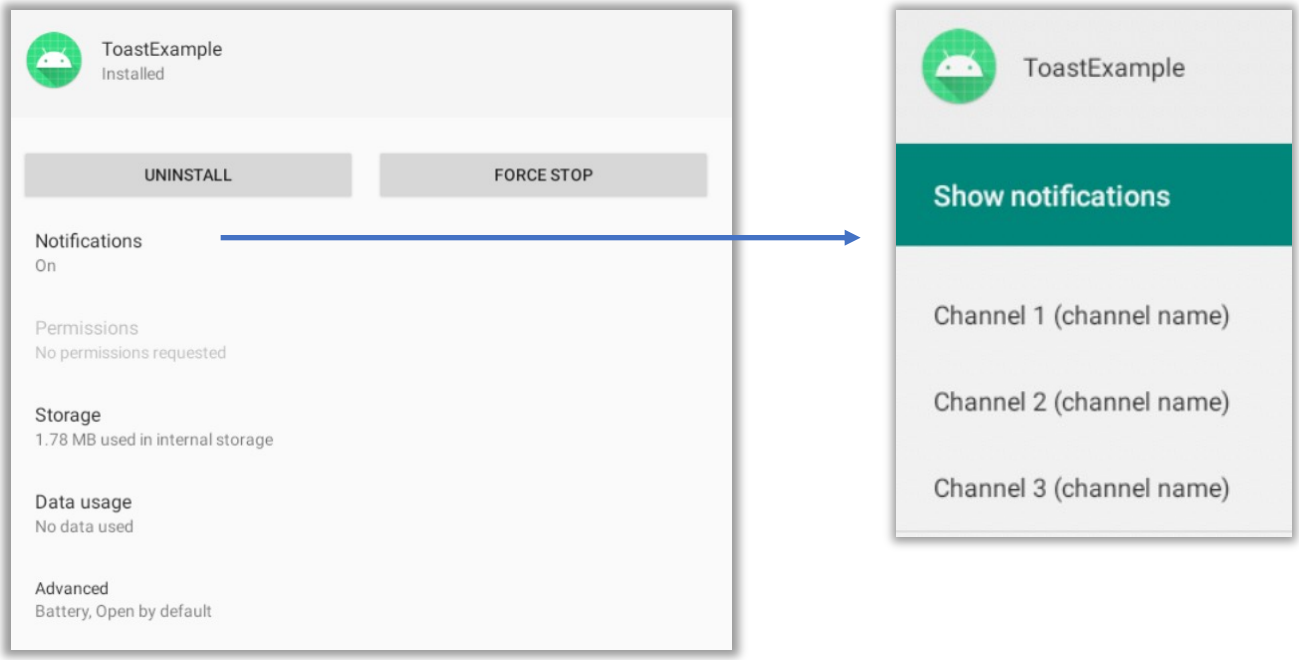
Creating Channels

```java
final String[] CHANNEL_IDS = {"CH1", "CH2", "CH3"};
private void createNewMessageNotificationChannel() {
    // Create the NotificationChannel, but only on API 26+ because
    // the NotificationChannel class is new and not in the support library
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        NotificationChannel channel1 =
                new NotificationChannel(CHANNEL_IDS[0],
                        name: "Channel 1 (channel name)",
                        NotificationManager.IMPORTANCE_HIGH);
        channel1.setDescription("description 1");

        NotificationChannel channel2 =
                new NotificationChannel(CHANNEL_IDS[1],
                        name: "Channel 2 (channel name)",
                        NotificationManager.IMPORTANCE_DEFAULT);
        channel2.setDescription("description 2");

        NotificationChannel channel3 =
                new NotificationChannel(CHANNEL_IDS[2],
                        name: "Channel 3 (channel name)",
                        NotificationManager.IMPORTANCE_MIN);
        channel3.setDescription("description 3");

        // Register the channel with the system; you can't change the importance
        // or other notification behaviors after this
        NotificationManager notificationManager = getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel1);
        notificationManager.createNotificationChannel(channel2);
        notificationManager.createNotificationChannel(channel3);
    }
}
```

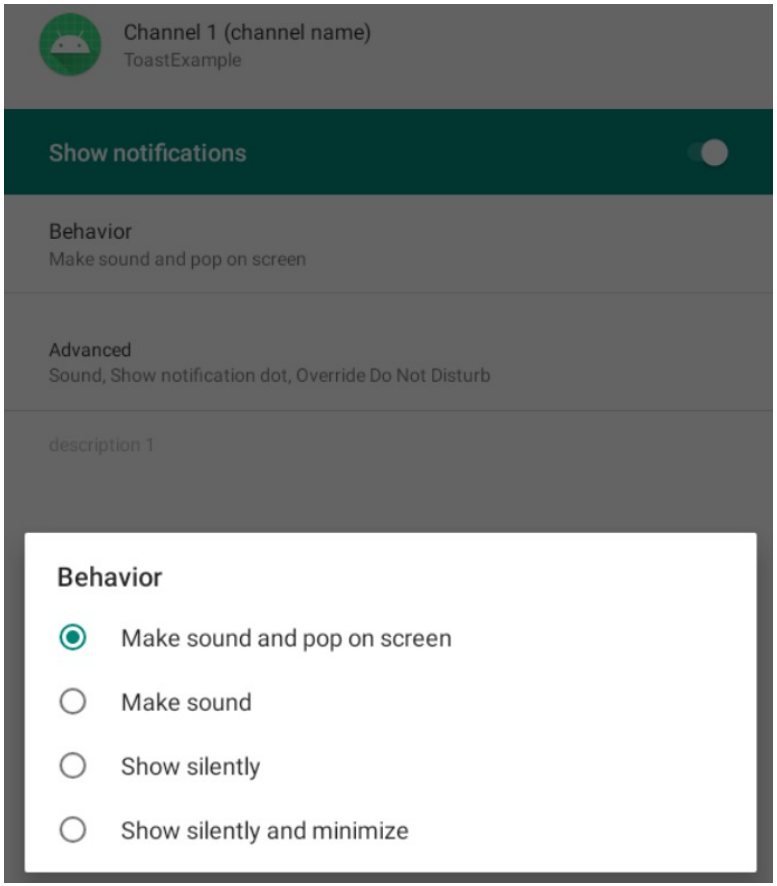
All channels

Channels in the System Settings

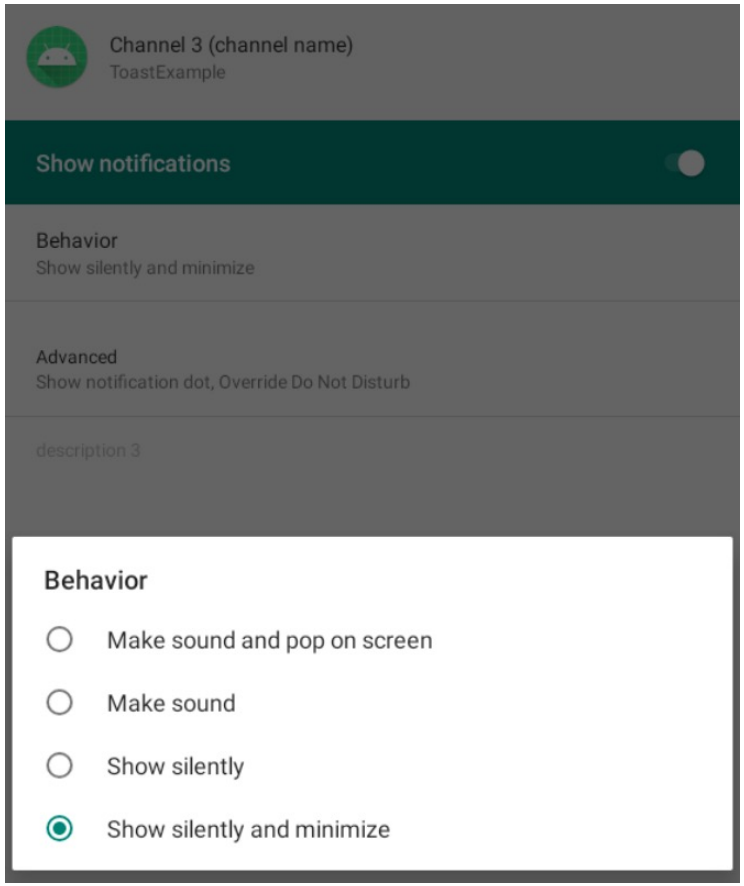
- Once this function is run, three channels will be created for this app.
- Users can change the behaviours of each channel.



Channel Importance Level



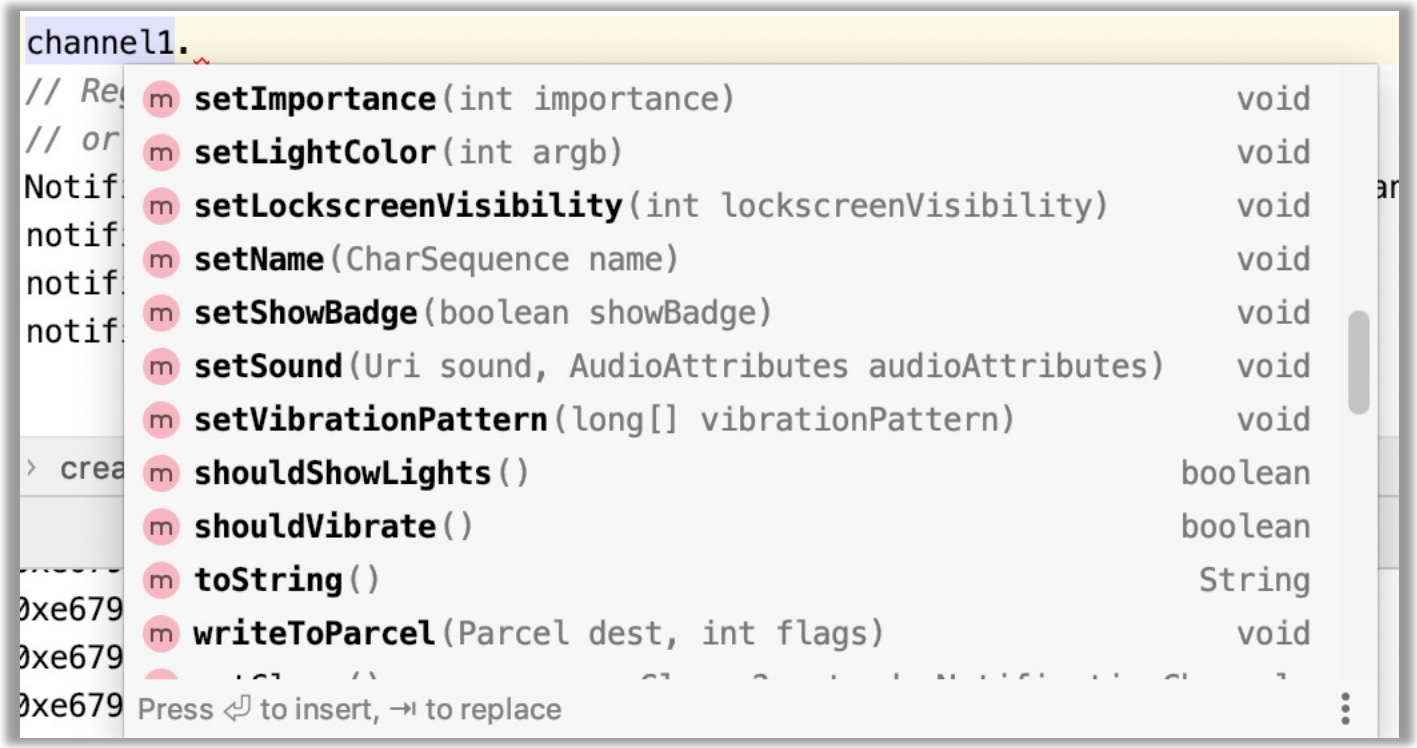
NotificationManager.***IMPORTANCE_HIGH***



NotificationManager.***IMPORTANCE_MIN***

Other Channel Settings

- There are many settings available. Please check the functions of the NotificationChannel class.



Preparing Notification Content

Notification is shown when this button is clicked

```
createNotificationChannel();
Button notif = findViewById(R.id.notif_btn);
notif.setOnClickListener(new Button.OnClickListener() {
  @Override
  public void onClick(View v) {
    Notification.Builder builder =
        new Notification.Builder(MainActivity.this, CHANNEL_IDS[0]);
```

Context

This function is for API 26 and higher. You can use
`new Notification.Builder(MainActivity.this);`
for lower versions.

That's the channel we created

```
    builder.setSmallIcon(R.mipmap.ic_launcher);
    builder.setContentTitle("Notification title");
    builder.setContentText("My app's notification content");
    ......
```

Setting Notification's Tap Action

- You can ask the notification to redirect the user to a certain activity:

```
Intent redir = new Intent(MainActivity.this,
Activiteee.class);
PendingIntent pendingIntent =
        PendingIntent.getActivity(MainActivity.this,
                0, redir, 0);

builder.setContentIntent(pendingIntent);
```

getActivity

Added in API level 1

PendingIntent getActivity (Context context,
int requestCode,
Intent intent,
int flags)

Retrieve a PendingIntent that will start a new activity, like calling `Context.startActivity(Intent)`. Note that the activity will be started outside of the context of an existing activity, so you must use the `Intent.FLAG_ACTIVITY_NEW_TASK` launch flag in the Intent.

PendingIntent

- "…can be handed to other applications so that they can perform the action you described on your behalf at a later time."
  - PendingIntent still exists even if the app that created it is ended.

- "By giving a PendingIntent to another application, you are granting it the right to perform the operation you have specified as if the other application was yourself (with the same permissions and identity)"

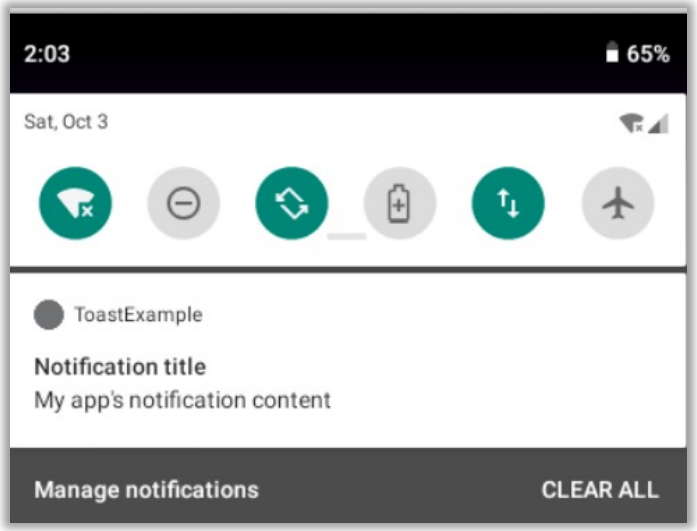
Showing/Cancelling a Notification

```
Notification notification = builder.build();
NotificationManager manager =
        (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
manager.notify(1, notification);
```

Show notification, using id “1”.

```
// after a long time
manager.cancel(1);
```

Cancel notification, using id “1”. IDs must match



More Details about Notifications

- https://developer.android.google.cn/guide/topics/ui/notifiers/notifications?hl=en

Service

startService, bindService

Service

- A Service is an application component
  - to perform a longer-running operation while not interacting with the user, or
  - to supply functionality for other applications to use.
- Services also run on UI thread.
  - Thus, processes that require long waiting time should still run in a separate thread.
- Common example usages:
  - Manage network transactions
  - Manage music playing at the background.
  - The actual work should still be run by threads.

Creating a Service

- Create a class that extends `Service`.
  - `android.app.Service`.
- Override callback functions of `Service`.
- Register your class inside AndroidManifest.

```
<service
   android:name=".MyService"
   android:enabled="true"></service>
```

Class name: MyService

app
manifests
AndroidMa
java
cn.edu.xjtl
Main2A
MainAct
MyServi
ThreadT
cn.edu.xjtl
cn.edu.xjtl
generatedJav
res
drawable
layout
activity_
activity_
activity_
layout.x
layout3.
mipmap
values

New
Link C++ Project with Gradle
Cut ⌘X
Copy ⌘C
Copy Path ⇧⌘C
Copy Relative Path ⌥⇧⌘C
Paste ⌘V
Find Usages ⌥F7
Find in Path... ⇧⌘F
Replace in Path... ⇧⌘R
Analyze
Refactor
Add to Favorites
Show Image Thumbnails ⇧⌘T
Reformat Code ⌥⌘L
Optimize Imports ^⌥O
Run 'All Tests' ^⇧R
Debug 'All Tests' ^⇧D
Run 'All Tests' with Coverage
Create 'All Tests'...
Reveal in Finder
Open in Terminal
Local History
Synchronize 'app'
Edit Scopes...
Compare With... ⌘D

Module
Android Resource File
Android Resource Directory
Sample Data Directory
File
Scratch File ⇧⌘N
Directory
C++ Class
C/C++ Source File
C/C++ Header File
Image Asset
Vector Asset
module-info.java
.editorconfig file
Kotlin Script
Gradle Kotlin DSL Build Script
Gradle Kotlin DSL Settings
Edit File Templates...
AIDL
Activity
Android Auto
Folder
Fragment
Google
Other
Service
UI Component

Service
Service (IntentService)

rvice.class);
!");
,peer=0x15540020,"Signal Cat

Run: app
I/System.
I/System.
I/System.
I/.xjtlu.
I/.xjtlu.
I/System.
I/System.
I/System.

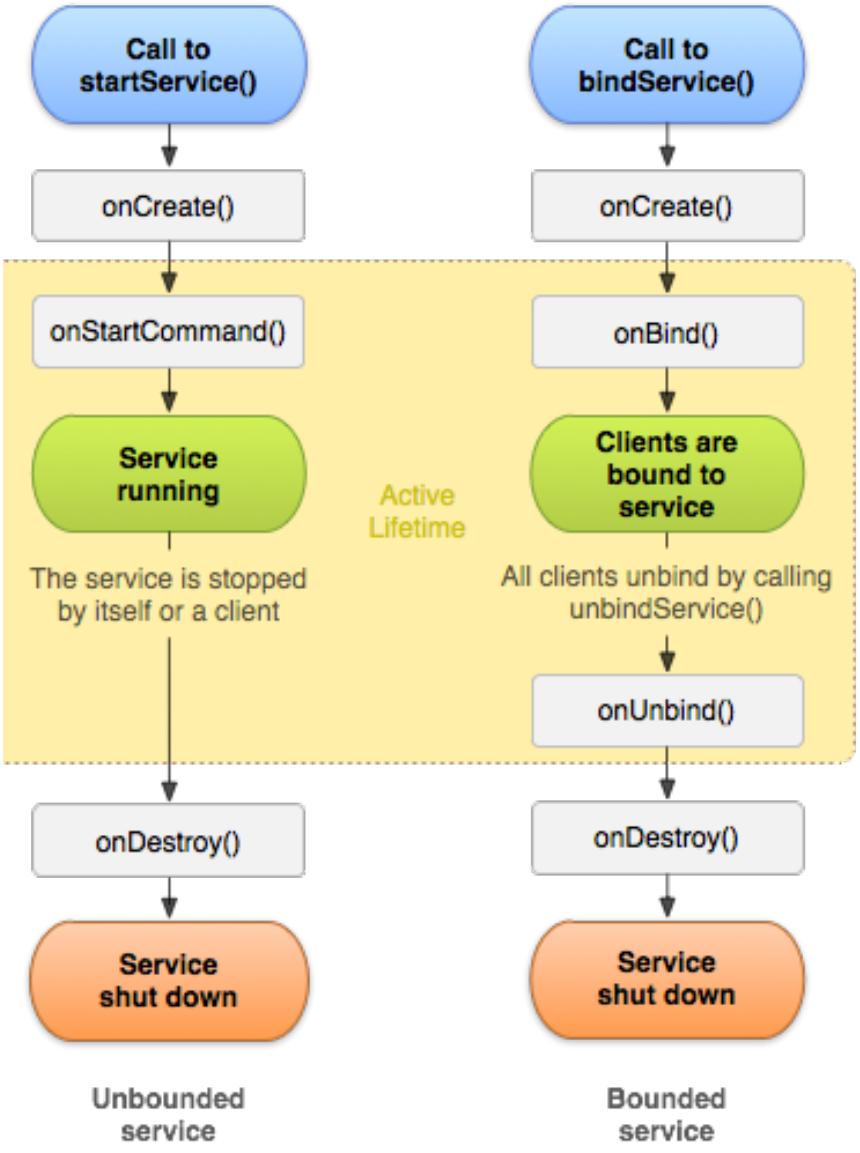
Service

- Two forms of services:

  `Context.startService()`

  `Context.bindService()`

- They lead to two different callback sequences.



startService VS. bindService

- `startService()`:
  - This service can run in the background indefinitely, even if the component that started it is destroyed.
  - It lacks methods to interact with other components.

- `bindService()`:
  - This service only lives while it serves another component.
  - If its last component unbinds, the service will be destroyed by the Android System.
  - It provides mechanisms to interact with callers.

Foreground Services

- A non-foreground service will be stopped if its app was destroyed due to low memory.
  - That's the kind of services created by default.

- If you want a service that is never get killed, use foreground service: https://developer.android.com/guide/components/services

Form 1: Starting a Service

```
Intent intent = new Intent(this, MyService.class);
startService(intent);
```

Remember to implement other call-back functions.

```
public class MyService extends Service {
    public int onStartCommand(Intent intent, int flags, int startId) {
        int res = super.onStartCommand(intent, flags, startId);
        ......
        return res;
    }
    @Override
    public IBinder onBind(Intent intent) {
        throw new UnsupportedOperationException("Not yet implemented");
    }
......
}
```

`onBind()` will not be called when using `startService()`, but you must implement this abstract method.

Form 1: Stopping a Service

You can manually stop a started service by calling

- `stopService(intent)`

```
Intent intent = new Intent(this, MyService.class);
startService(intent);
stopService(intent);
```

- `stopSelf()` inside the Service class

Service VS Thread

- In the previous example, Can we use a thread instead of using service?
  - Possible, but lacks functionalities and the code will be complicated.
- Strengths of using service:
  - Service have life circle functions and can be controlled by any activity. (Even from a different app!)
  - Service can receive broadcast messages. This allows you to adjust app’s behaviours according to some system events (battery life etc.).

Form 2: bindService

- In the next few slides, we will try to create a media player that plays an MP3 file located inside “res/raw”.
- The service will use `MediaPlayer` to play the MP3.
- In our activity (the client), we have a button. When this button is clicked, the client will ask the service to play the MP3 for us.

Form 2: bindService

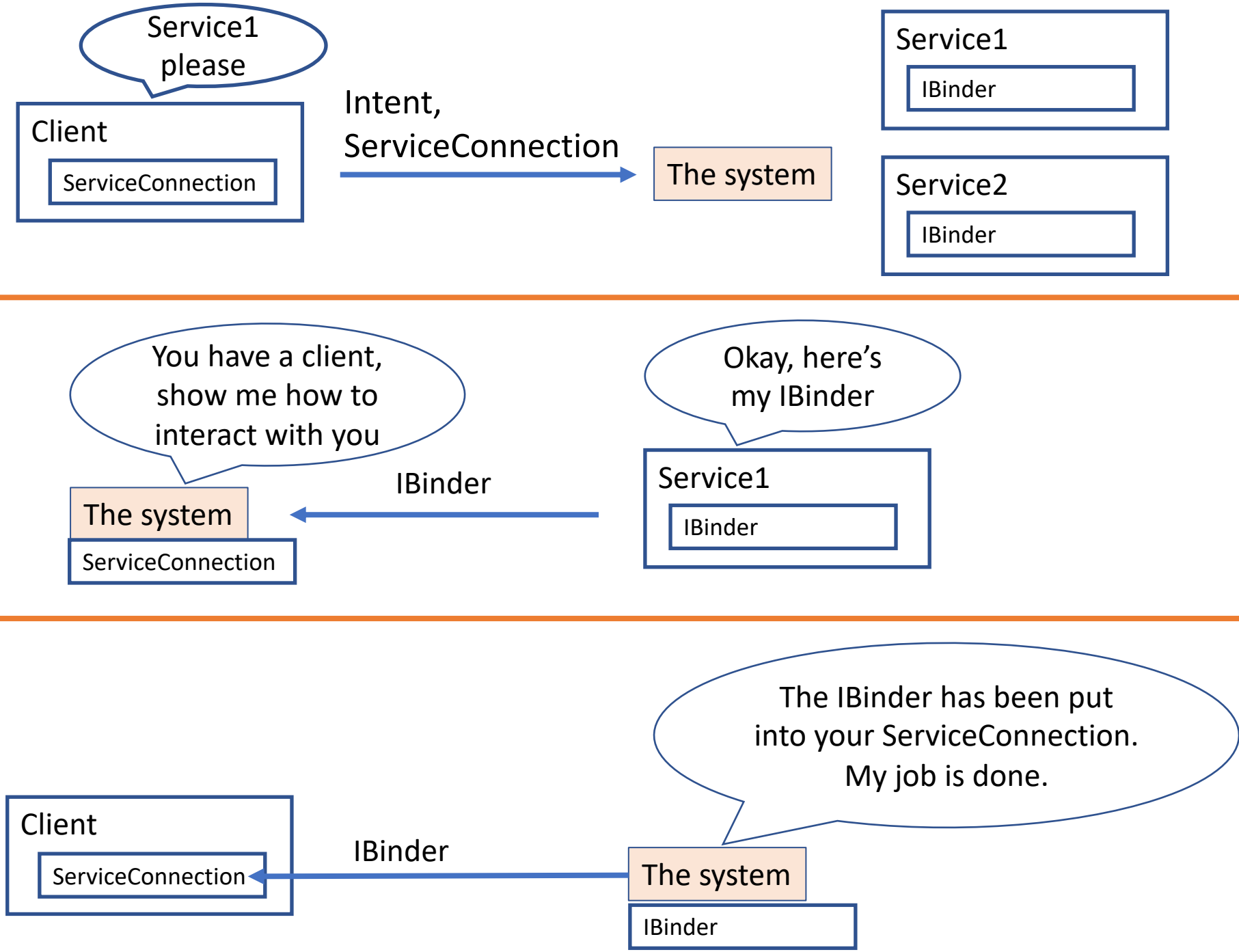
There are four key components involved with `bindService()`:

1. The **service** itself: We showed how to create a `service` class earlier

2. The **client**, another component (e.g. activity) who binds to a service.

3. A **Binder** object, defines the <u>programming interface</u> that clients can use to interact with the service.

4. An implementation of **ServiceConnection** for client, which is the interface for monitoring the state of an application service


Service1 please
Client
ServiceConnection
Intent, ServiceConnection
The system
Service1
IBinder
Service2
IBinder
You have a client, show me how to interact with you
Okay, here's my IBinder
IBinder
The system
ServiceConnection
Service1
IBinder
The IBinder has been put into your ServiceConnection. My job is done.
Client
ServiceConnection
IBinder
The system
IBinder

Step 1: Create a Binder

- Create an instance of `Binder` that either:
  - Contains public methods that the client can call.
  - Returns the current Service instance, which has public methods the client can call.
  - Or, returns an instance of another class hosted by the service with public methods the client can call.
- Return this `Binder` from the `onBind()` method.
- Do not implement IBinder directly.
  https://developer.android.com/reference/android/os/IBinder.html

Step 1: Create a Binder

```
public class MyService extends Service {
   class MyBinder extends Binder {
      public MyService getService() {
         return MyService.this;
      }
   }
   private IBinder binder = new MyBinder();
   MediaPlayer player;

   @Override
   public IBinder onBind(Intent intent) {
      player = MediaPlayer.create(this, R.raw.sample_song);
      return binder;
   }
……
```

res/raw can store audio files

Step 2: the Rest Part of Service

```
……
    public void play() {
        if (!isPlaying())
            player.start();
    }

    public boolean isPlaying() {
        if (player != null) {
            return player.isPlaying();
        } else {
            return false;
        }
    }
}
```

Step 3: Create The Client

```
public class ServiceTest extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ......
        Button playBtn = findViewById(R.id.playBtn);
        playBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(ServiceTest.this,
                                           MyService.class);
                bindService(intent,
                            mConnection, BIND_AUTO_CREATE);
            }
        });
    }
```

This `ServiceConnection` object will be sent to the system, the system will automatically call the methods inside when the service is connected.

"automatically create the service"

Step 4: ServiceConnection

```
bindService(intent, mConnection,
            BIND_AUTO_CREATE);
```

```
……
MyService myService;
boolean myServiceConnected;
ServiceConnection mConnection = new ServiceConnection() {
    @Override
    public void onServiceConnected
         (ComponentName componentName, IBinder iBinder) {
        MyService.MyBinder myBinder = (MyService.MyBinder) iBinder;
        myService = myBinder.getService();
        myServiceConnected = true;
        myService.play();
    }

    @Override
    public void onServiceDisconnected(ComponentName componentName) {
        myService = null;
        myServiceConnected = false;
    }
};
```

Still inside the client class

Step 5: Finishing up

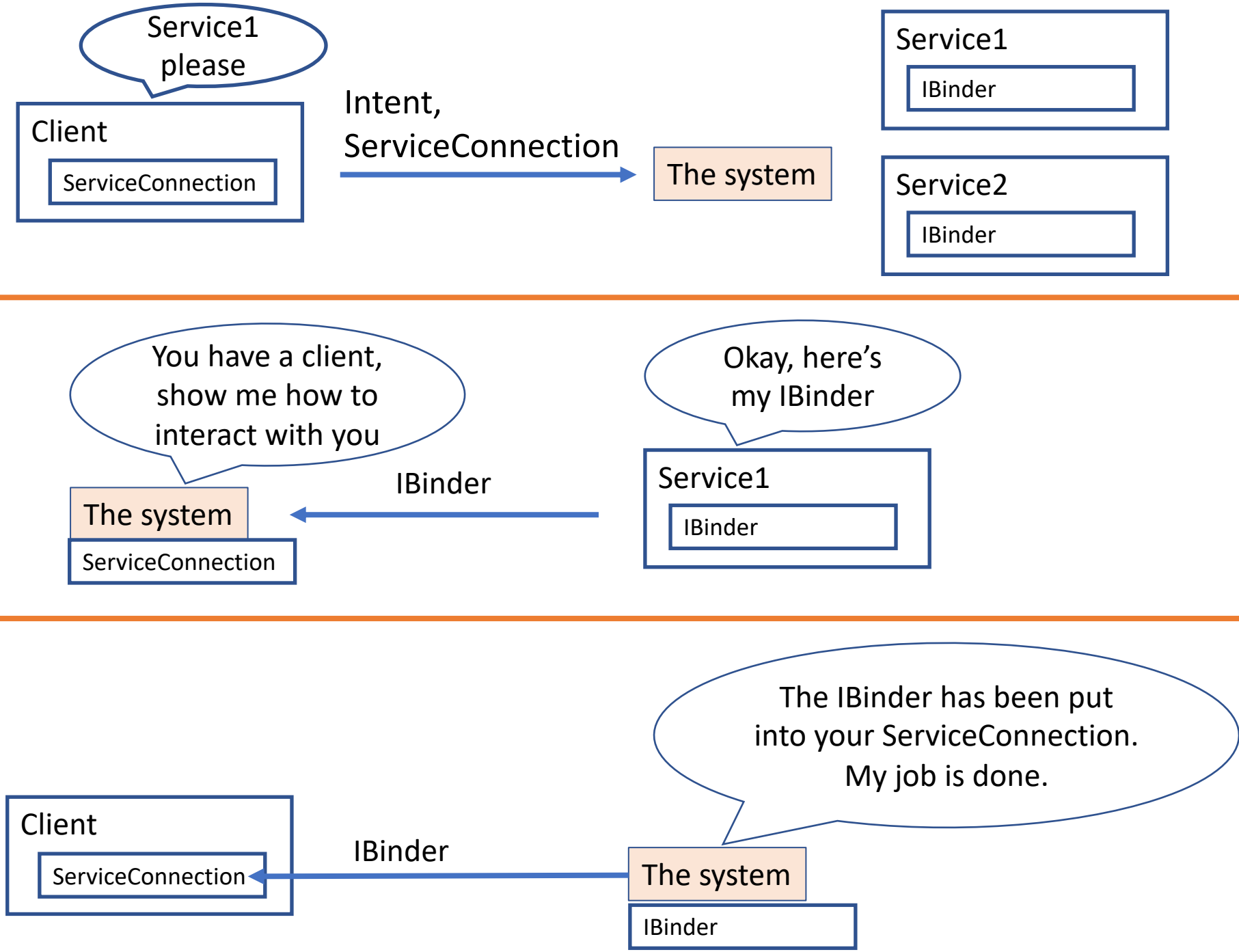
```
……
    @Override
   protected void onStop() {
      super.onStop();
      if (myServiceConnected) {
         unbindService(mConnection);
         myServiceConnected = false;
      }
   }
} // end of the client class
```

The Whole Process

- The client creates the intent to inform the system of its target service.
- The client prepares a `ServiceConnection` object.
- The client calls `bindService()`.
  - A ServiceConnection object is passed to the system so that later, the system will call `onServiceConnected()` when the service is connected.

The Whole Process

- The system receives its first bind request. It calls `onCreate()` method of the service class.
- The system then calls your service's `onBind()` method to retrieve the `IBinder`.
  - This only happens when the first client binds.
  - The system then delivers the same IBinder to any additional clients that bind, without calling onBind() again.
- Then, it calls `onServiceConnected()` with the Ibinder object it just received from `onBind()`.


Service1 please
Client
ServiceConnection
Intent, ServiceConnection
The system
Service1
IBinder
Service2
IBinder
You have a client, show me how to interact with you
Okay, here's my IBinder
IBinder
The system
ServiceConnection
Service1
IBinder
The IBinder has been put into your ServiceConnection. My job is done.
Client
ServiceConnection
IBinder
The system
IBinder

Lab Tasks

- Follow the official documentation and slides, re-create the music player. With proper play, pause and stop buttons

- An additional discussion about thread versus service is available: https://stackoverflow.com/questions/22933762/service-vs-thread-in-android

- Read about foreground service, as they are involved with notifications.