## [Сервисы, приемники сообщений](https://docs.google.com/presentation/d/1wLCJpd5S9O9fHRtSsKd539TpYLFW7YY--gkhqBm2wi8/edit?usp=sharing)

### Цель работы

Познакомится с сервисами Android Studio и научиться создавать приложения принимающие и отправляющие сообщения.

### Задания для выполнения

1. Создайте кнопку, которая будет выводить всплывающее сообщение на экран при нажатии с помощью объекта **Toast.**
2. Создайте проигрыватель PlayService.java для воспроизведения и остановки одной песни.
3. Создайте приложение, которое будет отправлять и принимать сообщения.

### Методические указания

*Создание Toast-сообщения*

Всплывающее уведомление (Toast Notification) является сообщением, которое появляется на поверхности окна приложения, заполняя необходимое ему количество пространства, требуемого для сообщения. При этом текущая деятельность приложения остаётся работоспособной для пользователя. В течение нескольких секунд сообщение плавно закрывается. Всплывающее уведомление также может быть создано службой, работающей в фоновом режиме. Как правило, всплывающее уведомление используется для показа коротких текстовых сообщений.

Для создания всплывающего уведомления необходимо инициализировать объект Toast при помощи метода Toast.makeText(), а затем вызвать метод show() для отображения сообщения на экране:

Toast toast = Toast.makeText(getApplicationContext(),

 "Текстовое сообщение", Toast.LENGTH\_SHORT);

toast.show();

У метода makeText() есть три параметра:

* Контекст приложения
* Текстовое сообщение
* Продолжительность времени показа уведомления. Можно использовать только две константы

Уведомления выводятся на 3 с половиной секунды или на 2 секунды.



По умолчанию стандартное всплывающее уведомление появляется в нижней части экрана. Изменить место появления уведомления можно с помощью метода setGravity(int, int, int). Метод принимает три параметра:

* стандартная константа для размещения объекта в пределах большего контейнера (например, GRAVITY.CENTER, GRAVITY.TOP и др.);
* смещение по оси X
* смещение по оси Y

Например, если вы хотите, чтобы уведомление появилось в центре экрана, то используйте следующий код (до вызова метода show()):

toast.setGravity(Gravity.CENTER, 0, 0);

Для вывода в левом верхнем углу.

toast.setGravity(Gravity.TOP or Gravity.LEFT, 0, 0)

Если нужно сместить уведомление направо, то просто увеличьте значение второго параметра. Для смещения вниз нужно увеличить значение последнего параметра. Соответственно, для смещения вверх и влево используйте отрицательные значения.

## PlayService.java

### На примере создания проигрывателя в Android Studio разберемся как работают службы (сервисы).

### Службы (Сервисы) в Android работают как фоновые процессы и представлены классом android.app.Service. Они не имеют пользовательского интерфейса и нужны в тех случаях, когда не требуется вмешательства пользователя. Сервисы работают в фоновом режиме, выполняя сетевые запросы к веб-серверу, обрабатывая информацию, запуская уведомления и т.д. Служба может быть запущена и будет продолжать работать до тех пор, пока кто-нибудь не остановит её или пока она не остановит себя сама.

Сервисы предназначены для длительного существования, в отличие от активностей. Они могут работать, постоянно перезапускаясь, выполняя постоянные задачи или выполняя задачи, требующие много времени.

Android даёт службам более высокий приоритет, чем бездействующим активностям, поэтому вероятность того, что они будут завершены из-за нехватки ресурсов, заметно уменьшается. По сути, если система должна преждевременно завершить работу запущенного сервиса, он может быть настроен таким образом, чтобы запускаться повторно, как только станет доступно достаточное количество ресурсов. В крайних случаях прекращение работы сервиса (например, задержка при проигрывании музыки) будет заметно влиять на впечатления пользователя от приложения, и в подобных ситуациях приоритет сервиса может быть повышен до уровня активности, работающей на переднем плане.

Чтобы определить службу, необходимо создать новый класс, расширяющий базовый класс Service. Можно воспользоваться готовым мастером создания класса для сервиса в Android Studio. Щёлкаем правой кнопкой мыши на папке java (или на имени пакета) и выбираем New | Service | Service:



В следующем окне выбираем имя сервиса (флажки оставляем) и нажимаем кнопку Finish.



Получим заготовку.

package ru.alexanderklimov.service;

import android.app.Service;

import android.content.Intent;

import android.os.IBinder;

public class MyService extends Service {

 public MyService() {

 }

 @Override

 public IBinder onBind(Intent intent) {

 // TODO: Return the communication channel to the service.

 throw new UnsupportedOperationException("Not yet implemented");

 }

}

При этом сервис автоматически зарегистрируется в манифесте в секции <application>.

<service

 android:name=".MyService"

 android:enabled="true"

 android:exported="true" >

</service>

Если бы мы убрали флажки на экране мастера, то оба атрибута имели бы значение false. Например, атрибут exported даёт возможность другим приложениям получить доступ к вашему сервису.

Имеются и другие атрибуты, например, permission, чтобы сервис запускался только вашим приложением.

Также вы можете обойтись без мастера и создать вручную класс сервиса и запись в манифесте, теперь вы знаете, из чего он состоит.

Подобно активностям служба имеет свои методы жизненного цикла:

* onCreate()
* onStartCommand()
* onDestroy()



Для быстрого создания заготовок нужных методов используйте команду меню Code | Override Methods... или набирайте сразу имя метода, используя автодополнение.

Чтобы создать службу PlayService можно воспользоваться мастером или создать класс сервиса вручную. Опишите его функционал:

package com.example.myapplication;

import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.IBinder;

import android.widget.Toast;

public class PlayService extends Service {

 private MediaPlayer mPlayer;

 @Override
 public IBinder onBind(Intent intent) {
 return null;
 }

 @Override
 public void onCreate() {
 super.onCreate();
 Toast.*makeText*(this, "Служба создана",
 Toast.*LENGTH\_SHORT*).show();
 mPlayer = MediaPlayer.*create*(this, R.raw.*flower\_romashka*);
 mPlayer.setLooping(false);
 }

 @Override
 public int onStartCommand(Intent intent, int flags, int startId) {
 Toast.*makeText*(this, "Служба запущена",
 Toast.*LENGTH\_SHORT*).show();
 mPlayer.start();
 return super.onStartCommand(intent, flags, startId);
 }

 @Override
 public void onDestroy() {
 super.onDestroy();
 Toast.*makeText*(this, "Служба остановлена",
 Toast.*LENGTH\_SHORT*).show();
 mPlayer.stop();
 }
}

Зарегистрируем службу в файле манифеста.

<service

 android:enabled="true"

 android:name=".PlayService">

</service>

В файле разметки для активности определим две кнопки: Старт и Стоп:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

 android:layout\_width="match\_parent"

 android:layout\_height="match\_parent"

 android:gravity="center"

 android:orientation="vertical">

 <Button

 android:id="@+id/button\_start"

 android:layout\_width="match\_parent"

 android:layout\_height="wrap\_content"

 android:text="Старт" />

 <Button

 android:id="@+id/button\_stop"

 android:layout\_width="match\_parent"

 android:layout\_height="wrap\_content"

 android:text="Стоп" />

</LinearLayout>

В классе активности в обработчиках событий кнопок будем вызывать методы startService() и stopService() для управления службой.

package com.example.myapplication;

import static android.app.Service.START\_FLAG\_RETRY;

import androidx.appcompat.app.AppCompatActivity;

import android.app.Service;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity\_main);

 final Button btnStart = findViewById(R.id.button\_start);
 final Button btnStop = findViewById(R.id.button\_stop);

 // запуск службы
 btnStart.setOnClickListener(new View.OnClickListener() {
 @Override
 public void onClick(View view) {
 // используем явный вызов службы
 startService(
 new Intent(MainActivity.this, PlayService.class));
 }
 });

 // остановка службы
 btnStop.setOnClickListener(new View.OnClickListener() {
 @Override
 public void onClick(View view) {
 stopService(
 new Intent(MainActivity.this, PlayService.class));
 }
 });
 }
}

Чтобы воспроизводить аудио, MediaPlayer должен знать, какой именно ресурс (файл) нужно производить. Установить нужный ресурс для воспроизведения можно тремя способами:

* в метод create() объекта MediaPlayer передается id ресурса, представляющего аудиофайл
* в метод create() объекта MediaPlayer передается объект Uri, представляющего аудиофайл
* в метод setDataSource() объекта MediaPlayer передается полный путь к аудиофайлу

После установки ресурса вызывается метод prepare() или prepareAsync() (асинхронный вариант prepare()). Этот метод подготавливает аудиофайл к воспроизведению, извлекая из него первые секунды. Если мы воспроизводим файл из сети, то лучше использовать prepareAsync().

Для управления воспроизведением в классе MediaPlayer определены следующие методы:

* start(): запускает аудио
* pause(): приостанавливает воспроизведение
* stop(): полностью останавливает воспроизведение

Итак, создадим новый проект. Аудиофайл должен находиться в папке res/raw, поэтому добавим в проект в Android Studio такую папку. Для этого нажмем на папку res правой кнопкой мыши и в появившемся меню выберем New -> Android Resource Directory:



Затем в появившемся окне в качестве типа папки укажем raw (что также будет использоваться в качестве названия папки):



И скопируем в нее какой-нибудь аудио-файл, предварительно открыв папку в файловом менеджере.





Запустите приложение, прослушайте музыку. Обратите внимание, что когда приложение сворачивается, музыка продолжает играть.


## Передача сообщений

Создайте новый проект и разместите на экране кнопку с надписью "Отправить сообщение". Присвойте атрибуту onClick название метода, в котором будет происходит отправка широковещательного сообщения.

<Button

 android:layout\_width="wrap\_content"

 android:layout\_height="wrap\_content"

 android:text="Отправить сообщение"

 android:id="@+id/button"

 android:layout\_gravity="center\_horizontal"

 android:onClick="sendMessage" />

В классе активности создаём уникальную строку и реализуем метод для щелчка кнопки. Также добавим дополнительные данные - первую часть послания радистки.

public static final String WHERE\_MY\_CAT\_ACTION = "ru.alexanderklimov.action.CAT";

public static final String ALARM\_MESSAGE = "Срочно пришлите кота!";

public void sendMessage(View view) {

 Intent intent = new Intent();

 intent.setAction(WHERE\_MY\_CAT\_ACTION);

 intent.putExtra("ru.alexanderklimov.broadcast.Message", ALARM\_MESSAGE);

 intent.addFlags(Intent.FLAG\_INCLUDE\_STOPPED\_PACKAGES);

 sendBroadcast(intent);

}

Запустив пример, вы можете нажать на кнопку и отправлять сообщение. Только ваше сообщение уйдёт в никуда, так как ни одно приложение не оборудовано приёмником для него. Исправим ситуацию и создадим приёмник в своём приложении. Мы будем сами принимать свои же сообщения.

Приёмник представляет собой обычный Java-класс на основе BroadcastReceiver. Вы можете создать вручную класс и наполнить его необходимыми методами. Раньше так и поступали. Но в студии есть готовый шаблон, который поможет немного сэкономить время.

Щёлкаем правой кнопкой мыши на названии пакета и выбираем New | Other | Broadcast Receiver



В диалоговом окне задаём имя приёмника, остальные настройки оставляем без изменений.



Студия создаст изменения в двух местах. Во-первых, будет создан класс MessageReceiver:

package ru.alexanderklimov.testapplication;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

public class MessageReceiver extends BroadcastReceiver {

 public MessageReceiver() {

 }

 @Override

 public void onReceive(Context context, Intent intent) {

 // TODO: This method is called when the BroadcastReceiver is receiving

 // an Intent broadcast.

 throw new UnsupportedOperationException("Not yet implemented");

 }

}

Во-вторых, в манифесте будет добавлен новый блок.

<receiver

 android:name=".MessageReceiver"

 android:enabled="true"

 android:exported="true" >

</receiver>

В него следует добавить фильтр, по которому он будет ловить сообщения.

<receiver

 android:name=".MessageReceiver"

 android:enabled="true"

 android:exported="true">

 <intent-filter>

 <action android:name="ru.alexanderklimov.action.CAT" />

 </intent-filter>

</receiver>

Вернёмся в класс приёмника и модифицируем метод onReceive().

@Override

public void onReceive(Context context, Intent intent) {

 Toast.makeText(context, "Обнаружено сообщение: " +

 intent.getStringExtra("ru.alexanderklimov.broadcast.Message"),

 Toast.LENGTH\_LONG).show();

}

Снова запустим пример и ещё раз отправим сообщение. Так как наше приложение теперь оборудовано не только передатчиком, но и приёмником, то оно должно уловить сообщение и показать его нам.

Если приложение работает не корректно, необходимо реализовать следующие изменения.

Когда приложение запускается в фоновом режиме, оно потребляет некоторые ограниченные ресурсы устройства, такие как оперативная память. Это может привести к ухудшению работы пользователя, особенно если пользователь использует ресурсоемкое приложение, такое как игра или просмотр видео. Чтобы улучшить пользовательский интерфейс, Android 8.0 (уровень API 26) накладывает ограничения на то, что приложения могут делать во время работы в фоновом режиме. В этом скриншоте приводятся изменения в коде приложения, чтобы оно хорошо работало в соответствии с новыми ограничениями.



### Контрольные вопросы

1. Чем сервисы отличаются от Activity?
2. Возможно ли в Toast Notification разместить LinearLayout?
3. Сколько времени отображается всплывшее сообщение созданное с помощью объекта **Toast?**

### Дополнительные задания

1. Создайте кнопку, которая будет выводить всплывающую панель с картинкой и текстом на экран при нажатии с помощью объекта **Toast.**
2. Создайте проигрыватель PlayService.java для воспроизведения списка песен из плей-листа.

**Материалы**

<http://developer.alexanderklimov.ru/android/notification.php>

<http://developer.alexanderklimov.ru/android/toast.php>

<http://developer.alexanderklimov.ru/android/theory/services-theory.php>

<https://metanit.com/java/android/11.2.php>

<http://developer.alexanderklimov.ru/android/broadcast.php>

<https://developer.android.com/about/versions/oreo/background.html>