Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский политехнический университет

Петра Великого»

Институт компьютерных наук и технологий

# Отчёт по лабораторной работе №7 Дисциплина: Проектирование мобильных приложений

Выполнил студент гр. 3530901/70202 \_\_\_\_\_\_\_\_\_\_\_ Павлов Д.В.

*(подпись)*

Преподаватель \_\_\_\_\_\_\_\_\_\_\_ Кузнецов А.Н.

*(подпись)*

Санкт-Петербург

2019

**1. Цели работы**

* Получить практические навыки разработки сервисов (started и bound) и Broadcast Receivers.

**2. Выполнение работы**

# 2.1. Задание 1. Started сервис для скачивания изображения

Сервисы в Android не имеют пользовательского интерфейса и работают в фоновом режиме. Используются сервисы в тех случаях, когда не требуется вмешательства пользователя, например, для выполнения сетевых запросов к веб-серверу, обработки информации, запуска уведомлений и т.д. Сервис может продолжать работать до тех пор, пока какой-либо компонент не остановит его, либо пока он сам себя не остановит по каким-либо условиям. Таким образом, сервисы в отличие от активностей предназначены для длительного функционирования.

Сервисы является компонентом приложения. Запускается сервис другим компонентом приложения, который может привязаться к сервису для взаимодействия с ним. Сервис может принимать одну из двух форм : *started and bound.*

В данном пункте требуется реализовать Started service. Сервис является «запущенным», когда он стартует вызовом метода активности *startService()*. После останова/уничтожения вызывающего *Service* компонента приложения сервис будет функционировать в фоновом режиме в течение неограниченного времени. Обычно запущенный сервис не возвращает результатов вызывающему компоненту, и должен остановиться самостоятельно после завершения операции. Например, сервис может загружать или выгружать файл по сети.

Код приложения разделен на 2 части – активити и сервис. Он представлен в Листингах 2.1.1-2 соответственно.

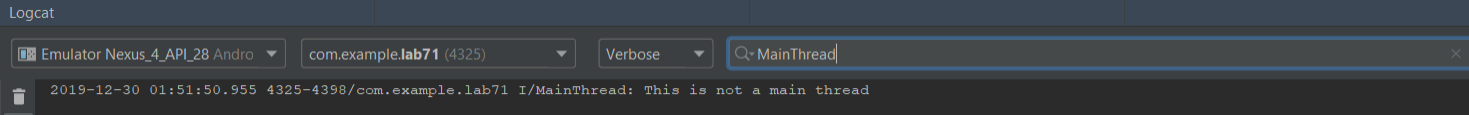
Листинг 2.1.1. MainActivity

const val *IMAGE\_KEY* = "image\_downloaded"  
const val *IS\_LOADED\_KEY* = "is\_downloaded"  
const val *URL* = "https://im0-tub-ru.yandex.net/i?id=389e4c5fcd7e6a3fd8b022fad23329a4&n=13"  
  
class MainActivity : AppCompatActivity() {  
 private lateinit var broadcastReceiver: BroadcastReceiver  
  
 companion object {  
 private lateinit var imgToDownload: ImageView  
 private lateinit var dIcon: Bitmap  
 }  
  
 private var imageIsLoaded = false  
  
 override fun onSaveInstanceState(outState: Bundle) {  
 super.onSaveInstanceState(outState)  
 if (imageIsLoaded) {  
 outState.putParcelable(*IMAGE\_KEY*, dIcon)  
 outState.putBoolean(*IS\_LOADED\_KEY*, imageIsLoaded)  
 }  
 }  
  
 override fun onRestoreInstanceState(savedInstanceState: Bundle) {  
 super.onRestoreInstanceState(savedInstanceState)  
 if (savedInstanceState.getBoolean(*IS\_LOADED\_KEY*)) {  
 imageIsLoaded = true  
 findViewById<ImageView>(R.id.*downloadedImage*).setImageBitmap(  
 savedInstanceState.getParcelable(  
 *IMAGE\_KEY* )  
 )  
 }  
 }  
  
 override fun onResume() {  
 super.onResume()  
 this.registerReceiver(broadcastReceiver, IntentFilter(DownloadService.notification))  
 }  
  
 override fun onPause() {  
 super.onPause()  
 this.unregisterReceiver(broadcastReceiver)  
 }  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
 imgToDownload = findViewById(R.id.*downloadedImage*)  
 val butToDownload = findViewById<Button>(R.id.*downloadButton*)  
 butToDownload.setOnClickListener **{** if (!imageIsLoaded) {  
 val downloadIntent = Intent(this, DownloadService::class.*java*)  
 downloadIntent.putExtra(  
 "url", *URL* )  
 startService(downloadIntent)  
 }  
 **}** broadcastReceiver = object : BroadcastReceiver() {  
 override fun onReceive(context: Context?, intent: Intent) {  
 try {  
 dIcon = BitmapFactory.decodeByteArray(  
 intent.getByteArrayExtra(DownloadService.publish), 0,  
 intent.getByteArrayExtra(DownloadService.publish)!!.size  
 )  
 imgToDownload.setImageBitmap(dIcon)  
 imageIsLoaded = true  
 } catch (e: NullPointerException) {  
 Log.e("Error", "" + e.message)  
 e.printStackTrace()  
 }  
 }  
 }  
 }  
}

Листинг 2.1.2. DownloadService

class DownloadService : IntentService("DownloadService") {  
  
 companion object {  
 const val notification = "com.example.lab71"  
 const val publish = "publish"  
 }  
  
 override fun onHandleIntent(intent: Intent?) {  
 val urlPath = intent?.getStringExtra("url")  
 val inputURL = URL(urlPath).openStream()  
 val broadcastedImg = BitmapFactory.decodeStream(inputURL)  
 if (Looper.getMainLooper().*thread* != Thread.currentThread()) {  
 Log.i("MainThread", "This is not a main thread")  
 } else {  
 Log.i("MainThread", "This is a main thread")  
 }  
 val intentToSend = Intent(notification)  
 val byteStream = ByteArrayOutputStream()  
 broadcastedImg?.compress(Bitmap.CompressFormat.PNG, 50, byteStream)  
 intentToSend.putExtra(publish, byteStream.toByteArray())  
 sendBroadcast(intentToSend)  
 }  
}

Ниже приведено докозательство, что код исполняется не в UI потоке.



Загруженное изображение представлено на Рис.2.1.1.:

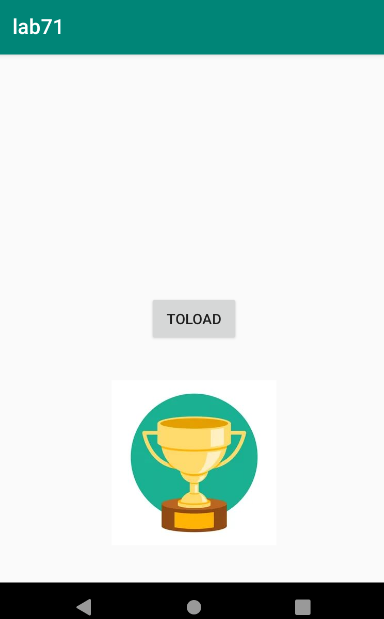


Рис.2.1.1.

Проверка правильности работы приложения при сворачивании, разворачивании и изменении ориентации экрана прошла успешно.

# 2.2. Задание 2. Broadcast Receiver

Код приложений представлен в Листингах 2.2.1-4 соответственно.

Листинг 2.2.1. MainActivity (service)

const val *URL* = "https://im0-tub-ru.yandex.net/i?id=389e4c5fcd7e6a3fd8b022fad23329a4&n=13"  
const val *SERVICE\_NAME* = "com.example.lab72.DownloadService"  
  
class MainActivity : AppCompatActivity() {  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
 val butToDownload = findViewById<Button>(R.id.*downloadButton*)  
 butToDownload.setOnClickListener **{** val downloadIntent = Intent(this, DownloadService::class.*java*)  
 downloadIntent.putExtra("url", *URL*)  
 startService(downloadIntent)  
 **}** }  
}

Листинг 2.2.2. DownloadService (service)

const val *FILE\_NAME* = "cup.jpg"  
const val *PUBLISH\_KEY* = "publish"  
  
class DownloadService : IntentService("DownloadService") {  
  
 override fun onHandleIntent(intent: Intent?) {  
 val url = intent?.getStringExtra("url")  
 val inputURL = URL(url).openStream()  
 val broadcastedImg = BitmapFactory.decodeStream(inputURL)  
  
 val file = File(*applicationContext*.*filesDir*, *FILE\_NAME*)  
 try {  
 val outputStream: OutputStream = FileOutputStream(file)  
 broadcastedImg?.compress(Bitmap.CompressFormat.PNG, 50, FileOutputStream(file))  
 outputStream.flush()  
 outputStream.close()  
 } catch (e: IOException) {  
 Log.e("Error", "" + e.message)  
 e.printStackTrace()  
 }  
  
 val intentToSend = Intent(*SERVICE\_NAME*)  
 intentToSend.putExtra(*PUBLISH\_KEY*, Uri.parse(file.*absolutePath*).toString())  
 sendBroadcast(intentToSend)  
 }  
}

Листинг 2.2.3. MainActivity (client)

const val *SERVICE\_NAME* = "com.example.lab72.DownloadService"  
const val *PUBLISH\_KEY* = "publish"  
  
class MainActivity : AppCompatActivity() {  
  
 private lateinit var receiver: MyBroadcastReceiver  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
 receiver = MyBroadcastReceiver()  
 val filter = IntentFilter(*SERVICE\_NAME*)  
 registerReceiver(receiver, filter)  
 }  
}

Листинг 2.2.4. MyBroadcastReceiver (client)

class MyBroadcastReceiver : BroadcastReceiver() {  
  
 override fun onReceive(context: Context, intent: Intent) {  
 try {  
 (context as MainActivity)  
 .path.*text* =  
 intent.getStringExtra(*PUBLISH\_KEY*)  
 } catch (e: NullPointerException) {  
 Log.e("Error", "" + e.message)  
 e.printStackTrace()  
 }  
 }  
}

На Рис.2.2.1-2 показано 2 приложения. Текст (путь к картинке) в приложении “client” появился после нажатия на кнопку toload, что означает корректность работы сервиса.

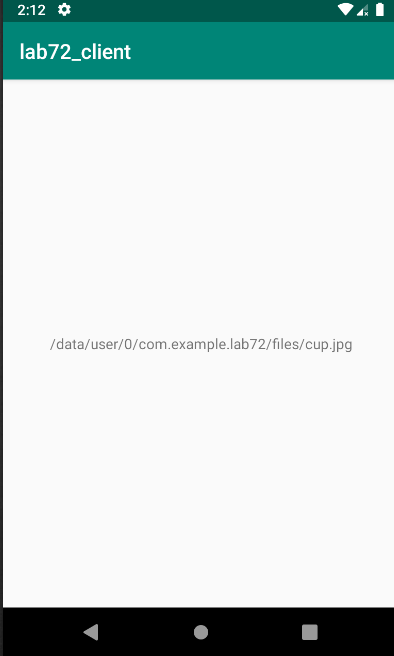
 

Рис.2.2.1. Рис.2.2.2.

# 2.3. Задание 3. Bound Service для скачивания изображения

# Сервис является Bound (привязанным), когда он стартуется вызовом метода активности *bindService()*. Привязанный сервис функционирует в режиме «клиент-сервер», позволяющий компонентам взаимодействовать с сервисом для отправки запросов и получения результатов. Привязанный сервис функционирует только пока к нему привязан другой компонент приложения. К сервису могут быть привязаны несколько компонентов одновременно, но, когда все они «отменяют привязку», сервис уничтожается.

В качестве данного пункта будет представлена часть моего курсового проекта, а именно музыкальный сервис. Данный сервис отвечает за воспроизведение музыки в каждом из Activity. В зависимости от булевой переменной soundIsOff мы будем либо привязывать сервис к Activity (doBindService()), либо отвязывать (doUnbindService()), если изначально был привязан. Также здесь прописаны методы pauseMusic() & resumeMusic(), чтобы при сворачивании приложения музыка уходила на паузу (а не начиналась с начала после возвращения в приложение).

Код музыкального сервиса представлен в Листинге 2.3.1.

Листинг 2.3.1. MusicService

public class MusicService extends Service implements MediaPlayer.OnErrorListener {  
  
 private final IBinder mBinder = new ServiceBinder();  
 MediaPlayer mPlayer;  
 private int length = 0;  
 HomeActivity homeActivity;  
  
 public MusicService() {  
 }  
  
 public class ServiceBinder extends Binder {  
 MusicService getService() {  
 return MusicService.this;  
 }  
 }  
  
 @Override  
 public IBinder onBind(Intent arg0) {  
 return mBinder;  
 }  
  
 @Override  
 public void onCreate() {  
 super.onCreate();  
  
 mPlayer = MediaPlayer.*create*(this, R.raw.*track*);  
 mPlayer.setOnErrorListener(this);  
  
 if (mPlayer != null) {  
 mPlayer.setLooping(true);  
 mPlayer.setVolume(50, 50);  
 }  
  
  
 mPlayer.setOnErrorListener(new MediaPlayer.OnErrorListener() {  
  
 public boolean onError(MediaPlayer mp, int what, int  
 extra) {  
 onError(mPlayer, what, extra);  
 return true;  
 }  
 });  
 }  
  
 @Override  
 public int onStartCommand(Intent intent, int flags, int startId) {  
 if (!homeActivity.*soundIsOff*)  
 if (mPlayer != null) {  
 mPlayer.start();  
 }  
 return *START\_NOT\_STICKY*;  
 }  
  
 public void pauseMusic() {  
 if (mPlayer != null) {  
 if (mPlayer.isPlaying()) {  
 mPlayer.pause();  
 length = mPlayer.getCurrentPosition();  
 }  
 }  
 }  
  
 public void resumeMusic() {  
 if (mPlayer != null) {  
 if (!mPlayer.isPlaying()) {  
 mPlayer.seekTo(length);  
 mPlayer.start();  
 }  
 }  
 }  
  
 @Override  
 public void onDestroy() {  
 super.onDestroy();  
 if (mPlayer != null) {  
 try {  
 mPlayer.stop();  
 mPlayer.release();  
 } finally {  
 mPlayer = null;  
 }  
 }  
 }  
  
 public boolean onError(MediaPlayer mp, int what, int extra) {  
 Toast.*makeText*(this, "Проигрывание музыки завершилось ошибкой",  
 Toast.*LENGTH\_SHORT*).show();  
 if (mPlayer != null) {  
 try {  
 mPlayer.stop();  
 mPlayer.release();  
 } finally {  
 mPlayer = null;  
 }  
 }  
 return false;  
 }  
}

Методы привязки и отвязки сервиса представлены в Листинге 2.3.2.

Листинг 2.3.2.

void doBindService() {  
 bindService(new Intent(this, MusicService.class),  
 Scon, Context.*BIND\_AUTO\_CREATE*);  
 mIsBound = true;  
}  
  
void doUnbindService() {  
 if (mIsBound) {  
 unbindService(Scon);  
 mIsBound = false;  
 }  
}

# Запуск сервиса и его остановка представлены в Листинге 2.3.3.

Листинг 2.3.3.

@Override  
public boolean onOptionsItemSelected(MenuItem optionItem) {  
 int id = optionItem.getItemId();  
 if (id == R.id.*sound*) {  
 if (!*soundIsOff*) {  
 doUnbindService();  
 Intent music = new Intent();  
 music.setClass(this, MusicService.class);  
 stopService(music);  
 optionItem.setIcon(R.drawable.*volume\_down*);  
 *soundIsOff* = true;  
 } else {  
 doBindService();  
 Intent music = new Intent();  
 music.setClass(this, MusicService.class);  
 startService(music);  
 optionItem.setIcon(R.drawable.*volume\_up*);  
 *soundIsOff* = false;  
 }  
 }  
 return super.onOptionsItemSelected(optionItem);  
}

# 3. Выводы

В ходе выполнения данной лабораторной работы была продолжена работа с многопоточными приложениями. Были рассмотрены 2 вида сервисов – просто started и started & bound.

Сервисы Android — это мощный механизм, позволяющий приложениям выполнять задачи в фоновом режиме. После запуска сервис будет продолжать работать независимо от того, является ли вызывающее приложение приоритетной задачей или нет, и даже в том случае, если компонент, инициировавший службу, будет уничтожен.

Started сервисы работают до тех пор, пока они не будут остановлены или уничтожены и по своей природе не обеспечивают механизм взаимодействия или обмена данными с другими компонентами. Bound сервисы, с другой стороны, предоставляют интерфейс связи с другими компонентами клиента и обычно работают до тех пор, пока последний клиент не отсоединится от службы.

По умолчанию службы работают локально в том же процессе и главном потоке, что и вызывающее приложение. Поэтому в сервисе должен быть создан новый поток для обработки ресурсоемких задач.