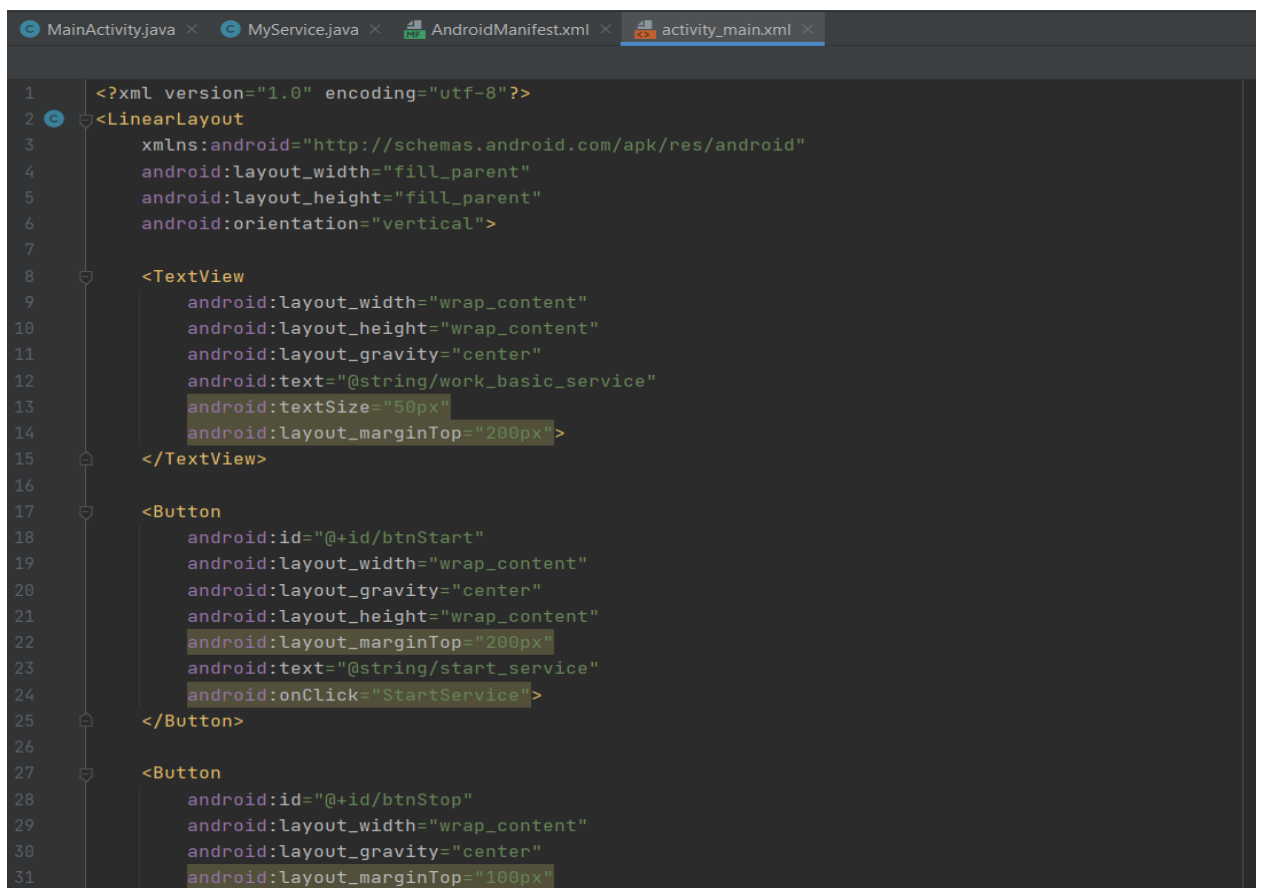


Отчёт по проделанной работе на семинаре по мобильной разработке (16.02.2022).

(выполнил Валяев Георгий Анатольевич)

В наше время в работе большинства многофункциональных мобильных приложений не менее важную и большую (местами рутинную) роль занимают сервисы (небольшая программа как компонент приложения, которая выполняет конкретный пул долгих задач и операций в фоновом режиме без отображения на интерфейсах пользователю). Поэтому в процессе разработки создание сервиса значительно облегчает жизнь как приложению, так и самому разработчику при выполнении задач.

Перейдём к поэтапному созданию простого сервиса, который будет работать в фоновом режиме телефона и выдавать сообщения о работе стадий жизненного цикла в логи. Начинаем разработку с визуальной составляющей приложения – файла `activity_main.xml`:



```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      android:orientation="vertical">
7
8      <TextView
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:layout_gravity="center"
12         android:text="@string/work_basic_service"
13         android:textSize="50px"
14         android:layout_marginTop="200px">
15     </TextView>
16
17     <Button
18         android:id="@+id/btnStart"
19         android:layout_width="wrap_content"
20         android:layout_gravity="center"
21         android:layout_height="wrap_content"
22         android:layout_marginTop="200px"
23         android:text="@string/start_service"
24         android:onClick="StartService">
25     </Button>
26
27     <Button
28         android:id="@+id/btnStop"
29         android:layout_width="wrap_content"
30         android:layout_gravity="center"
31         android:layout_marginTop="100px"
```

```
MainActivity.java × MyService.java × AndroidManifest.xml × activity_main.xml ×
20         android:layout_gravity="center"
21         android:layout_height="wrap_content"
22         android:layout_marginTop="200px"
23         android:text="@string/start_service"
24         android:onClick="StartService">
25     </Button>
26
27     <Button
28         android:id="@+id/btnStop"
29         android:layout_width="wrap_content"
30         android:layout_gravity="center"
31         android:layout_marginTop="100px"
32         android:layout_height="wrap_content"
33         android:text="@string/stop_service"
34         android:onClick="StopService">
35     </Button>
36
37     <ProgressBar
38         android:layout_width="wrap_content"
39         android:layout_gravity="center"
40         android:layout_marginTop="200px"
41         android:layout_height="wrap_content"
42         android:indeterminate="true">
43     </ProgressBar>
44
45 </LinearLayout>
```

Здесь мы разместили по центру главной активности 2 кнопки для запуска сервиса и его остановки, а также текстовое окно с кратким описанием самого приложения и визуала работы сервиса при помощи `<ProgressBar/>`.

Сам файл работы сервиса мы будем создавать через обыкновенный Java-класс, но просто дополнительно при создании класса пропишем `extends Service`, что будет означать унаследование всех методов и свойств главного класса сервиса.

Помимо этого, пропишем все основные методы в рамках всего жизненного цикла сервиса (`onCreate()`, `onStartCommand()`, `onBind()`, `onUnbind()`, `onRebind()`, `onDestroy()`), в одном из которых добавим полезное действие — вывод реального времени в мире через уведомление в приложении и генерация процента успешного выполнения всех дел для пользователя относительно этого времени:

```

1 package com.example.service_basic;
2
3 import ...
4
11
12 public class MyService extends Service {
13
14     final String LOG_TAG = "My_LOGs";
15     final IBinder MyBinder = new MyBinder();
16
17     public class MyBinder extends Binder {
18         MyService getService() {
19             return MyService.this;
20         }
21     }
22
23     @Override
24     public void onCreate(){
25         Log.i(LOG_TAG, msg: "Method onCreate() started...");
26         super.onCreate();
27     }
28
29     @Override
30     public int onStartCommand(Intent intent, int flags, int startId){
31         Log.i(LOG_TAG, msg: "Method onStartCommand() started...");
32         interestTask();
33         return super.onStartCommand(intent, flags, startId);
34     }
35
36     @Override
37     public IBinder onBind(Intent intent) {
38         Log.i(LOG_TAG, msg: "Method onBind() started...");
39         return MyBinder;
40     }
41

```

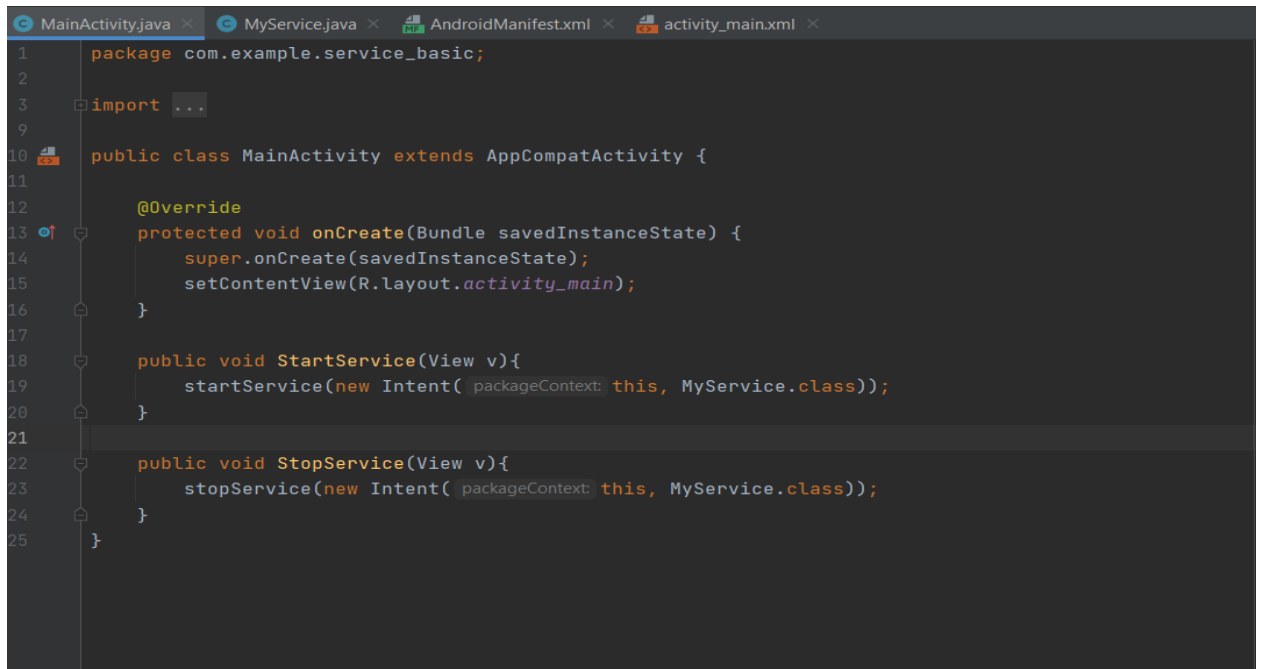
```

40 }
41
42 @Override
43 public boolean onUnbind(Intent intent){
44     Log.i(LOG_TAG, msg: "Method onUnbind() started...");
45     return super.onUnbind(intent);
46 }
47
48 @Override
49 public void onRebind(Intent intent){
50     Log.i(LOG_TAG, msg: "Method onRebind() started...");
51     super.onRebind(intent);
52 }
53
54 @Override
55 public void onDestroy(){
56     Log.i(LOG_TAG, msg: "Method onDestroy() started...");
57     Toast.makeText(getApplicationContext(), text: "Сервис завершил свою работу!",
58         Toast.LENGTH_LONG).show();
59     super.onDestroy();
60 }
61
62 // Интересное действие в рамках полезности сервиса приложения
63 public void interestTask(){
64     Date date = new Date();
65     double value = Math.random() * 100;
66     String result = String.format("%.2f", value);
67     Toast.makeText(getApplicationContext(), text: "Now is " + date.toString(),
68         Toast.LENGTH_LONG).show();
69     Toast.makeText(getApplicationContext(), text: "Шанс сделать все дела = " + (result) + "%",
70         Toast.LENGTH_LONG).show();
71 }
72 }

```

При помощи библиотеки Toast и метода `makeText()` мы формируем всплывающее уведомление приложения при старте работы сервиса и отправляем в параметры сами строки для вывода (реальное время во всём мире + рандомный процент выполнения всех дел на сегодня).

Перейдём теперь к файлу `MainActivity.java`:

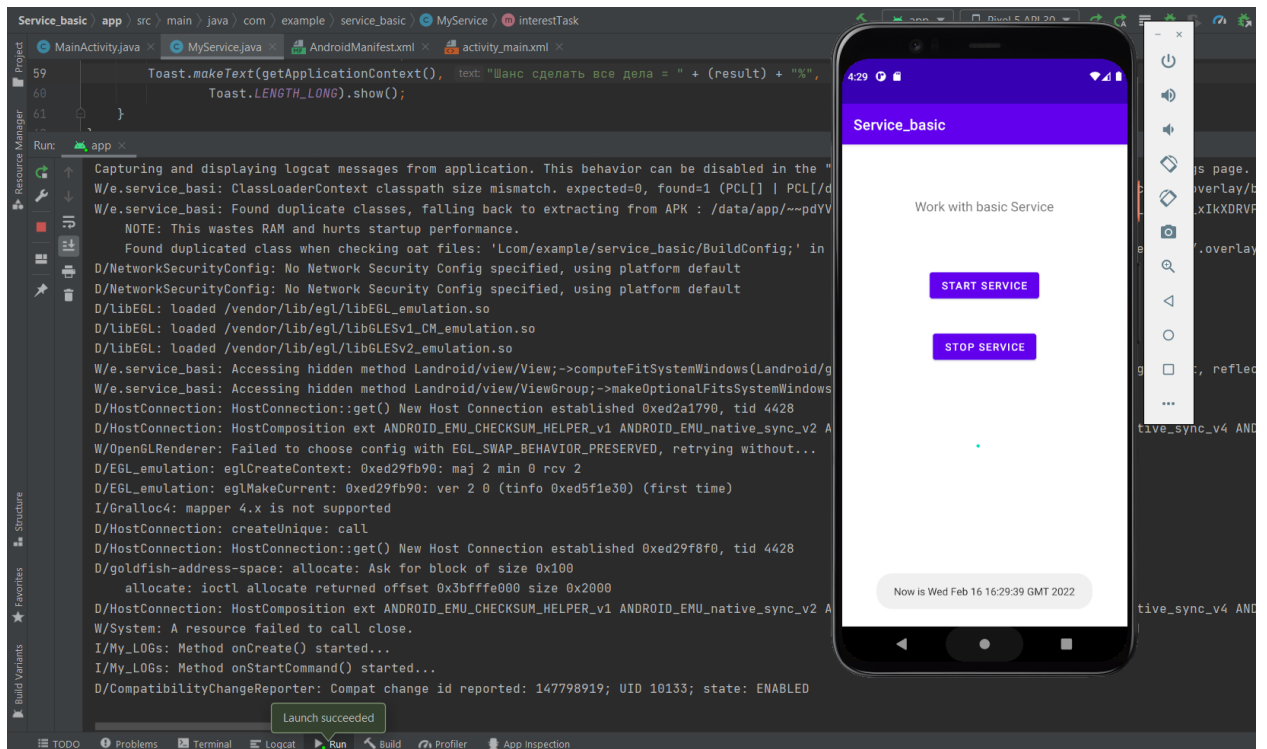
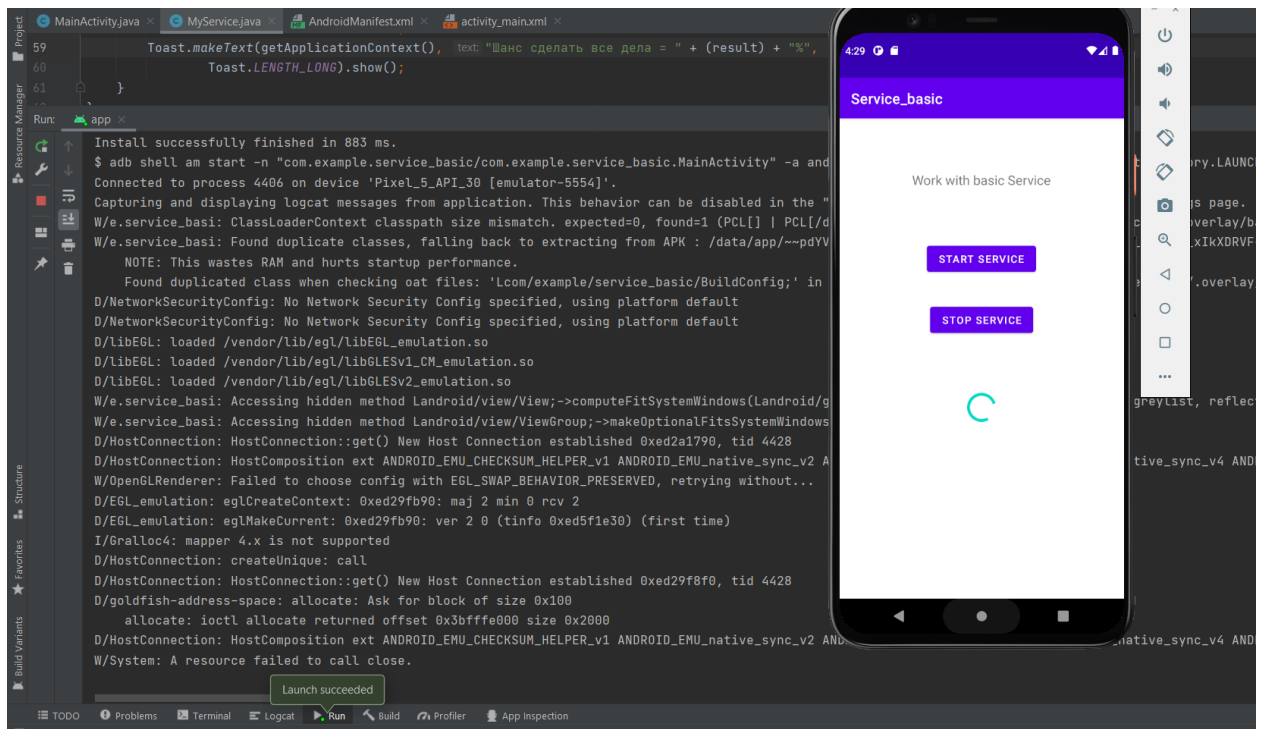


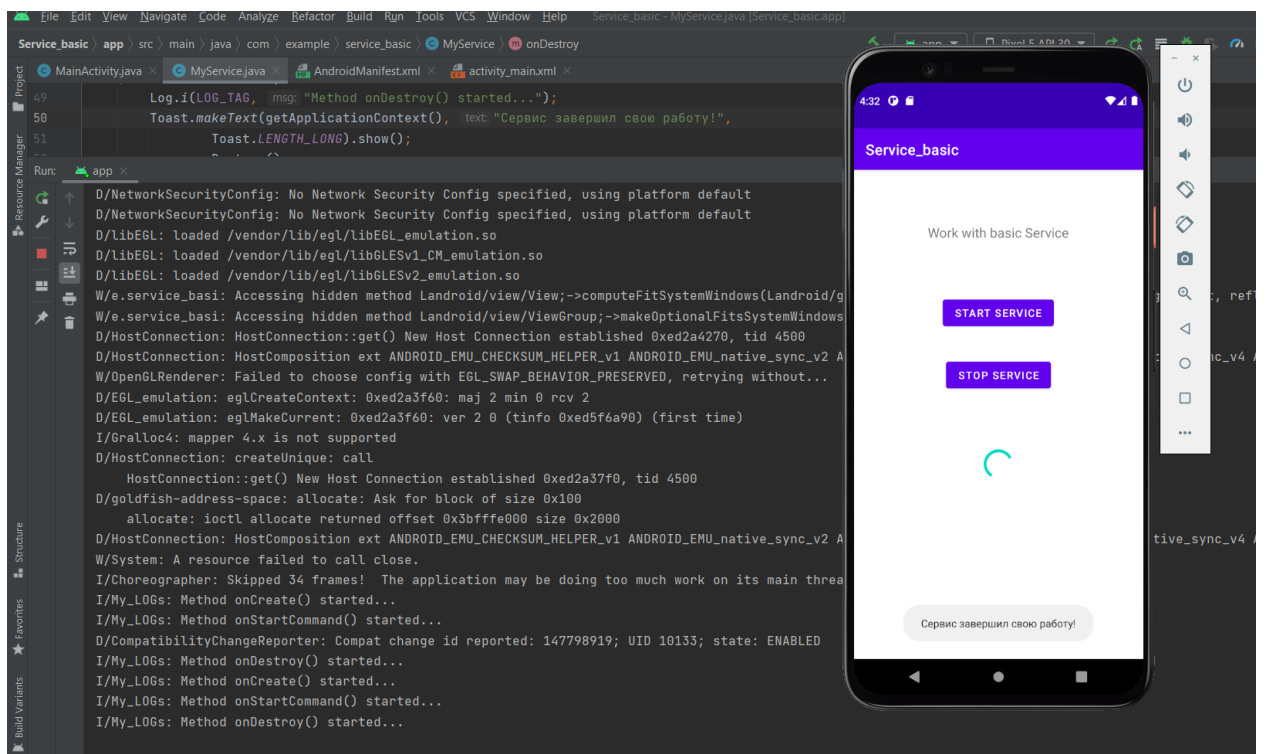
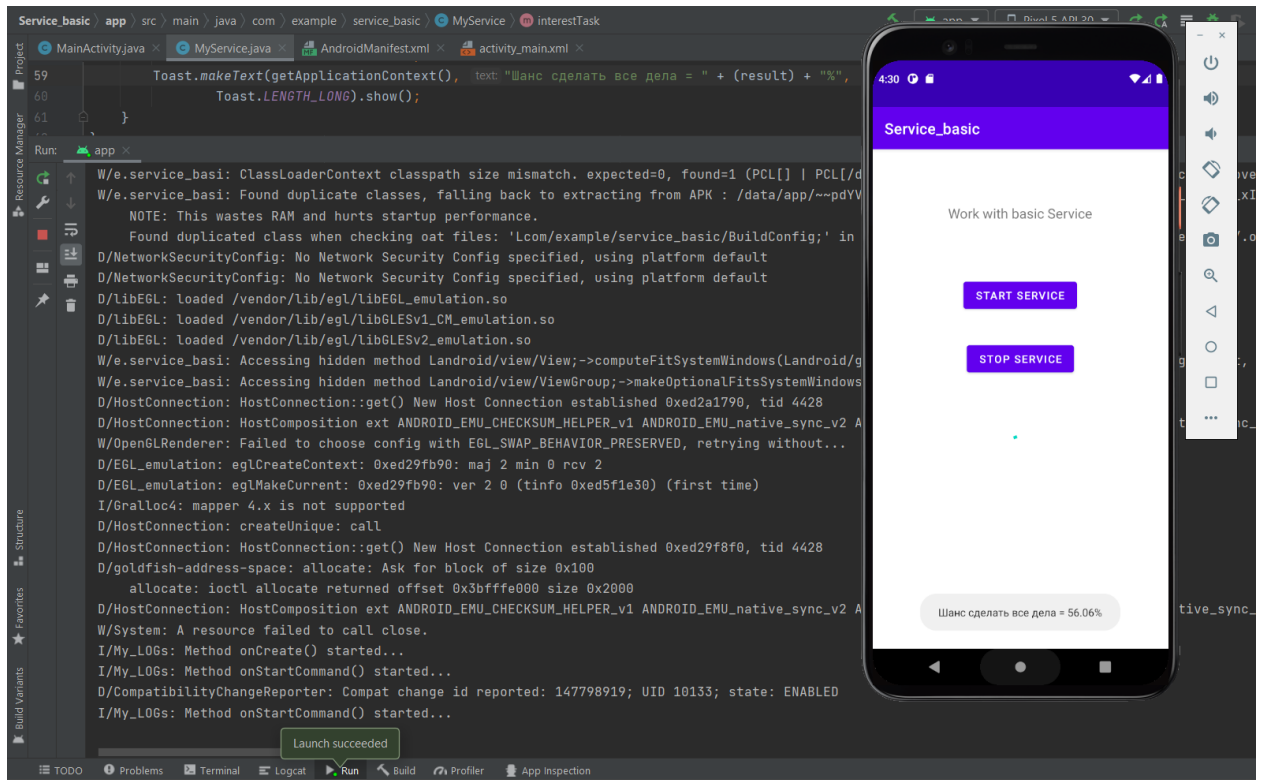
```
1 package com.example.service_basic;
2
3 import ...
4
5
6
7
8
9
10 public class MainActivity extends AppCompatActivity {
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_main);
16     }
17
18     public void StartService(View v){
19         startService(new Intent( packageContext: this, MyService.class));
20     }
21
22     public void StopService(View v){
23         stopService(new Intent( packageContext: this, MyService.class));
24     }
25 }
```

Здесь всё довольно просто: создаём 2 метода для запуска и остановки сервиса через методы `startService()` и `stopService()`, в каждый из которых передаём интент с уже созданным Java-классом `MyService`, который унаследовал все методы и свойства класса `Service`.

Таким образом мы реализовали 1 вариант сервиса с полезным действием (несвязный сервис, работающий в фоновом режиме).

Осталось только увидеть сам результат работы:





Контрольные вопросы

1. Когда происходит создание и удаление связанного сервиса?
2. Какие методы жизненного цикла вызываются у сервиса и в какой последовательности?
3. Как соотносится ЖЦ активности и сервиса?

1. Сам процесс создания связанного сервиса происходит при вызове метода `bindService(intent, connection, context)`, удаление при вызове `unbindService(connection)`.
2. В начале создаётся сам сервис через `onCreate()`, затем идёт либо сама работа сервиса (его функционал через базовый метод `onStartCommand()`), либо начинается работа связанного с клиентом сервиса через уже метод `onBind()`.

В это время идёт активный период работы сервиса (и в фоновом режиме, в том числе). В первом случае он заканчивается принудительным клиентским вызовом (или автоматическим способом в рамках работы самого сервиса – самовыключение) метода `onDestroy()`.

На этом он заканчивает свою работу в рамках несвязанного сервиса.

А уже во 2 случае у нас идёт вызов клиентом `onUnbind()` и дальнейшим отключением через метод `onDestroy`, как и в 1 случае. Таким образом распределены все методы в рамках жизненного цикла сервиса в 2 вариантах: несвязанный и связанный сервисы.

3. У них обоих есть методы `onCreate()` и `onDestroy()`, однако соответствие между работой активности и сервиса следующее:

Activity 1

