Лабораторная работа

Использование объектов Intent

1 Запуск Activity с помощью явного объекта Intent

Краткие теоретические сведения

В предыдущих лабораторных работах все приложения состояли из одного Activity. Реальные приложения, как правило, имеют несколько окон - состоят из нескольких Activity, которыми надо уметь управлять и которые должны взаимодействовать между собой.

Один из Activity приложения (окно которого открывается при запуске приложения) помечен как *главный*. Из открытого Activity можно запустить другой Activity, даже если он определен в другом приложении. Пользователю будет казаться, что все запускаемые им Activity являются частями одного приложения, хотя на самом деле они могут быть определены в разных приложениях и работать в различных процессах.

Компоненты Android-приложений, в том числе и Activity, запускаются через объекты `Intent`.

`Intent` (в переводе с английского ***Намерение)*** в операционной системе Android - это программный механизм, который позволяет пользователям координировать функции различных действий для достижения цели. `Intent` представляет собой объект описания операции, которую необходимо выполнить через систему Android. Т.е. необходимо сначала описать некоторую операцию в виде объекта `Intent`, после чего отправить её на выполнение в систему вызовом одного из методов активности Activity.

В каждом случае система Android находит соответствующий Activity, чтобы ответить на `Intent`, инициализируя его в случае необходимости.

Объекты `Intent` могут быть разделены на две группы:

- ***явный*** `Intent` - определяет целевой компонент по имени;
- ***неявный*** `Intent` - не называет адресата.

Явный `Intent` обычно используют для сообщений внутри приложения, например, когда один Activity запускает другой Activity из этого приложения.

Объект Intent является структурой данных, содержащей абстрактное описание выполняемой операции. Чтобы вызвать другой Activity, в объект Intent надо передать имя этого Activity. Имя устанавливается методами `setComponent`(), `setClass`() или `setClassName`().

Далее, чтобы запустить Activity, объект Intent передают в метод *Context.startActivity*(). Этот метод принимает единственный параметр - объект Intent, описывающий Activity, который будет запускаться.

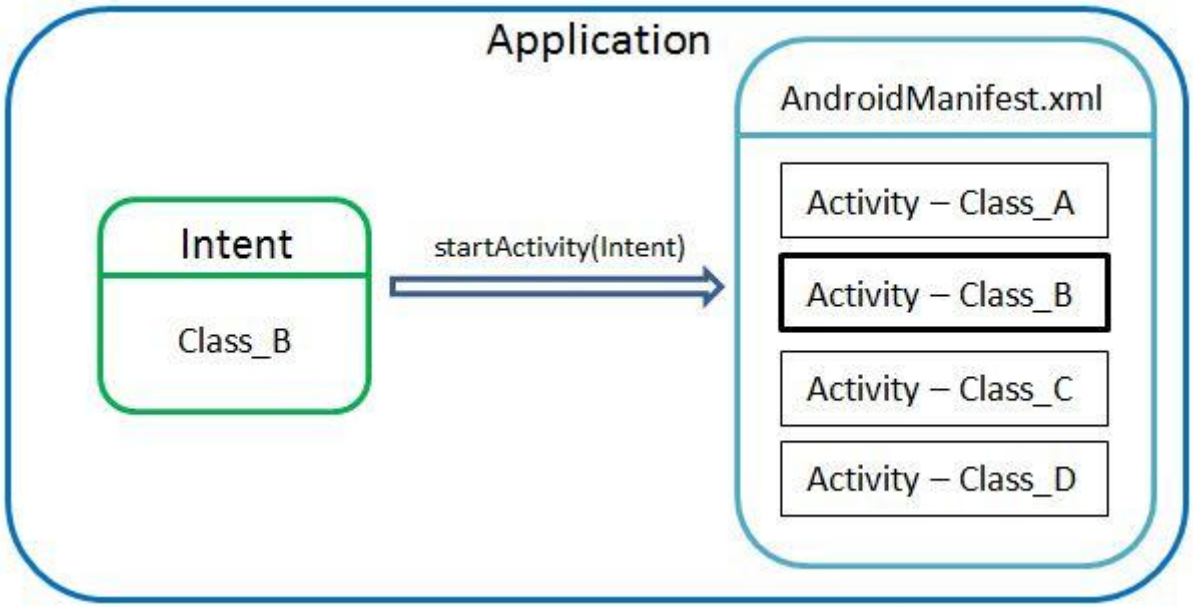
Например, вызвать Activity с именем *NewActivity* в коде программы можно следующим образом:

```
// Создаем объект Intent
Intent intent = new Intent();
// Устанавливаем имя вызываемого компонента
intent.setClass(getApplicationContext(), NewActivity.class);
// Запускаем компонент
startActivity(intent);
```

В качестве первого параметра конструктора *Intent* указывается *Context*. Поскольку активность является наследником *Context*, то можно использовать укороченную запись *this* или полную как *Class_name.this*.

Во втором параметре конструктора указывается наименование класса активности. Этот класс зарегистрирован в манифесте приложения. Таким образом, приложение может иметь несколько активностей, каждую из которых можно вызвать по наименованию класса. После вызова метода *startActivity* будет создана новая активность, которая запустится или возобновит свою работу, переместившись на вершину стека активностей.

```
Intent intent = new Intent(this, SecondActivity.class);
startActivity(intent);
```



Неявный *Intent* применяют для запуска компонентов других приложений. Неявные *Intent* часто используют, чтобы активизировать компоненты в других приложениях.

Неявный вызов Activity отличается тем, что при создании Intent используется не класс, а заполняются параметры ***action (действия), data (данные), category (категория)*** определенными значениями. Комбинация этих значений определяют цель, которую требуется достичь. Например, отправка письма, открытие гиперссылки, редактирование текста, просмотр картинки, звонок по определенному номеру и т.д.

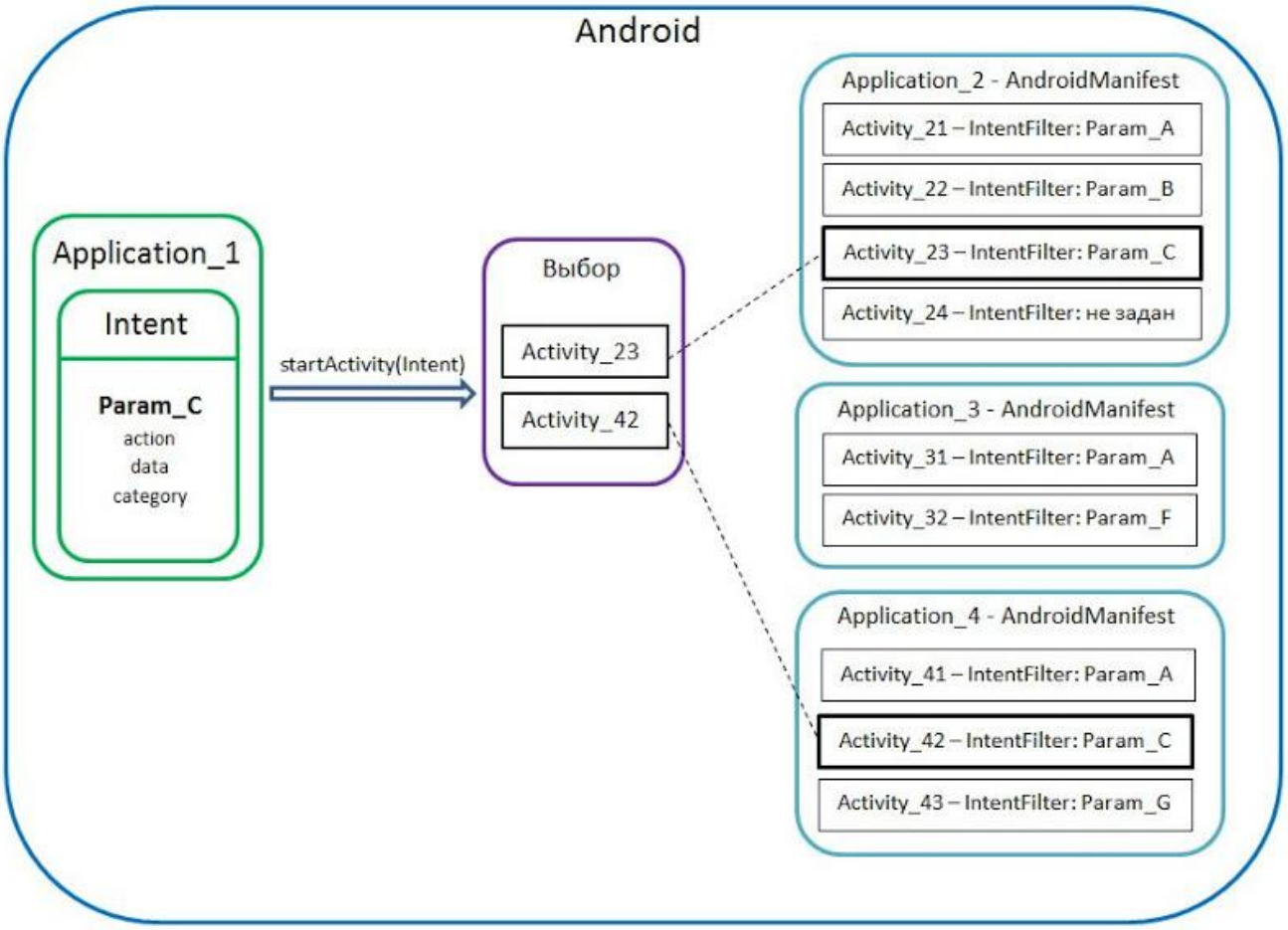
Для Activity прописывается Intent Filter - это набор тех же параметров: `action, data, category` (но значения уже свои - зависят от того, что умеет делать Activity). Фильтр `Intent` объявляется в файле манифеста приложения.

И если параметры нужного Intent совпадают с условиями этого фильтра, то Activity вызывается. Но при этом поиск уже идет по всем Activity всех приложений в системе.

Если находится несколько, то система предоставляет выбор, какой именно программой пользователь хочет воспользоваться. Схематично это можно описать так.

В Application_1 создается Intent, заполняются параметры *action, data, category*. Для удобства, получившийся набор параметров назовем Param_C. С помощью `startActivity` этот Intent отправляется на поиски подходящей Activity, которая сможет выполнить то, что нам нужно (т.е. то, что определено с помощью Param_C). В системе есть разные приложения, и в каждом из них несколько Activity. Для некоторых Activity определен Intent Filter (наборы Param_A, Param_B и т.д.), для некоторых нет. Метод `startActivity` сверяет набор параметров Intent и наборы параметров Intent Filter для каждой Activity. Если наборы совпадают (Param_C для обоих), то Activity считается подходящей.

Если в итоге нашлась только одна Activity – она и отображается. Если же нашлось несколько подходящих Activity, то пользователю выводится список, где он может сам выбрать какое приложение ему использовать.



Например, если в системе установлено несколько музыкальных плееров, и вы запускаете mp3, то система выведет вам список Activity, которые умеют играть музыку и попросит выбрать, какое из них использовать. А те Activity, которые умеют редактировать текст, показывать картинки, звонить и т.п., будут проигнорированы.

Фильтр `Intent` объявляется в манифесте приложения в элементе `<intent-filter>`. Например, в любом приложении есть главный Activity, который устанавливается как точка входа для задания:

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Фильтр такого вида в элементе помечает Activity, как запускаемый по умолчанию. Элемент заставляет значок и метку для Activity отображаться на панели Application Launcher, давая пользователям возможность запускать задание и возвращаться к этому заданию в любое время после того, как оно было запущено.

ОБРАТИТЕ ВНИМАНИЕ!

Фильтр `Intent` - экземпляр класса `IntentFilter`. Однако, так как система Android должна узнать о возможностях компонента прежде, чем она сможет запустить этот

компонент, фильтры Intent всегда устанавливаются только в файле манифеста приложения как элементы, а не в коде приложения.

У фильтра есть поля, которые соответствуют действию, данным и категориям объекта `Intent`. Неявный `Intent` проверяется в фильтре по всем трем полям.

Чтобы `Intent` запустил компонент, которому принадлежит фильтр, он должен пройти все три теста. Если один из них не проходит, то система не запустит этот компонент - по крайней мере, на основе этого фильтра. Однако, так как у компонента может быть несколько фильтров, `Intent`, который не проходит через один из фильтров компонента, может пройти через другой. Каждый фильтр описывает возможность компонента и набор `Intent`, которые компонент желает получать.

Упражнение 1. *Создать приложение, отображающее события, происходящие при переключении между Activity*

1) создайте проект *ActivityEvents*.
2) в файл компоновки добавьте кнопку *Call Activity* (вызов активности), текстовое поле для отображения событий, происходящих с главной активностью при переключении между двумя активностями.
3) создайте вторую Activity с именем `SecondActivity`. Для этого щелкните правой кнопкой мыши на значке пакета `com.samples.app.activityevents`, в котором находится файл `MainActivity`, и в контекстном меню выберите пункт `New|Java Class`. Класс `SecondActivity` будет пустым, он не будет иметь даже своего файла компоновки. Отредактируйте его как показано в листинге 1.

Листинг 1. Код класса SecondActivity.java

```
public class SecondActivity extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.setTitle("Second Activity");
    }
}
```

4) чтобы приложение могло "видеть" вторую Activity, необходимо указать ее в файле манифеста приложения (листинг 2).

Листинг 2. Файл AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.applr18">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="ActivityEvents"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
```

```
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:name="SecondActivity">
        </activity>

    </application>

</manifest>
```

5) теперь в приложении будут две Activity с именами *MainActivity* и *SecondActivity*. В классе *MainActivity* переопределите все защищенные методы класса Activity для отслеживания событий, происходящих в Activity. Код класса MainActivity представлен в листинге 3.

Листинг 3. Файл класса окна приложения MainActivity.java

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    private TextView text;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        text = (TextView)findViewById(R.id.text);
        text.append("onCreate()\n");
    }
    @Override
    public void onRestart() {
        super.onRestart();
        text.append("onRestart()\n");
    }
    @Override
    public void onStart() {
        super.onStart();
        text.append("onStart()\n");
    }
    @Override
    public void onResume() {
        super.onResume();
        text.append("onResume()\n");
    }
    @Override
    public void onPause() {
        super.onPause();
        text.append("onPause()\n");
    }
    @Override
    public void onStop() {
        super.onStop();
        super.onStart();
        text.append("onStop()\n");
    }
    @Override
    public void onDestroy() {
```

```
            super.onDestroy();
            text.append("onDestroy()\n");
        }
        @Override
        public void onClick(View arg0) {
            Intent intent = new Intent();
            intent.setClass(getApplicationContext(), SecondActivity.class);
            startActivity(intent);
        }
    }
```

6) скомпилируйте приложение и запустите его на эмуляторе Android. При запуске приложения главная Activity отработает последовательно три состояния: *onCreate*(), *onStart*(), *onResume*().
7) Если нажать кнопку *Call Activity*, запустится второй *Activity* приложения - *SecondActivity*.
8) Если теперь нажать клавишу <BACK> на эмуляторе, на экран снова будет выведен *MainActivity*. Пока *MainActivity* находился в остановленном состоянии, были последовательно отработаны события *onPause*(), *onStop*(), а при возвращении *MainActivity* на передний план - события *onCreate*(), *onStart*(), *onResume*(), как представлено на рис. 1.

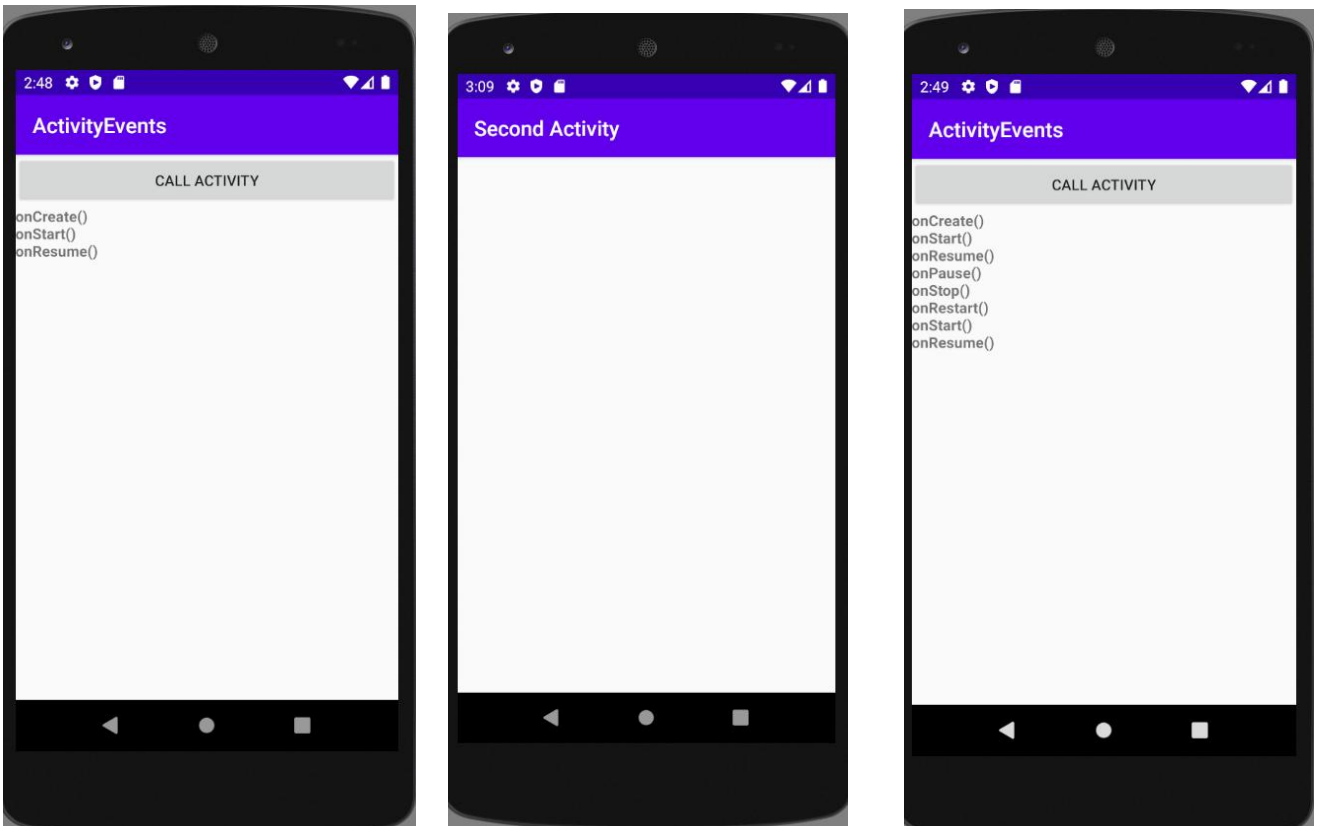


Рисунок 1 - События, происходящие при переключении Activity

9) ***Самостоятельно*** измените приложение так, чтобы отработанные события отображались и в дочернем окне *SecondActivity*.

2 Обмен данными между Activity

Краткие теоретические сведения

Обмен данными между Activity актуален для большинства приложений. Иногда требуется вернуть результат выполнения из вызываемого Activity в вызывающий Activity при его закрытии. Например, можно запустить Activity, который позволяет пользователю выбирать человека в списке контактов. При закрытии Activity возвращает данные человека, который был выбран: его полное имя и телефон.

2.1 Передача простых типов

Чтобы запустить Activity и получить результат его выполнения, необходимо вызвать метод `startActivityForResult(Intent, int)` со вторым параметром, идентифицирующим запрос.

Результат возвращается через метод `onActivityResult(int, int, Intent)`, определенный в родительском Activity.

Кроме этого, можно передавать дополнительные параметры (extra-параметры) в вызываемый Activity. Это пары *ключ-значение* для информации, которую нужно предоставить вызываемому компоненту. Объект Intent имеет ряд методов `put...()` для вставки различных типов дополнительных данных и аналогичный набор методов `get...()` для чтения данных.

Extra-параметры устанавливаются и читаются как объекты класса `Bundle` с использованием методов `putExtras()` и `getExtras()`.

Например, запустить Activity с именем класса `SecondActivity` и передать ему два дополнительных параметра можно следующим образом:

```
// Идентификатор запроса
private static final int ACTION_EDIT = 101;
// Идентификаторы extra-параметров
private static final String ID_EXTRA_1 = "Param1";
private static final String ID_EXTRA_2 = "Param2";
...
// Создаем объект Intent
Intent intent = new Intent();
// Добавление extra-параметров
String param1 = "Some Text... ";
int patam2 = "12345";
intent.putExtra(ID_EXTRA_1, param1);
intent.putExtra(ID_EXTRA_2, param2);
...
// Определение класса, запускаемого Activity
intent.setClass(this, SecondActivity.class);
```

```
// Вызов Activity
startActivityForResult(intent, ACTION_EDIT);
```

В дочернем Activity можно прочитать переданные extra-параметры, получив сначала объект класса *Bundle* как набор параметров, представляющих собой пары *ключ-значение*, а затем приводя их к нужному типу данных с помощью методов класса *Bundle: get(), getBoolean(), getString()* и т. д. В эти методы в качестве параметра передается идентификатор параметра, который является уникальным ключом для всего набора пар ключ-значение. Эти ключи определяются теми же параметрами, которые были определены в вызывающем Activity (в данном случае - это *ID_EXTRA_1* и *ID_EXTRA_2*).

В целом код обработки может выглядеть примерно так:

```
@Override public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
// Получаем объект Bundle и читаем параметры
Bundle extras = getIntent().getExtras();
String param1 = extras.getString(MainActivity.ID_EXTRA_1));
int param2 = extras.getInt(MainActivity.ID_EXTRA_2));
}
```

Когда дочерний Activity закроется, в нем можно вызвать метод *setResult(int)*, чтобы возвратить данные в родительский Activity. Этот метод возвращает код результата закрытия Activity, который может быть стандартным результатом, определяемым константами *RESULT_CANCELED*, *RESULT_OK* или определяемым пользователем результатом *RESULT_FIRST_USER*.

```
Intent intent = new Intent();
intent.putExtra(MainActivity.ID_EXTRA_1, param1);
intent.putExtra(MainActivity.ID_EXTRA_1, param2);
// Возвращаем результат в вызывающий Activity
setResult(RESULT_OK, intent);
finish();
```

Кроме того, дочерний Activity может произвольно возвратить назад объект *Intent*, содержащий любые дополнительные данные. Вся эта информация в родительском Activity появляется через метод обратного вызова *Activity.onActivityResult*() вместе с идентификатором, который был передан в метод *startActivityForResult*() при вызове Activity:

```
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
 super.onActivityResult(requestCode, resultCode, data);
```

```
if (resultCode == RESULT_OK) {
Bundle extras = data.getExtras();
switch (requestCode) {
case IDM_ADD:
String param1 = extras.getString(ID_EXTRA_1), int param2 =
extras.getInt(ID_EXTRA_2);
...
 break;
...
}
...
}
}
```

Если дочерний Activity завершится неудачно или будет закрыт пользователем без подтверждения ввода, то родительский Activity получит результат с кодом `RESULT_CANCELED`.

Упражнение 2. *Создать приложение ContactEditor для редактирования контактов. В главном окне будут данные телефонного контакта, который можно редактировать в другом окне. Результат редактирования должен возвращаться в главное окно*

1) создайте проект *ContactEditor*. Всего в приложении будет две Activity:
   - `MainActivity` - главное окно с данными контакта для редактирования – `Имя и телефон` и кнопкой `Edit Data` для вызова окна редактирования;
   - `SecondActivity` – окно для редактирования контакта.
2) добавьте в приложение вторую активность `SecondActivity`. Для этого нажмите правой кнопкой мыши в папку, в которой находится класс MainActivity, и затем в контекстном меню выберите `New->Activity->Empty Activity`.
3) в файле компоновки главного окна приложения *activity_main.xml* отобразите данные контакта: имя человека, его телефон и кнопку ***Edit Data*** для вызова дочернего `SecondActivity` для модификации контакта. Полный код файла компоновки представлен в листинге 4.

Листинг 4. Файл компоновки главного окна приложения activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Name: "
            android:textSize="18sp"
            android:layout_margin="2sp"/>
```

```
        <TextView
            android:id="@+id/textName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:layout_margin="2sp" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Phone: "
            android:textSize="18sp"
            android:layout_margin="2sp" />
        <TextView
            android:id="@+id/textPhone"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:layout_margin="2sp" />
    </LinearLayout>
    <Button
        android:id="@+id/bEditData"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="Edit Data" />
</LinearLayout>
```

4) в классе *MainActivity* реализуйте вызов дочернего *SecondActivity* и передачу ему двух строковых параметров - NAME и PHONE (листинг 5).

Листинг 5. Файл класса MainActivity.java для проекта ContactEditor

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    private TextView textName;
    private TextView textPhone;

    // Константы, идентифицирующие extra-параметры
    public final static String NAME = "Name";
    public final static String PHONE = "Phone";
    private final static int ACTION_EDIT = 101;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        textName = (TextView)findViewById(R.id.textName);
        textPhone = (TextView)findViewById(R.id.textPhone);
        textName.setText("Andrew Ivanov");
        textPhone.setText("123456789");
    }

    @Override
    public void onClick(View arg0) {
        Intent intent = new Intent(getApplicationContext(), SecondActivity2.class);

        // Добавляем extra-параметры
```

```
        intent.putExtra(NAME, textName.getText());
        intent.putExtra(PHONE, textPhone.getText());
        // Вызываем Activity
        startActivityForResult(intent, ACTION_EDIT);
    }

}
```

5) также потребуется реализовать метод `onActivityResult`() для чтения результата из дочернего `SecondActivity`.

Метод ***onActivityResult*** принимает три параметра:

- `requestCode`: числовой код запроса, который был отправлен вторым параметром в `startActivityForResult`
- `resultCode`: числовой код результата. В качестве результата, как правило, применяются встроенные константы `RESULT_OK` и `RESULT_CANCELED`.
- `data`: отправленные данные из `SecondActivity` в `MainActivity`.

В данном случае в методе `onActivityResult`() выводятся полученные данные в элементы `EditText` (листинг 6).

Листинг 6. Код метода onActivityResult()

```
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode == RESULT_OK) {
    Bundle extras = data.getExtras();
    textName.setText(extras.getString(NAME));
    textPhone.setText(extras.getString(PHONE));
}
else
{
    super.onActivityResult(requestCode, resultCode, data);
}
    }
```

6) класс `SecondActivity` представляет функциональность для редактирования контакта: два текстовых поля для редактирования имени и телефона и командные кнопки для сохранения изменений ***Save*** или отмены ввода ***Cancel*** и перехода в основной Activity.

Код файла компоновки для окна редактирования контакта `activity_second.xml` представлен в листинге 7.

Листинг 7. Файл компоновки дочернего окна activity_second.xml

```
<?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:orientation="vertical" >
    <LinearLayout
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" >
        <TextView
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:text="Name:"/>
        <EditText
            android:id="@+id/editName"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent"/>
    </LinearLayout>
    <LinearLayout
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" >
        <TextView
            android:text="Pnone:"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
        <EditText
            android:id="@+id/editPhone"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent"/>
    </LinearLayout>
    <LinearLayout
        android:layout_height="wrap_content"
        android:layout_width="fill_parent">
        <Button
            android:id="@+id/bSave"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent"
            android:layout_weight="1"
            android:onClick="onClick"
            android:text="Save"/>
        <Button
            android:id="@+id/bCancel"
            android:text="Cancel"
            android:layout_width="fill_parent"
            android:layout_weight="1"
            android:layout_height="wrap_content"
            android:onClick="onClick"/>
    </LinearLayout>
</LinearLayout>
```

7) класс *SecondActivity* должен принимать от вызывающего Activity данные редактируемого контакта и передавать отредактированные данные обратно в родительский Activity.

Для возврата результата необходимо вызвать метод ***setResult***(), в который передается два параметра:

- числовой код результата;
- отправляемые данные.

После вызова метода *setResult*() нужно вызвать метод ***finish***, который уничтожит текущую activity.

Если нажата кнопка *Cancel*, то в *setResult* передается только код результата - RESULT_CANCELED.

То есть условно говоря, *SecondActivity* получает данные из *MainActivity* и после нажатия кнопки *Save* результат возвращается в *MainActivity*.

Код класса *SecondActivity* представлен в листинге 8.

Листинг 8. Файл класса дочернего окна SecondActivity.java

```
public abstract class SecondActivity extends AppCompatActivity  implements
View.OnClickListener {

    private EditText editName;
    private EditText editPhone;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.setTitle("Second Activity");
        setContentView(R.layout.second);
        editName = (EditText)findViewById(R.id.editName);
        editPhone = (EditText)findViewById(R.id.editPhone);
        Bundle extras = getIntent().getExtras();
        editName.setText(extras.getString(MainActivity.NAME));
        editPhone.setText(extras.getString(MainActivity.PHONE));
    }
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.bSave:
                Intent intent = new Intent();
                intent.putExtra(MainActivity.NAME, editName.getText().toString());
                intent.putExtra(MainActivity.PHONE, editPhone.getText().toString());
                setResult(RESULT_OK, intent);
                finish();
                break;
            case R.id.bCancel:
                setResult(RESULT_CANCELED);
                finish();
                break;
        }
    }
}
```

8) при запуске приложения на экране устройства появляется окно главного Activity с контактными данными (рис.2).

Рисунок 2

9) при нажатии кнопки *Edit Data* будет запущен дочерний `SecondActivity` с текстовыми полями для редактирования данных (рис.3).

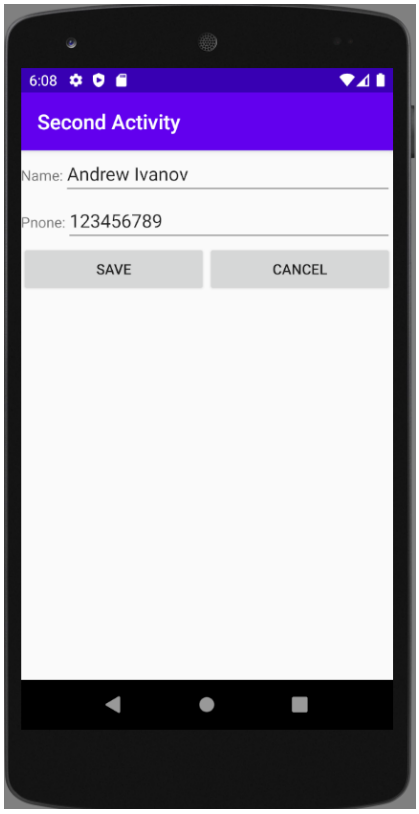


Рисунок 3

10) в зависимости от результата закрытия окна (кнопкой Save или Cancel), измененные данные будут прочитаны в родительском Activity и выведены на экран (рис. 4).

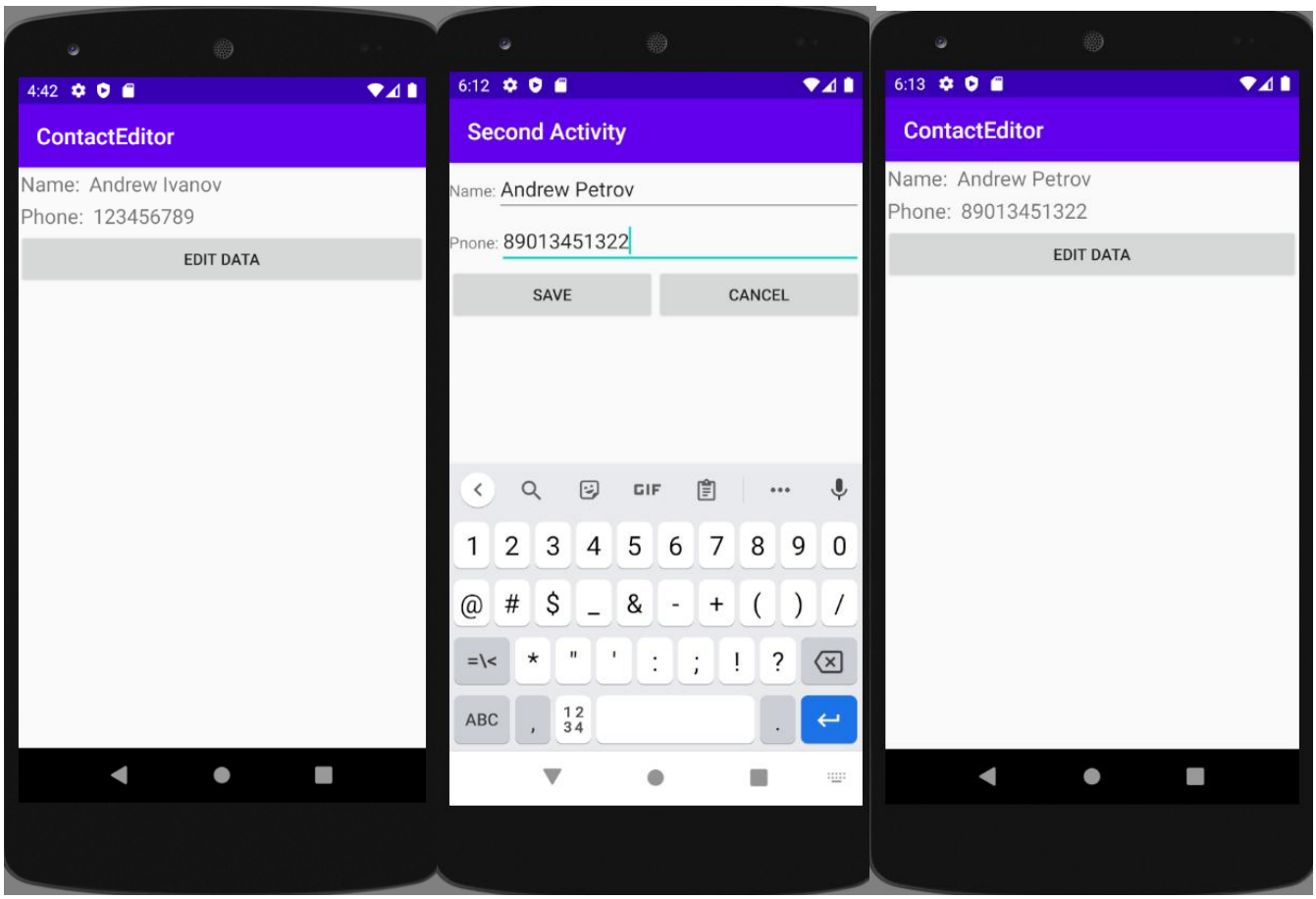


Рисунок 4 - Обмен данными между Activity

2.2 Передача объектов

Через объекты `Intent` можно передавать сложные объекты, например, изображения. Отправка изображения через `Intent` аналогична передаче простых типов.

```
Intent intent = new Intent();
intent.setClass(this, SecondActivity.class);
intent.putExtra("Bitmap", bitmap);
```

Упражнение 3. *Создать приложение, которое будет передавать картинку в другое окно*

1) создайте проект `SendBitmapBetweenActivities`.
2) добавьте в проект еще один `Activity` с именем `SecondActivity`
3) в файл компоновки главного окна приложения поместите элемент `ImageView` и кнопку *Send Bitmap* (листинг 9)

Листинг 9. Файл компоновки главного окна приложения activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <ImageView
        android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:src="@drawable/ic_launcher" />
    <Button
        android:id="@+id/bSendBitmap"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="Send bitmap" />
</LinearLayout>
```

4) в файл компоновки окна-приемника картинки *activity_second.xml* приложения поместите элемент *ImageView* (листинг 10).

Листинг 10. Файл компоновки главного окна приложения activity_second.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:gravity="center"
    android:layout_height="fill_parent">
    <ImageView
        android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

5) в классе *MainActivity* в обработчике *onClick*() кнопки создайте объект *Intent*, который будет вызывать *SecondActivity* и передавать ему объект *Bitmap* через *extra*-параметр (листинг 11).

Листинг 11. Класс главного окна приложения MainActivity.java

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public void onClick(View v) {
        Bitmap bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.android);
        Intent intent = new Intent();
        intent.setClass(this, SecondActivity.class);
        intent.putExtra("Bitmap", bitmap);
        startActivity(intent);
    }
}
```

6) в классе *SecondActivity* надо прочитать *extra*-параметр с помощью метода *getParcelableExtra*() и привести его к типу *Bitmap*, а после загрузить его в *ImageView* (листинг 12).

Листинг 12. Класс окна SecondActivity.java

```
public class SecondActivity extends AppCompatActivity {

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);

        Bitmap bitmap = (Bitmap)getIntent().getParcelableExtra("Bitmap");
        ImageView image = (ImageView)findViewById(R.id.image);
        image.setImageBitmap(bitmap);
    }
}
```

7) запустите приложение и протестируйте его (рис. 5).

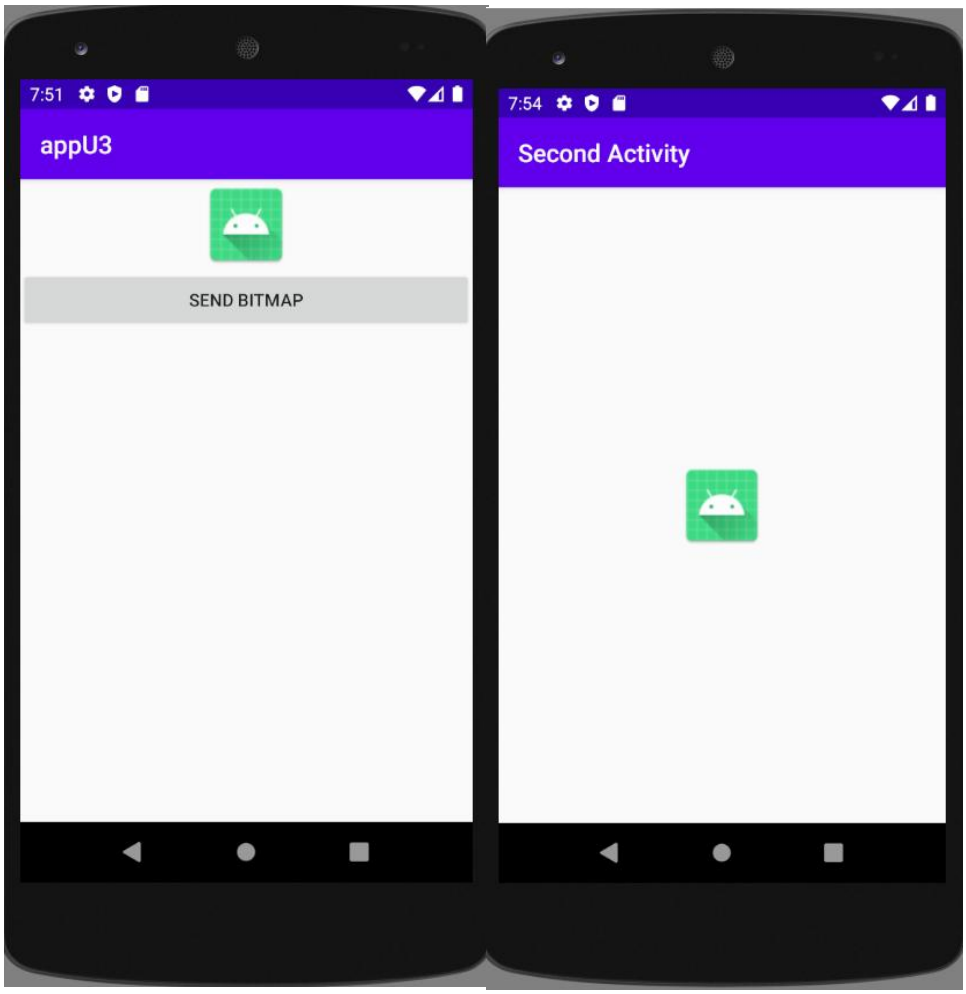


Рисунок 5 - Передача объекта Bitmap в другой Activity

Замечание. Таким способом можно передавать объекты, которые реализуют интерфейс *Parcelable*. Класс *Bitmap* его и реализует. Но если вы захотите таким способом передавать объекты какого-либо своего класса, надо будет реализовать в нем интерфейс *Parcelable*.

2.3 Вызов Activity из другого приложения

Для вызова Activity из другого приложения используется *неявный* объект `Intent`. Этот способ вызова Activity основан на том, что Activity вызывается не по имени, а по функционалу.

Для создания `Intent` используется конструктор: `Intent (String action)`. Т.е. при создании заполняется атрибут объекта `Intent`, который называется `action`. Это обычная строковая константа. `action` обычно указывает действие, которое требуется произвести. Например, есть следующие системные action-константы: ACTION_VIEW - просмотр, ACTION_EDIT – редактирование, ACTION_PICK – выбор из списка, ACTION_DIAL – сделать звонок.

Если действие производится с чем-либо, то в пару к `action` идет еще один `Intent`-атрибут – `data`. В нем можно указать какой-либо объект: пользователь в адресной книге, координаты на карте, номер телефона и т.п. Т.е. `action` указывает что делать, а `data` – с чем делать.

Итак, `action` – это просто текст.

После создания `Intent` с `action` его запускают в систему искать нужную `Activity`. Чтобы Activity подошла, надо чтобы ее `Intent Filter` содержал атрибут `action` с тем же значением, что и `action` в `Intent`.

Упражнение 4. *Создать приложение ContactLauncher, которое будет вызывать MainActivity приложения ContactEditor*

1) откройте проект *ContactEditor*
2) измените файл манифеста приложения *ContactEditor*, добавив дополнительный фильтр `Intent` к `Activity` (листинг 13)

Листинг 13. Изменения в файле манифеста приложения AndroidManifest.xml

```
    ……
<intent-filter>
    <action android:name="VIEW_CONTACTS" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>

….
```

Этот фильтр нужен, чтобы система смогла разыскать и запустить необходимый Activity среди множества приложений, содержащих Activity, которые были инсталлированы на этом устройстве.

Строка `action android:name="VIEW_CONTACTS"` определяет имя действия.

Для вызова Activity из другого приложения необходимо создать объект `Intent` и передать ему в качестве параметра строку "`VIEW_CONTACTS`", определенную в Intent-фильтре компонента вызываемого приложения.

3) создайте проект `ContactLauncher`.
4) приложение будет представлять окно с одной кнопкой *Launch Contact Editor*. Файл компоновки приложения приведен в листинге 14

Листинг 14. Файл компоновки главного окна приложения activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center">
    <Button
        android:id="@+id/btn_launch"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Launch Contact Editor"
        android:onClick="onClick"/>
</LinearLayout>
```

5) в коде класса *MaimActivity* в обработчике события кнопки создайте объект Intent с действием*VIEW_CONTACTS*. Вызов метода *startActivity*() с созданным объектом *Intent* в качестве входного параметра запустит Activity из приложения для редактирования контактов. Код класса *MaimActivity* представлен в листинге 15.

Листинг 15. Файл MaimActivity .java класса окна приложения

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    // Константа, идентифицирующая вызываемый Activity
    private final static String ACTION_VIEW_CONTACTS ="VIEW_CONTACTS";
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public void onClick(View v) {
        // Вызов Activity приложения ContactEditor
        startActivity(new Intent(ACTION_VIEW_CONTACTS));
    }
}
```

6) Запустите проект на выполнение. При нажатии кнопки должен запуститься главный Activity приложения *ContactEditor* (рис. 5).

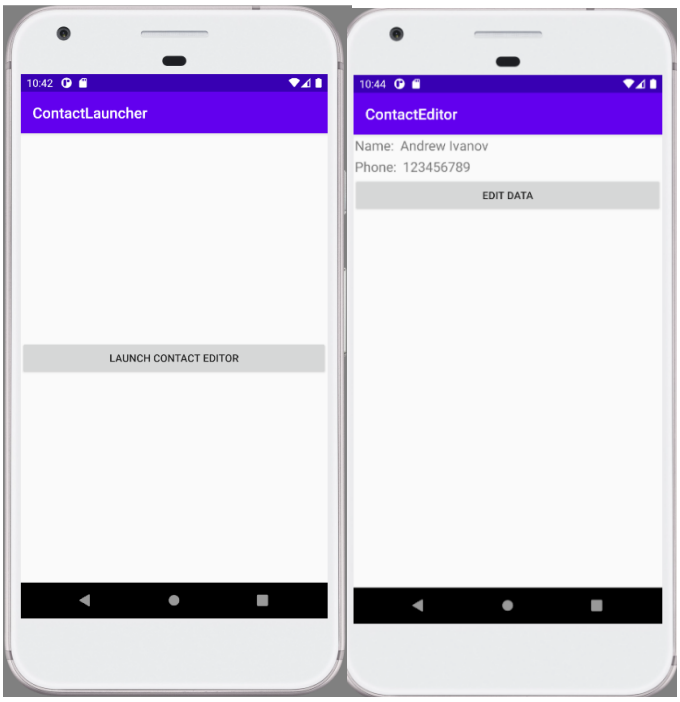


Рисунок 5 - Запуск Activity из другого приложения

Таким образом можно запускать любой Activity, добавив соответствующие изменения в файл манифеста, аналогично листингу 13. Точно так же система Android позволяет запускать Activity посторонних приложений.

2.4 Переход на домашний экран

Программный переход на домашний экран также требует создания неявного объекта Intent. Это значит, что задается не имя класса, а определенные действия (ACTION), объявленные как строковые константы в классе Intent. Действие, которое нужно задать - ACTION_MAIN.

```
Intent intent = new Intent(Intent.ACTION_MAIN);
```

Но это действие прописано во всех приложениях, имеющих GUI, и, соответственно, главное окно, в том числе и домашний экран. Поэтому, кроме действия, надо также установить категорию для объекта Intent. Для перехода на домашний экран это будет CATEGORY_HOME:

```
intent.addCategory(Intent.CATEGORY_HOME);
```

Упражнение 5. *Создать приложение Intent_BackToHomeScreen, в котором по кнопке осуществляется переход на домашний экран*

1) создайте проект `Intent_BackToHomeScreen`
2) в файл компоновки главного окна поместите кнопку `To home screen`
3) в классе MainActivity создайте объект Intent с действием ACTION_MAIN и категорией CATEGORY_HOME.

4) запустите приложение. При нажатии кнопки должен быть осуществлен переход на домашний экран (рис. 6).

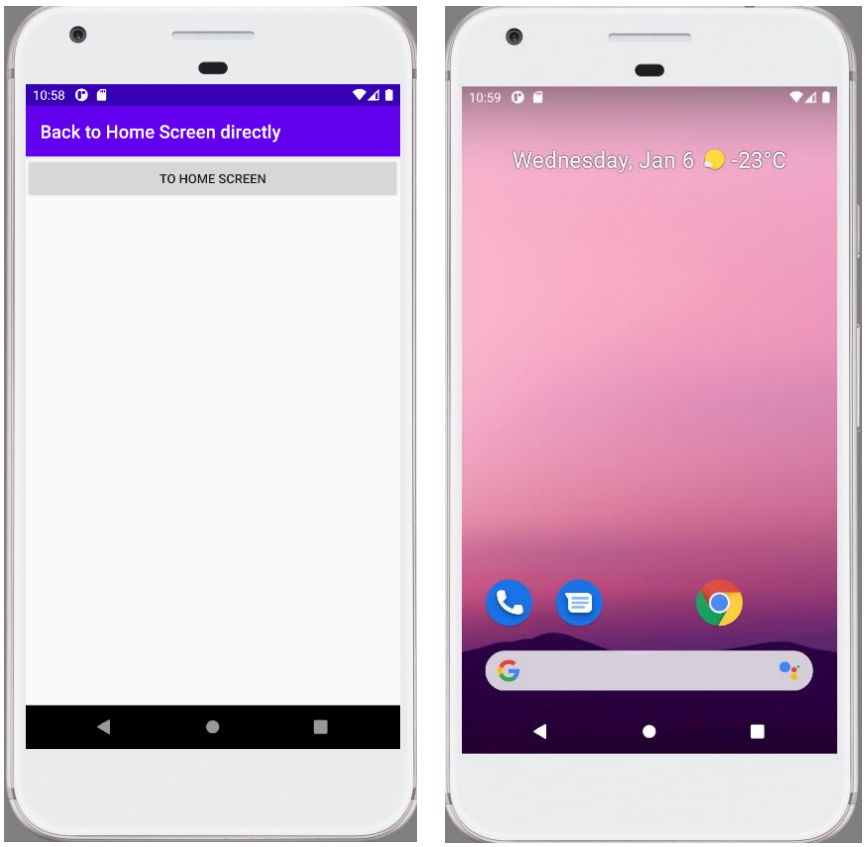


Рисунок 6 - Программный переход на домашний экран

2.5 Вызов стандартных Activity

Система Android и приложения, которые идут в комплекте с платформой, также используют объекты Intent для активизации определенных системой компонентов (например, различных приложений и служб при загрузке системы). Для вызова системных компонентов в классе Intent существует множество констант Standard Activity Actions, определяющих действия для запуска стандартных Activity. Например, действие `INTENT_ACTION_MUSIC_PLAYER` инициализирует обращение по телефону и запускает Activity, представляющий собой медиаплеер. Это окно можно вызвать следующим образом:

```
Intent intent = new Intent("INTENT_ACTION_MUSIC_PLAYER");

startActivity(intent);
```

Кроме класса Intent, в других классах также определены действия для вызова специализированных Activity. Например, в классах `MediaStore` и `MediaStrore.Audio.Media` определены константы действий для запуска окон с медиаплеером, поиска музыки и записи с микрофона. Также действия определены в классах `DownloadManager` для запуска Activity, управляющего

загрузкой файлов через Интернет, `RingtoneManager` для запуска окна, устанавливающего режим звонка, и т. д.

Упражнение 6. *Создать приложение IntentActionsMedia, в котором при нажатии по одной кнопке запускается видеокамера, по другой – окно медиаплеера*

1) создайте проект `IntentActionsMedia`
2) в файл компоновки главного окна поместите 2 кнопки `Video Camera` и `Music Player`
3) в классе MainActivity создайте объект Intent с параметрами для вызова соответствующих стандартных Activity:
   - ***INTENT_ACTION_VIDEO_CAMERA***
   - ***INTENT_ACTION_MEDIA_PLAY_FROM_SEARCH***
4) после компиляции приложения можно запускать из приложения два стандартных Activity: диктофон для записи звука и медиаплеер. Внешний вид приложения и открываемых окон показан на рис. 7.

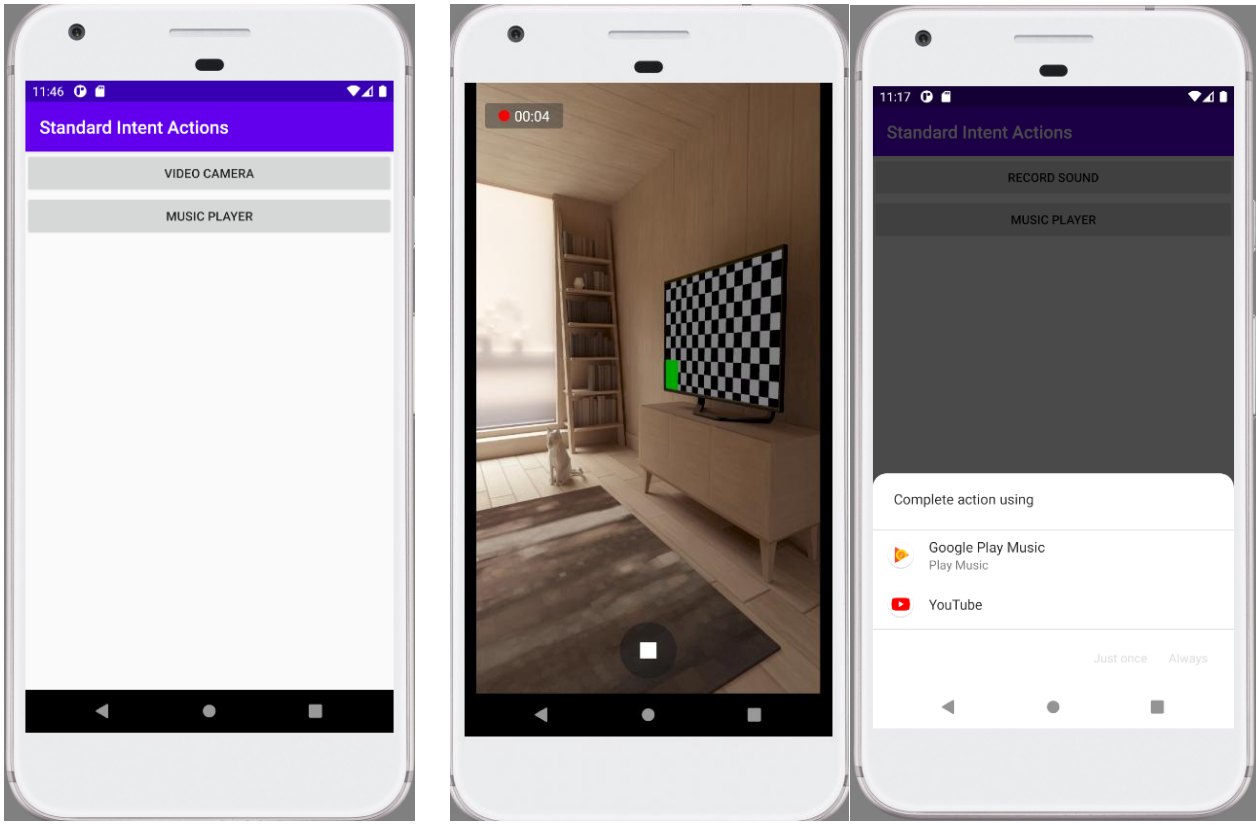


Рисунок 7 - Вызов стандартных Activity

Внимание!. Большинство стандартных Activity просто так запустить нельзя, т. к. при запуске они требуют наличие соответствующих разрешений

2.6 Вызов Activity с использованием разрешений

В системе Android, по умолчанию, доступ почти ко всем системным компонентам ограничен для внешних приложений. Поэтому, если приложению требуется использовать какую-либо системную

функциональность, например, необходимо послать сообщение SMS, подключиться к мобильному Интернету или к удаленному сетевому сервису, нужно задать для приложения соответствующие разрешения.

При проектировании приложения программисты довольно часто забывают подключить разрешения. Впрочем, это сразу обнаруживается при запуске приложения - при отсутствии нужного разрешения генерируется исключение

Для вызова системных компонентов в классе Intent определено множество констант *Standard Activity Actions*, определяющих действия для запуска стандартных Activity. Например, действие *ACTION_DIAL* инициализирует телефонный вызов и запускает Activity, представляющий собой окно для набора телефонного номера. Это окно можно вызвать следующим образом:

```
Intent intent = new Intent("Intent.ACTION_DIAL");

startActivity(intent);
```

Кроме класса Intent, в других классах тоже определены действия для вызова специализированных Activity. Также действия определены в классах *DownloadManager* для запуска Activity, управляющего загрузкой файлов через Интернет, *RingtoneManager* для запуска окна, устанавливающего режим звонка, и т. д.

Разрешение требуется только для действия *Intent.ACTION_DIAL*, для двух других действий разрешения устанавливать не нужно.

Упражнение 7. *Создать приложение, в котором при нажатии по кнопке запускается наборная панель для вызова абонента*

1) создайте проект `IntentActions`
2) объявите разрешение в манифесте приложения. Разрешение можно задать напрямую в коде файла манифеста приложения, добавив элемент с атрибутом, содержащим имя нужного разрешения
```
<uses-permission
    android:name="android.permission.CALL_PHONE">
</uses-permission>
```
3) в файл компоновки главного окна поместите кнопку для вызова окна набора телефонного номера с надписью `Phone Call`
4) в классе окна приложения в теле обработчика события onClick() создайте объект Intent с параметрами для вызова соответствующей стандартной Activity
5) после компиляции приложения можно запускать из приложения стандартный Activity с ограниченным доступом - наборную панель для телефонного вызова абонента. Внешний вид приложения и наборной панели показан на рис. 8.

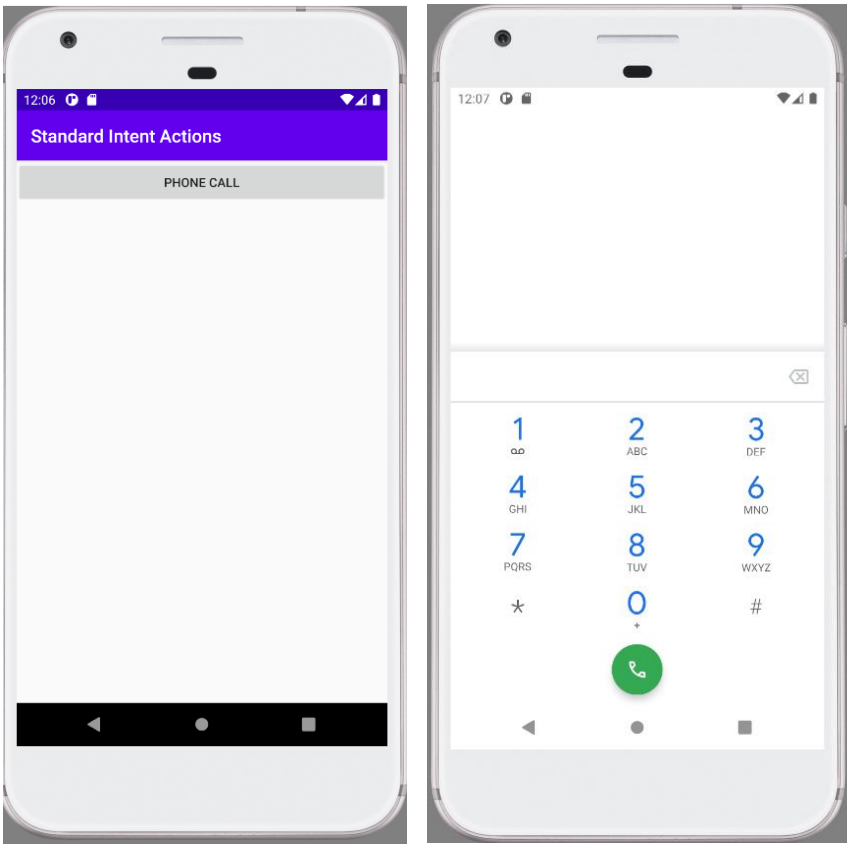


Рисунок 8 - Вызов стандартного Activity: наборной панели телефона с использованием разрешения