

Министерство науки и высшего образования Российской Федерации

Калужский филиал
федерального государственного бюджетного образовательного
учреждения высшего образования
**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»**
(КФ МГТУ им. Н.Э. Баумана)

Ю.С. Белов, С.С. Гришунов

**РАЗРАБОТКА ANDROID-ПРИЛОЖЕНИЙ С НЕСКОЛЬКИМИ
ACTIVITY. РАБОТА С INTENT-ОБЪЕКТАМИ**

Методические указания к выполнению лабораторной работы
по курсу «Разработка мобильного программного обеспечения»

Калуга – 2019

УДК 004.42
ББК 32.972.13
Б435

Методические указания составлены в соответствии с учебным планом КФ МГТУ им. Н.Э. Баумана по направлению подготовки 09.03.04 «Программная инженерия» кафедры «Программного обеспечения ЭВМ, информационных технологий».

Методические указания рассмотрены и одобрены:


- Кафедрой «Программного обеспечения ЭВМ, информационных технологий» (ИУ4-КФ) протокол № 51.4/9 от « 10 » апреля 2019 г.

Зав. кафедрой ИУ4-КФ

 к.т.н., доцент Ю.Е. Гагарин

- Методической комиссией факультета ИУ-КФ протокол № 11 от « 19 » апреля 2019 г.


Председатель методической
комиссии факультета ИУ-КФ

 к.т.н., доцент М.Ю. Адкин

- Методической комиссией

КФ МГТУ им.Н.Э. Баумана протокол № 7 от « 7 » 05 2019 г.

Председатель методической комиссии
КФ МГТУ им.Н.Э. Баумана

 д.э.н., профессор О.Л. Перерва

Рецензент:

к.т.н., доцент кафедры ИУ6-КФ

 А.Б. Лачихина

Авторы

к.ф.-м.н., доцент кафедры ИУ4-КФ
асс. кафедры ИУ4-КФ

 Ю.С. Белов
 С.С. Гришунов

Аннотация

Методические указания по выполнению лабораторной работы по курсу «Разработка мобильного программного обеспечения» содержат информацию о механизме создания многоэкранных приложений на платформе Android, руководство по разработке приложений с использованием явного и неявного вызова классов, особенности реализации Android-приложений с использованием Intent-объектов и задание на выполнение лабораторной работы.

Предназначены для студентов 3-го курса бакалавриата КФ МГТУ им. Н.Э. Баумана, обучающихся по направлению подготовки 09.03.04 «Программная инженерия».

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	4
ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ.....	5
МНОГОЭКРАННЫЕ ПРИЛОЖЕНИЯ.....	6
ПЕРЕКЛЮЧЕНИЕ МЕЖДУ ЭКРАНАМИ ПРИЛОЖЕНИЯ.....	10
INTENT ОБЪЕКТЫ.....	19
ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ	27
ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ.....	27
ВАРИАНТЫ ЗАДАНИЙ.....	27
КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ	31
ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ	31
ОСНОВНАЯ ЛИТЕРАТУРА.....	32
ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА	32

ВВЕДЕНИЕ

Настоящие методические указания составлены в соответствии с программой проведения лабораторных работ по курсу «Разработка мобильного программного обеспечения» на кафедре «Программное обеспечение ЭВМ, информационные технологии и прикладная математика» факультета фундаментальных наук Калужского филиала МГТУ им. Н.Э. Баумана.

Методические указания, ориентированные на студентов 3-го курса направления подготовки 09.03.04 «Программная инженерия», содержат информацию о механизме создания многоэкранных приложений на платформе Android, руководство по разработке приложений с использованием явного и неявного вызова классов, особенности реализации Android-приложений с использованием Intent-объектов и задание на выполнение лабораторной работы.

Для выполнения лабораторной работы студенту необходимы минимальные знания по программированию на высокоуровневом языке программирования Kotlin.

Требования к программному обеспечению:

- ОС Windows 7/8/10 / Ubuntu 16.04 (или старше)
- Android Studio

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ

Целью выполнения лабораторной работы является приобретение практических навыков разработки приложений с несколькими Activity и Intent-объектами.

Основными задачами выполнения лабораторной работы являются:

1. Изучить механизм создания многоэкранных приложений на платформе Android
2. Научиться разрабатывать приложения с использованием явного и неявного вызова классов
3. Понять особенности реализации Android-приложений с использованием Intent-объектов

Результатами работы являются:

- Приложение, иллюстрирующее навыки разработки многоэкранных приложений на платформе Android
- Подготовленный отчет

МНОГОЭКРАННЫЕ ПРИЛОЖЕНИЯ

Activity

Реальные приложения, как правило, имеют несколько окон — состоят из нескольких Activity, которыми надо уметь управлять и которые должны взаимодействовать между собой.

Как правило, один из Activity приложения (окно которого открывается при запуске приложения) помечен как главный. Также 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 разных процессах.

Процессы в системе Android

Когда хотя бы один из компонентов приложения (или все приложение) будет востребован, система Android запускает процесс, который содержит единственный основной поток для выполнения. По умолчанию все компоненты приложения работают в этом процессе и потоке.

Однако можно принять меры, чтобы компоненты работали в других процессах и порождали дополнительные потоки для любого процесса.

Все компоненты инициализируются в основном потоке процесса. Отдельные потоки для каждого экземпляра обычно не создаются. Следовательно, все методы обратного вызова, определенные в компоненте и вызываемые системой, всегда работают в основном потоке процесса. Это означает, что компонент не должен выполнять в методах обратного вызова длительные операции (например, загрузку файлов из сети или циклы вычисления) или блокировать системный вызов, т. к. это блокирует любые другие компоненты в этом процессе. Для таких операций порождают отдельные потоки.

Система Android может решить завершить процесс в случае нехватки памяти или если память востребована другими, более важными процессами. Прикладные компоненты, выполняющиеся в

этих процессах, будут уничтожены. Процесс будет перезапущен для компонентов в случае их повторного вызова.

Порядок, в котором процессы уничтожаются для освобождения ресурсов, определяется приоритетом (рисунок 1).

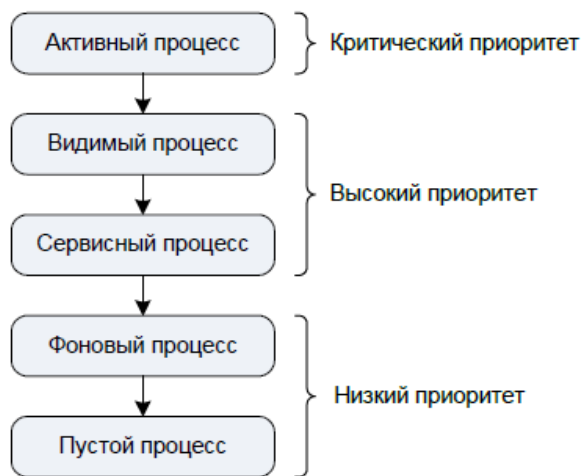


Рис. 1. Приоритет и статус процессов

Процессы с самой низкой важностью уничтожаются первыми. Есть пять уровней в иерархии важности.

1. **Активный процесс** (Foreground Process). Процесс считается активным, если выполняется любое из следующих условий:

- в процессе выполняется Activity, с которым взаимодействует пользователь;
- в процессе выполняется служба, связанная с Activity, с которым взаимодействует пользователь;
- процесс имеет объект Service, и выполняется один из методов обратного вызова, определенных в этом объекте;
- процесс имеет объект BroadcastReceiver, и выполняется его метод обратного вызова для приема Intent.

Одновременно могут существовать только несколько приоритетных процессов. Они будут уничтожены только в крайнем случае — если

памяти настолько мало, что они все вместе не в состоянии продолжать работу.

2. Видимый процесс (Visible Process) — компонент из этого процесса еще может вызываться пользователем. Это может быть процесс Activity, который не находится в фокусе, но все еще виден пользователю. Видимым может также быть процесс службы, которая в данный момент связана с Activity, находящимся на переднем плане (или частично закрытым другим Activity). Это может произойти, например, при вызове диалога, который не занимает весь экран, когда Activity потерял фокус, но виден пользователю и находится позади диалога. Видимый процесс считается важным и не будет уничтожен, пока остаются процессы с более низким приоритетом.

3. Сервисный процесс (Service Process) — процесс, в котором выполняется Service и который не относится ни к одной из двух предыдущих категорий. Хотя сервисные процессы обычно не привязаны к интерфейсу, видимому пользователем, они выполняют задания, нужные пользователю, например, фоновая работа медиа - плеера или загрузка данных из сети, так что система сохраняет их при наличии свободной памяти наряду со всеми активными и видимыми процессами.

4. Фоновый процесс (Background Process) — процесс, в котором выполняется Activity, который в настоящее время не виден пользователю. Эти процессы не имеют никакого прямого воздействия на пользовательский ввод и могут быть уничтожены в любое время, чтобы освободить память для активного, видимого или сервисного процесса. Обычно имеется много фоновых процессов, они сохраняются в списке LRU (least recently used, "не использующиеся дольше всех"), чтобы гарантировать, что находящийся в конце этого списка процесс, в котором выполняется Activity, был бы уничтожен в последнюю очередь.

5. Пустой процесс (Empty Process) — не содержит никаких активных компонентов приложения. Единственная причина сохранять такой процесс — только как кэш, чтобы уменьшить время запуска при вызове компонента. Система уничтожает эти процессы в первую очередь.

Состояния Activity

Когда пользователь работает с мобильным телефоном, он постоянно открывает, закрывает Activity или перемещает их на задний план. Activity может находиться в трех состояниях:

1. **Активный (active или running)** – Activity находится на переднем плане экрана мобильного устройства. Это Activity, который является центром для интерактивного взаимодействия с пользователем;

2. **Приостановленный (paused)** – Activity потерял фокус, но все еще видим пользователю. То есть другой Activity находится сверху и частично перекрывает данный Activity. Приостановленный Activity может быть уничтожен системой в критических ситуациях при нехватке памяти;

3. **Остановленный (stopped)** – если данный Activity полностью закрыт другим Activity. Он больше не видим пользователю и может быть уничтожен системой, если память необходима для другого, более важного процесса.

ПЕРЕКЛЮЧЕНИЕ МЕЖДУ ЭКРАНАМИ ПРИЛОЖЕНИЯ

Простое переключение на другой экран

[Как ранее говорилось](#), приложения не всегда состоят из одного экрана. Например, при создании приложения можно добавить кнопку «О программе», в которой указана информация о разработчике, версия и т.п. При нажатии на кнопку «О программе» открывается новый экран, где находится та информация. Можно воспринимать экран активности как веб-страницу с ссылкой на другую страницу. Если посмотреть на код в файле MainActivity.java, то можно увидеть, что класс MainActivity тоже относится к Activity (или его наследникам) или, если говорить точнее, наследуется от него.

```
class MainActivity : AppCompatActivity()
```

Далее следует создать новый класс, который может быть похож на MainActivity и затем переключиться на него при нажатии кнопки.

Рассмотрим пример. Создадим новый проект и поместим на главную активность кнопку. Далее создадим новую форму для отображения дополнительной информации. Создавать новую активность будем вручную.

Создадим новый XML-файл разметки activity_about.xml в папке res/layout. Щёлкните правой кнопкой мыши на папке layout и выберите из контекстного меню New | Layout resource file. Появится диалоговое окно. В первом поле вводим имя файла activity_about. Во втором нужно ввести корневой элемент. По умолчанию там стоит ConstraintLayout. Стираем текст и вводим ScrollView (рисунок 2).

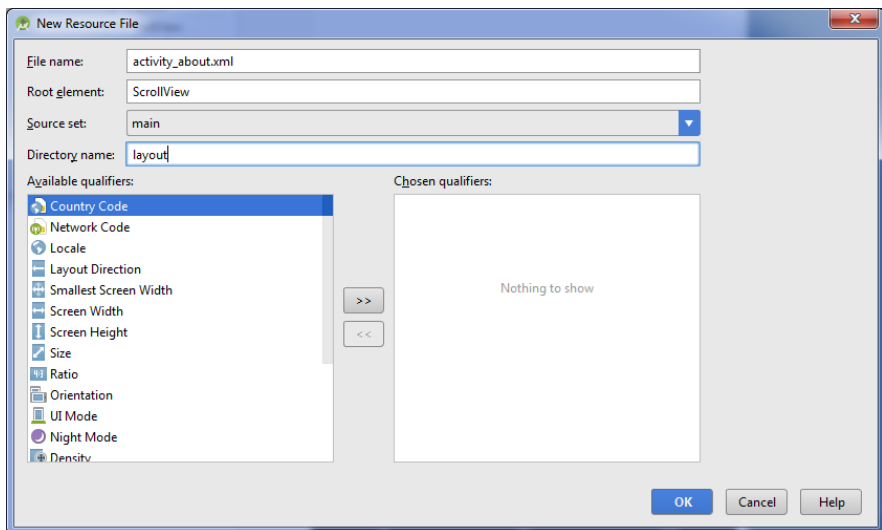


Рис. 2. Создание новой активности

Получится соответствующая заготовка, в которую вставим элемент `TextView`.

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/textView_about_content"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/about_text"/>
</ScrollView>
```

Информация будет извлекаться из ресурсов, а именно из строкового ресурса `about_text`. Откроем файл `res/values/strings.xml` и вводим текст. А также добавим строковый ресурс для заголовка нового экрана.

```
<string name="about_text">
```

Activity — это компонент приложения,\n

который выдает экран,\n

и с которым пользователи могут взаимодействовать,\n

для выполнения каких-либо действий.\n </string>

```
<string name="about_title">Новая активность</string>
```

Далее необходимо создать класс для окна AboutActivity.java. Выбираем в меню File | New | Java Class и заполняем нужные поля (рисунок 3).

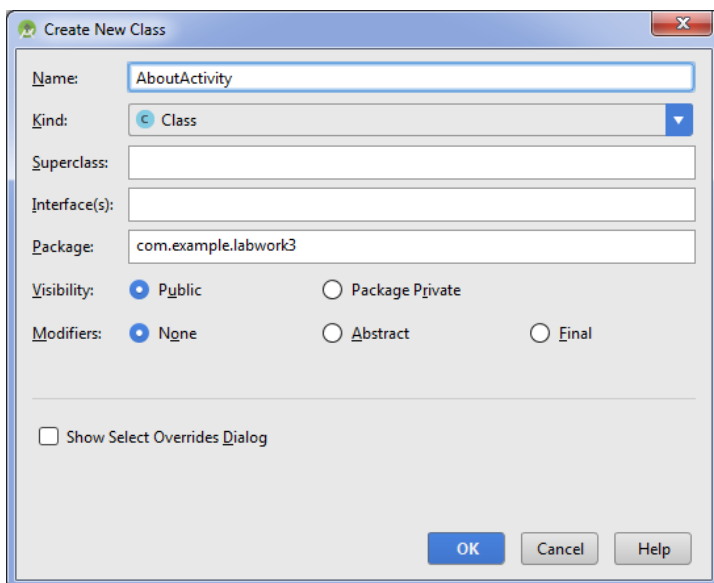


Рис. 3. Создание класса для новой активности

Сейчас класс практически пустой. Добавим код вручную. Класс должен наследоваться от абстрактного класса Activity или его родственников типа FragmentActivity, AppCompatActivity и т.д. Дописываем «: Activity()». У класса активности должен быть метод onCreate(). Ставим курсор мыши внутри класса и выбираем в меню Code | Override Methods (Ctrl+O). В диалоговом окне ищем нужный

класс, можно набирать на клавиатуре первые символы для быстрого поиска. В созданном методе нужно вызвать метод `setContentView()`, который подгрузит на экран подготовленную разметку.

```
class ResultsActivity : Activity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_results)  
    }  
}
```

Наша задача - перейти на новый экран при щелчку кнопки на первом экране. Переходим обратно к классу `MainActivity`. Напишем обработчик щелчка кнопки.

```
fun onClick(view: View) {  
    var intent = Intent(this@MainActivity, AboutActivity::class.java)  
    startActivity(intent)  
}
```

Для запуска нового экрана необходимо создать экземпляр класса `Intent` и указать в первом параметре текущий класс, а во втором - класс для перехода, у нас это `AboutActivity`. После этого вызывается метод `startActivity()`, который и запускает новый экран.

Далее необходимо зарегистрировать новый `Activity` в манифесте `AndroidManifest.xml`. Найдите этот файл в своем проекте и дважды щёлкните на нём. Откроется окно редактирования файла. Добавьте новый тег `<activity>` после закрывающего тега `</activity>` для первой активности.

```
<activity android:name=".AboutActivity"  
    android:label="@string/about_title">  
</activity>
```

Запускаем приложение (рисунок 4), щёлкаем на кнопке и получаем созданное нами новое окно (рисунок 5).

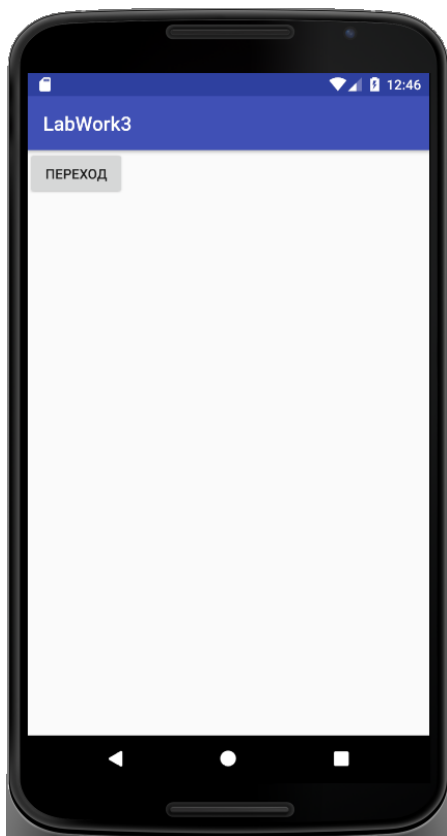


Рис. 4. Приложение после запуска

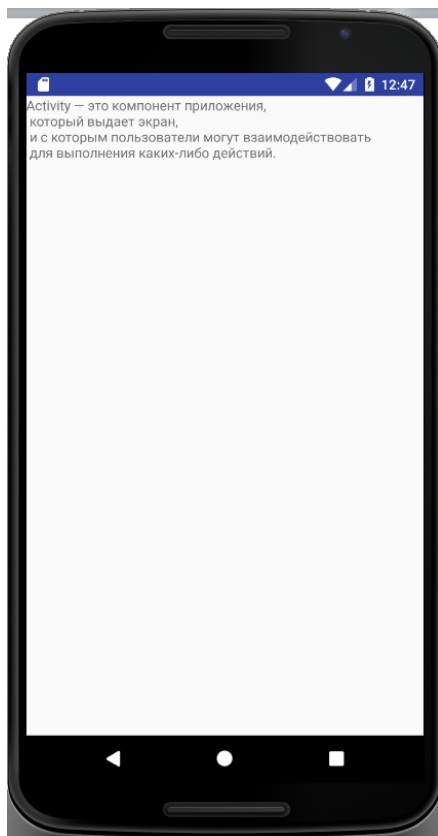


Рис. 5. Новая активность

После вызова метода `startActivity()` запустится новая активность (в данном случае `AboutActivity`), она станет видимой и переместится на вершину стека, содержащего работающие компоненты. При вызове метода `finish()` из новой активности (или при нажатии аппаратной клавиши возврата) она будет закрыта и удалена из стека. Разработчик также может перемещаться к предыдущей (или к любой другой) активности, используя всё тот же метод `startActivity()`.

Существует более быстрый способ создания новой активности. Для этого необходимо выбрать File | New | Activity | Basic Activity (или другой шаблон). В этом случае все манипуляции с созданием разметки, класса и нужных строчек в файле манифеста будут выполнены автоматически.

Передача данных между активностями

Мы использовали простейший пример для вызова другого экрана активности. Иногда требуется не только вызвать новый экран, но и передать в него данные. Например, имя пользователя. В этом случае нужно задействовать специальную область `extraData`, который имеется у класса `Intent`.

Область `extraData` - это список пар ключ/значение, который передаётся вместе с намерением. В качестве ключей используются строки, а для значений можно использовать любые примитивные типы данных, массивы примитивов, объекты класса `Bundle` и др.

Для передачи данных в другую активность используется метод `putExtra()`:

```
intent.putExtra("Ключ", "Значение")
```

Принимающая активность должна вызвать какой-нибудь подходящий метод: `getIntExtra()`, `getStringExtra()` и т.д.:

```
var count = intent.getIntExtra("name", 0)
```

Обратимся к предыдущему примеру. У первой активности разместим два текстовых поля и кнопку. Внешний вид может быть следующим:

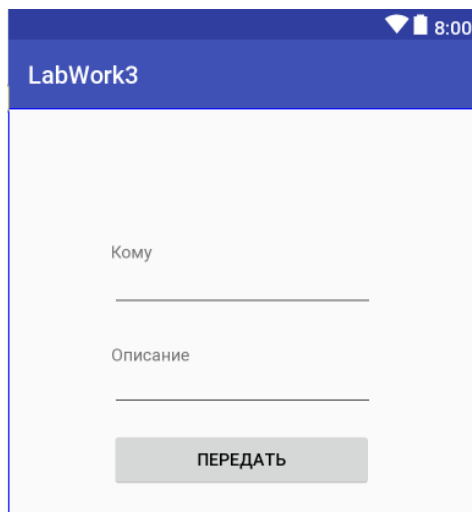


Рис. 6. Вид главной активности

Добавляем код у первой активности:

```
fun onClickv(view: View) {  
    val userEditText = findViewById<View>(R.id.editTextUser) as  
EditText  
    val giftEditText = findViewById<View>(R.id.editTextGift) as  
EditText  
  
    val intent = Intent(this@MainActivity, SecondActivity::class.java)  
  
    intent.putExtra("username", userEditText.text.toString())  
    intent.putExtra("gift", giftEditText.text.toString())  
    startActivity(intent)  
}
```

Мы поместили в специальный контейнер объекта Intent два [ключа со значениями](#), которые берутся из текстовых полей. Когда пользователь введёт данные в текстовые поля, они попадут в этот контейнер и будут переданы второй активности.

Вторая активность должна быть готова к приёму сообщений:


```
var user: String? = ""  
var gift: String? = ""
```

```
user = intent.extras!!.getString("username")  
gift = intent.extras!!.getString("gift")
```

```
val infoTextView = findViewById<View>(R.id.textViewInfo) as  
TextView
```

```
infoTextView.text = "$user , вам передали $gift"
```

```
var count = intent.getStringExtra("int")
```

При разработке собственных приложений желательно добавлять проверку при обработке данных. Возможны ситуации, когда вы запустите вторую активность с пустыми данными типа null.

В нашем случае мы знаем, что ждём строковое значение, поэтому код можно переписать так:

```
val intent = intent  
user = intent.getStringExtra("username")
```

Или так:

```
user = intent.getStringExtra("username")
```

Запустим программу и убедимся в успешной передаче данных (рисунки 7, 8).

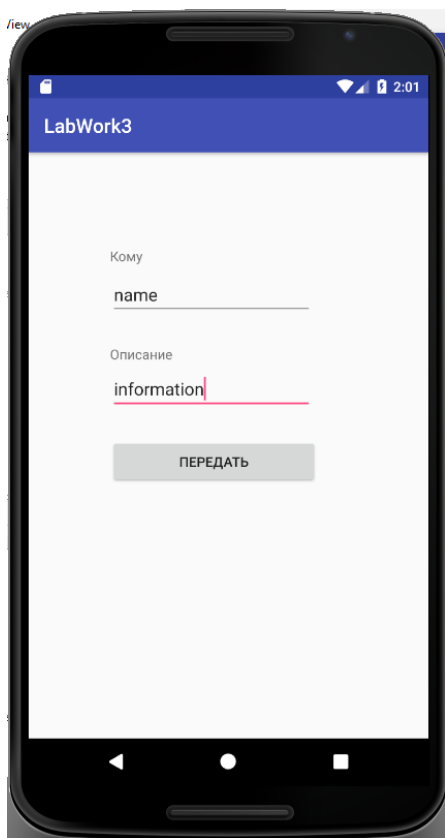


Рис. 7. Главная активность

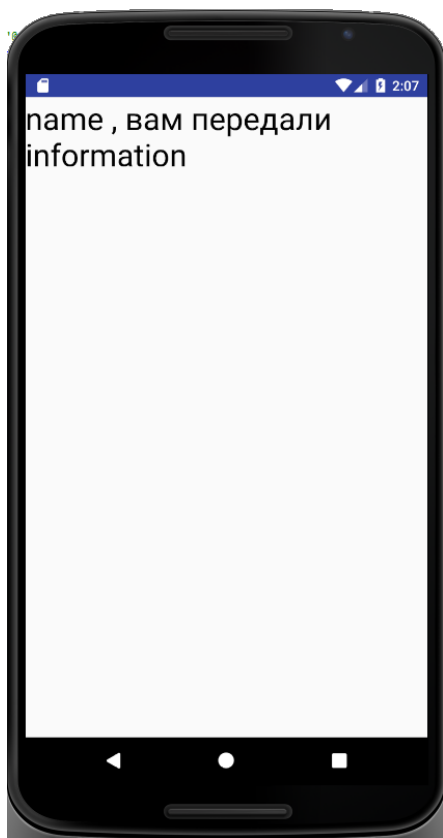


Рис. 8. Результат передачи информации

INTENT ОБЪЕКТЫ

Intent

Намерение (Intent) - это механизм для описания одной операции - выбрать фотографию, отправить письмо, сделать звонок, запустить браузер и перейти по указанному адресу. В Android-приложениях многие операции работают через намерения.

Наиболее распространенный сценарий использования намерения - запуск другой активности в своём приложении.

Но это не единственный вариант использования намерения. Также можно использовать для объявления о запуске активности или сервиса, направленных на выполнение каких-либо действий (как правило, речь о работе с определенной частью данных) или для передачи уведомлений о том, что произошло некое событие (или действие).

Намерения могут применяться для трансляции сообщений по системе. Любое приложение способно зарегистрировать широкоэвентный приёмник и отслеживать эти намерения с возможностью на них реагировать. Это позволяет создавать приложения, использующие событийную модель, в основе которой лежат внутренние, системные или сторонние события, передаваемые внешними программами.

Android транслирует намерения для объявления о системных событиях, например об изменениях в состоянии сетевого подключения или в уровне заряда батареи. Системные приложения в Android, такие как программы дозвола или управления SMS, регистрируют компоненты, отслеживающие заданные намерения, например, входящий звонок или получено новое SMS-сообщение, и соответствующим образом реагируют на них.

Явные намерения

Вызов Activity с помощью [Intent](#) рассмотренного в предыдущем примере – это явный вызов. Т.е. с помощью класса мы явно указываем какое Activity хотели бы увидеть. Это обычно используется внутри одного приложения. Схематично это можно изобразить так:

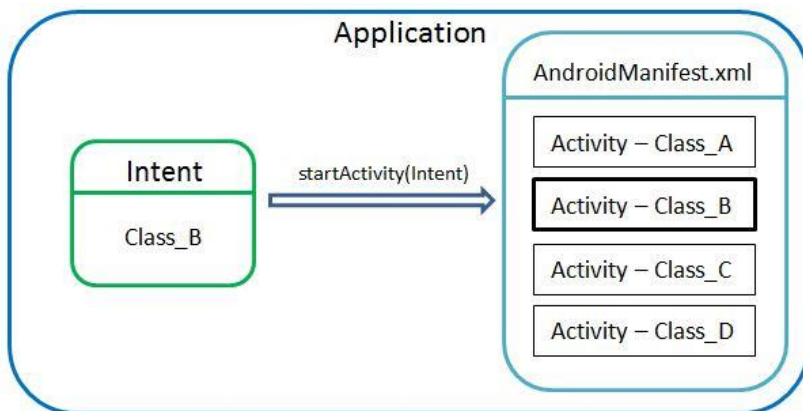


Рис. 9. Явный вызов

Здесь мы создаем Intent, в качестве параметра передаем ему класс Class_B. Далее вызываем метод startActivity с созданным Intent в качестве параметра. Метод проверяет AndroidManifest на наличие Activity связанной с классом Class_B и если находит, то отображает. Все это в пределах одного приложения.

Неявные намерения

Существует также неявный вызов активности. В этом случае при создании [намерения](#) мы используем не имя класса, а указываем параметры action, data, category с определёнными значениями. комбинация этих значений определяют цель, которую мы хотим достичь. Например: отправка письма, открытие гиперссылки, редактирование текста, просмотр картинки, звонок по определенному номеру и т.д. В свою очередь для Activity мы прописываем Intent Filter - это набор тех же параметров: action, data, category (но значения уже свои - зависят от того, что умеет делать Activity). И если параметры нашего 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 вам выбор, какой именно программой вы хотите воспользоваться. Схематично это можно изобразить так:

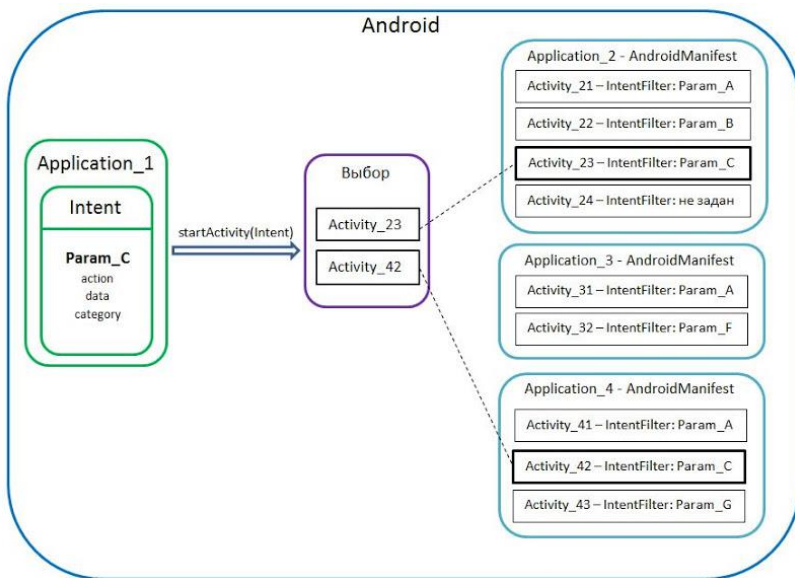


Рис. 10. Неявный вызов

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.

Как правило, объект Intent может содержать следующие поля:

1. **Имя компонента** – имя, который должен обработать намерение. Используется объект ComponentName, который является комбинацией полного имени класса целевого компонента (например, "MainActivity")

и набора имени пакета в файле манифеста приложения, где компонент постоянно находится (например, "com.samples.yourproject"). Составляющее имя является дополнительным. Если оно установлено, объект Intent поставляет образцу определяемого класса. Если имя не установлено, Android использует другую информацию в объекте Intent, чтобы определить местонахождение подходящего адресата. Составляющее имя устанавливается методами `setComponent()`, `setClass()` или `setClassName()` и читается методом `getComponent()`;

2. **Действие** – определяет действие, которое будет выполнено. Класс Intent содержит множество констант действия. Название метода определяет ряд параметров и возвращаемое значение. Вы можете также определить собственные действия для активизации активности. В этом случае вы должны включать имя пакета приложения в качестве префикса, например `com.samples.yourproject.CUSTOM_ACTION`. Действие в объекте Intent устанавливается в методе `setAction()` и читается методом `getAction()`;

3. **Данные** – это URI данных и тип MIME для этих данных. Разные активности соединены с разными видами спецификаций данных.

4. **Категория** – строка, содержащая дополнительную информацию о виде компонента, который должен обработать намерение. В объект Intent можно поместить любое количество описаний категорий. Класс Intent определяет несколько констант CATEGORY, например, CATEGORY_BROWSABLE

5. **Дополнения** – пары ключ-значения для дополнительной информации, которую нужно поставить компоненту, обращающемуся с намерением. Например, действие ACTION_TIMEZONE_CHANGED имеет дополнение time-zone, которое идентифицирует новый часовой пояс, ACTION_HEADSET_PLUG имеет дополнение state, указывающее, включены ли наушники или отключены, а также дополнение name для типа наушников. Объект Intent имеет ряд методов `put...()` для вставки различных типов дополнительных данных и подобного набора методов `get...()` для чтения данных. Дополнения устанавливаются и читаются как объекты Bundle с использованием методов `putExtras()` и `getExtras()`;

6. Флаги – указывают системе, как запускать активность (например, какому заданию должна принадлежать активность) и как обработать это после того, как активность запустили (например, принадлежит ли она списку недавних активностей). Все флаги определены в классе `Intent`.

Продemonстрируем работу неявного вызова. Преобразуем предыдущий пример. Определим намерение с предопределенным действием `ACTION_VIEW` для запуска браузера и перехода на нужный адрес:

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
}  
  
fun onClick(view: View) {  
    val address = Uri.parse("https://developer.android.com/index.html")  
    val openlinkIntent = Intent(Intent.ACTION_VIEW, address)  
    startActivity(openlinkIntent)  
}
```

В данном случае действие `ACTION_VIEW` означает просмотр чего-либо. Мы указали нужные данные (адрес), и происходит запуск новой активности (браузера). При этом исходная активность приостанавливается и переходит в фоновый режим. Когда пользователь нажимает на кнопку `Back`, то он возвращается к исходной активности. Обратите внимание, что мы нигде не указываем конкретную программу-браузер типа `Chrome`, `Opera` и т.п.

Теперь при нажатии на кнопку мы будем переходить на сайт разработки под систему `Android`, при этом для перехода на сайт будет доступно два приложения на выбор (рисунки 11, 12).

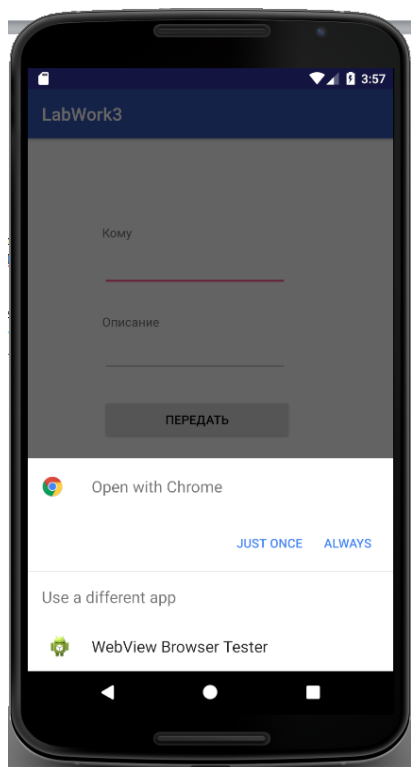


Рис. 11. Выбор Activity

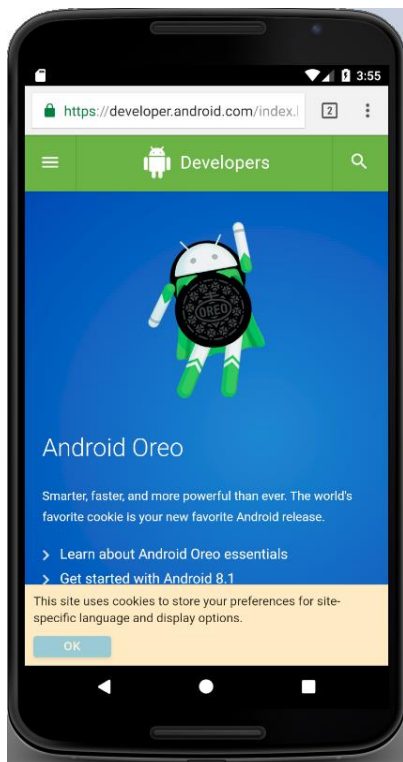


Рис. 12. Переход на сайт
с использованием неявного
вызова

Следует помнить, что нет никакой гарантии, что приложение, способное обработать ваше намерение, будет установлено и доступно на устройстве. Конечно, вероятность того, что у пользователя не будет приложений для звонков, достаточно низка. А если приложение установлено на планшете без телефонного модуля? Чтобы узнать, будет ли запущена активность для вашего намерения, можно отправить запрос Менеджеру пакетов при помощи метода `resolveActivity()`.

Константы действия

ACTION_ANSWER — открывает активность, которая связана с входящими звонками. Это действие обрабатывается стандартным экраном для приема звонков;

ACTION_CALL — инициализирует обращение по телефону;

ACTION_DELETE — запускает активность, с помощью которой можно удалить данные, указанные в пути URI внутри намерения;

ACTION_EDIT — отображает данные для редактирования пользователем;

ACTION_INSERT — открывает активность для вставки в Курсор (Cursor) нового элемента, указанного с помощью пути URI. Дочерняя активность, вызванная с этим действием, должна вернуть URI, ссылающийся на вставленный элемент;

ACTION_HEADSET_PLUG – подключение наушников;

ACTION_MAIN — запускается как начальная активность задания;

ACTION_PICK – загружает дочернюю Активность, позволяющую выбрать элемент из источника данных, указанный с помощью пути URI. При закрытии должен возвращаться URI, ссылающийся на выбранный элемент. Активность, которая будет запущена, зависит от типа выбранных данных, например при передаче пути `content://contacts/people` вызовется системный список контактов;

ACTION_SEARCH — запускает активность для выполнения поиска. Поисковый запрос хранится в виде строки в дополнительном параметре намерения по ключу `SearchManager.QUERY`;

ACTION_SEND — загружает экран для отправки данных, указанных в намерении. Контакт-получатель должен быть выбран с помощью полученной активности. Используйте метод `setType`, чтобы указать тип MIME для передаваемых данных. Эти данные должны храниться в параметре намерения `extras` с ключами `EXTRA_TEXT` или `EXTRA_STREAM`, в зависимости от типа. В случае с электронной почтой стандартное приложение в Android также принимает дополнительные параметры по ключам `EXTRA_EMAIL`, `EXTRA_CC`, `EXTRA_BCC` и `EXTRA_SUBJECT`. Используйте действие **ACTION_SEND** только в тех случаях, когда данные нужно передать удаленному адресату (а не другой программе на том же устройстве);

ACTION_SENDTO — открывает активность для отправки сообщений контакту, указанному в пути URI, который передаётся через намерение;

ACTION_SYNC — синхронизирует данные сервера с данными мобильного устройства;

ACTION_TIMEZONE_CHANGED – смена часового пояса;

ACTION_VIEW — наиболее распространенное общее действие. Для данных, передаваемых с помощью пути URI в намерении, ищется наиболее подходящий способ вывода. Выбор приложения зависит от схемы (протокола) данных. Стандартные адреса `http:` будут открываться в браузере, адреса `tel:` — в приложении для звонка, `geo:` — в программе Google Maps, а данные о контакте — отображаются в приложении для управления контактной информацией;

ACTION_WEB_SEARCH — открывает активность, которая ведет поиск в интернете, основываясь на тексте, переданном с помощью пути URI (как правило, при этом запускается браузер);

Константы категорий

CATEGORY_BROWSABLE — активность может быть безопасно вызвана браузером, чтобы отобразить ссылочные данные, например, изображение или почтовое сообщение;

CATEGORY_HOME — активность отображает Home Screen, первый экран, который пользователь видит после включения устройства и загрузки системы, или когда нажимает клавишу HOME;

CATEGORY_LAUNCHER — активность может быть начальной деятельностью задания из списка приложений в группе Application Launcher устройства

Методы

Для работы с категориями в классе Intent определена группа методов:

`addCategory()` — помещает категорию в объект Intent;

`removeCategory()` — удаляет категорию, которая была добавлена ранее;

`getCategories()` — получает набор всех категорий, находящихся в настоящее время в объекте Intent;

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Разработать приложения согласно заданию, указанному в варианте.

ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ

Задание состоит из двух проектов. Все проекты используют Intent объекты. В первом проекте все требуемые параметры устанавливаются в главном Activity и результат отображается на второстепенном Activity. Во втором проекте результат отображается на главном Activity, а для задания соответствующих параметров используется несколько второстепенных Activity. Один из проектов реализовать с использованием неявного вызова (action), второй проект реализовать с явным вызовом класса(-ов).

ВАРИАНТЫ ЗАДАНИЙ

Вариант 1

С использованием компонентов EditText определить RGB цвет текста, расположение текста по горизонтали на Layout определить с помощью RadioButton (слева, справа, по центру), расположение текста по вертикали (вверху, внизу, по центру) определить с использованием Button. С заданными характеристиками отобразить произвольный текст в компоненте TextView.

Вариант 2

С использованием компонента EditText задать размер текста, на основе предопределенных цветов (не менее 5 + один случайный) задать цвет текста с помощью компонентов RadioButton, в компоненте EditText задать сам текст. С заданными характеристиками отобразить пользовательский текст в компоненте TextView.

Вариант 3

С использованием компонентов RadioButton задать цвет для текста (не менее 5 цветов, один цвет случайный), в компоненте EditText задать сам текст, в компоненте EditText задать номер Layout, в котором будет выводиться текст. Предусмотреть анализ диапазона номеров Layout. Количество Layout не менее трех. В указанном Layout отобразить текст в компоненте TextView с заданными характеристиками.

Вариант 4

С использованием компонентов `RadioButton` задать цвет для текста (не менее 5 цветов, один цвет случайный), в компоненте `EditText` задать сам текст, в компонентах `ToggleButton` указать `Layout`, в котором будет выводиться текст. Количество `Layout` не менее трех. В указанном `Layout` отобразить текст в компоненте `TextView` с заданными характеристиками.

Вариант 5

С использованием `RadioButton` определить не менее 5 функций одного переменного с фиксированными параметрами, в компонентах `EditText` указать пределы интегрирования, в компоненте `TextView` отобразить результат. Интерфейс согласовать с преподавателем.

Вариант 6

Решить систему линейных уравнений двух переменных. Диапазон решений от -10 до 10. На второстепенной `Activity` отобразить графический результат. Интерфейс согласовать с преподавателем.

Вариант 7

С использованием компонентов `EditText` определить цвет текста в пространстве RGB, цвет фона текста задать с использованием `RadioButton` (не менее 5 цветов, один цвет случайный), в компоненте `EditText` задать размер текста. Отобразить произвольный текст с заданными параметрами в компоненте `TextView`.

Вариант 8

С использованием `RadioButton` определить один из четырех видов анимации, в компоненте `EditText` определить цвет в пространстве RGB, с использованием `Button` определить размер текста (не менее 6 размеров текста). Для заданных параметров отобразить анимацию для произвольного текста в компоненте `TextView`.

Вариант 9

С использованием RadioButton выбрать фигуру, с помощью EditText задать цвет в пространстве RGB для выбранной фигуры, с использованием Button определить место вывода фигуры (9 областей). Отобразить фигуру с заданными параметрами.

Вариант 10

На основе компонентов ImageButton создать палитру цветов 3x3 (один из компонентов задает произвольный цвет), в компоненте EditText задать размер текста, с использованием Button указать 9 положений вывода текста. Отобразить произвольный текст в компоненте TextView с заданными параметрами.

Вариант 11

С использованием компонентов ImageButton определить палитру цветов 3x2 (один из компонентов задает случайный цвет) для задания фонового цвета компонента TextView, с помощью RadioButton выбрать тип анимации (не менее 4 типов). Отобразить анимацию для произвольного текста в компоненте TextView с заданным цветом.

Вариант 12

С использованием компонентов Button задать выравнивание текста (слева, справа, по центру), компоненты ImageButton задают палитру 3x3 цветов (один цвет задается случайно) для цвета текста. Вывести текст с заданными параметрами в компоненте TextView со случайными размерами от 10 до 100.

Вариант 13

С использованием ImageButton определить палитру цветов 3x3 (один цвет задается случайно), с помощью компонентов RadioButton выбирается один из Layout для вывода текста, компонент CheckButton позволяет указать случайное выравнивание текста, иначе выравнивание текста по центру. Отобразить произвольный текст с заданными параметрами в выбранном Layout.

Вариант 14

С помощью компонентов RadioButton задать не менее 6 размеров текста, компоненты ImageButton определяют палитру цветов 4x4 (один цвет случайный) для задания цвета текста, Button определяют выравнивание текста по горизонтали (слева, справа, по центру). Отобразить текст в компоненте TextView с заданными параметрами.

Вариант 15

С использованием RadioButton задать четыре типа анимации, определить палитру 3x3 для задания цвета фона текста, с помощью Button задать 4 размера текста. Выполнить анимацию текста в компоненте TextView с заданными параметрами.

Вариант 16

С использованием компонента ImageButton определить палитру цветов 3x3 для задания цвета Layout, выбор Layout, для которого задается цвет определяется тремя RadioButton, компоненты ToggleButton определяют выбор Layout, для которого будет осуществляться вывод произвольного текста. Осуществить вывод произвольного текста и применить цвет к Layout.

Вариант 17

С использованием EditText задать цвет TextView с произвольным текстом, с помощью RadioButton выбрать один из 4 видов анимации, для Layout на основе палитры 3x3 из элементов ImageButton определить цвет. Выполнить анимацию для компонента TextView с заданным цветом текста и применить цвет к Layout.

Вариант 18

С использованием компонентов ImageButton определить цвет для фона текста, с помощью компонента ToggleButton указать наличие или отсутствие тени, компонент CheckButton указывает на случайное выравнивание или его отсутствие (по умолчанию выравнивание текста по центру), 6 компонентов Button определяют размер текста. Отобразить текст в компоненте TextView с заданными характеристиками.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Дайте определение понятию Activity.
2. Раскройте понятия процесс и поток в системе Android.
3. Опишите механизм определения порядка уничтожения процессов.
4. Перечислите существующие приоритеты процессов.
5. Перечислите состояния, в которых может находиться Activity.
6. Опишите реализацию переключения на другой экран в системе Android.
7. Опишите механизм передачи данных между несколькими Activity.
8. Раскройте понятие Намерение (Intent).
9. Перечислите существующие типы Intent.
10. Приведите примеры констант действий.
11. Приведите примеры констант категорий.
12. Перечислите методы, определенные в классе Intent для работы с категориями.

ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

На выполнение лабораторной работы отводится 2 занятия (4 академических часа: 3 часа на выполнение и сдачу лабораторной работы и 1 час на подготовку отчета).

Номер варианта студенту выдается преподавателем.

Отчет на защиту предоставляется в печатном виде.

Структура отчета (на отдельном листе(-ах)): титульный лист, формулировка задания (вариант), листинг (xml код разметки экрана, графическое представление, соответствующее разметке экрана, код программы и при необходимости наличие кода дополнительных классов), результаты выполнения работы (графические изображения примеров работы приложения), выводы).

ОСНОВНАЯ ЛИТЕРАТУРА

1. Семакова, А. Введение в разработку приложений для смартфонов на ОС Android / А. Семакова. - 2-е изд., испр. - М. : Национальный Открытый Университет «ИНТУИТ», 2016. - 103 с. : ил. ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=429181>
2. Введение в разработку приложений для ОС Android / Ю.В. Березовская, О.А. Юфрякова, В.Г. Вологодина и др. - 2-е изд., испр. - М. : Национальный Открытый Университет «ИНТУИТ», 2016. - 434 с. : ил. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=428937>
3. Разработка приложений для смартфонов на ОС Android / Е.А. Латухина, О.А. Юфрякова, Ю.В. Березовская, К.А. Носов. - 2-е изд., исправ. - М. : Национальный Открытый Университет «ИНТУИТ», 2016. - 252 с. : ил. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=428807>

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

4. Дэвид, Х. Разработка приложений Java EE 6 в NetBeans 7. [Электронный ресурс] — Электрон. дан. — М. : ДМК Пресс, 2013. — 330 с. — Режим доступа: <http://e.lanbook.com/book/58693> — Загл. с экрана.
5. Сильвен, Р. Android NDK. Разработка приложений под Android на C/C++. [Электронный ресурс] — Электрон. дан. — М. : ДМК Пресс, 2012. — 496 с. — Режим доступа: <http://e.lanbook.com/book/9126> — Загл. с экрана.
6. Ретабоуил, С. Android NDK: руководство для начинающих. [Электронный ресурс] — Электрон. дан. — М. : ДМК Пресс, 2016. — 518 с. — Режим доступа: <http://e.lanbook.com/book/82810> — Загл. с экрана.

7. Соколова, В.В. Разработка мобильных приложений: учебное пособие / В.В. Соколова ; Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Томский государственный университет», Министерство образования и науки Российской Федерации. - Томск: Издательство Томского политехнического университета, 2015. - 176 с.: ил., табл., схем. - Библиогр. в кн.. - ISBN 978-5-4387-0369-3; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=442808> (15.06.2017).
8. Баженова, И.Ю. Язык программирования Java / И.Ю. Баженова. - М.: Диалог-МИФИ, 2008. - 254 с. : табл., ил. - ISBN 5-86404-091-6 ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=54745> (15.06.2017).
9. Джошуа Блох Java. Эффективное программирование [Электронный ресурс]/ Джошуа Блох— Электрон. текстовые данные. — Саратов: Профобразование, 2017.— 310 с.— Режим доступа: <http://www.iprbookshop.ru/64057.html> .— ЭБС «IPRbooks»

Электронные ресурсы:

10. Научная электронная библиотека <http://eLIBRARY.RU>
11. Электронно-библиотечная система <http://e.lanbook.com>
12. Электронно-библиотечная система «Университетская библиотека онлайн» <http://biblioclub.ru>
13. Электронно-библиотечная система IPRBook <http://www.iprbookshop.ru>
14. Базовый сайт о системе Android - https://www.android.com/intl/ru_ru
15. Разработка приложения на базе Android - <https://developer.android.com/index.html>