ОГЛАВЛЕНИЕ

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ

Целью выполнения лабораторной работы является формирование практических навыков создания пользовательских виджетов.

Основными задачами выполнения лабораторной работы являются:

1. Научиться создавать пользовательские виджеты для рабочих экранов.
2. Научиться использовать конфигурационные окна для настройки виджетов.
3. Уметь понимать схемы взаимодействия виджета с другими элементами платформы Android.
4. Разработать эффективное приложение с учетом аппаратных ограничений мобильных устройств.
5. Научиться реализовывать логику работы приложения с учетом специфики платформы Android.

Результатами работы являются:

- Разработанный виджет согласно варианту задания
- Подготовленный отчет

ВИДЖЕТЫ

Виджет – это небольшая программа, или часть программы, которая располагается на рабочем столе устройства и предназначена для отображения информации, управления оборудованием устройства и при этом может запускать другую программу, частью которой он является.

Например, есть виджеты, с помощью которых можно отображать такие сведения, как загрузка процессора, состояние батареи, информацию о текущей погоде и прочем. Есть виджеты, с помощью которых можно быстро включить или выключить GPS, Wi-Fi, Bluetooth, динамики и управлять другим оборудованием Android устройства. Есть такие виджеты, как погодные информеры, которые отображают на экране информацию о текущей погоде и прогнозе погоды, и которые могут вызывать погодное приложение, частью которого они являются.

Для создания простейшего виджета необходимы три детали:

1. layout-файл – в нем мы формируется внешний вид виджета. Все аналогично layout-файлам для Activity и фрагментов, только набор доступных компонентов здесь ограничен.

2. xml-файл с метаданными – в нем задаются различные характеристики виджета (layout-файл, размер виджета, интервал его обновления).

3. класс, наследующий *AppWidgetProvider*

СОЗДАНИЕ ПРОСТОГО ВИДЖЕТА

При создании простого виджета Activity не понадобится, поэтому во время создания нового проекта необходимо выбрать пункт «Add No Activity» (рис. 1).

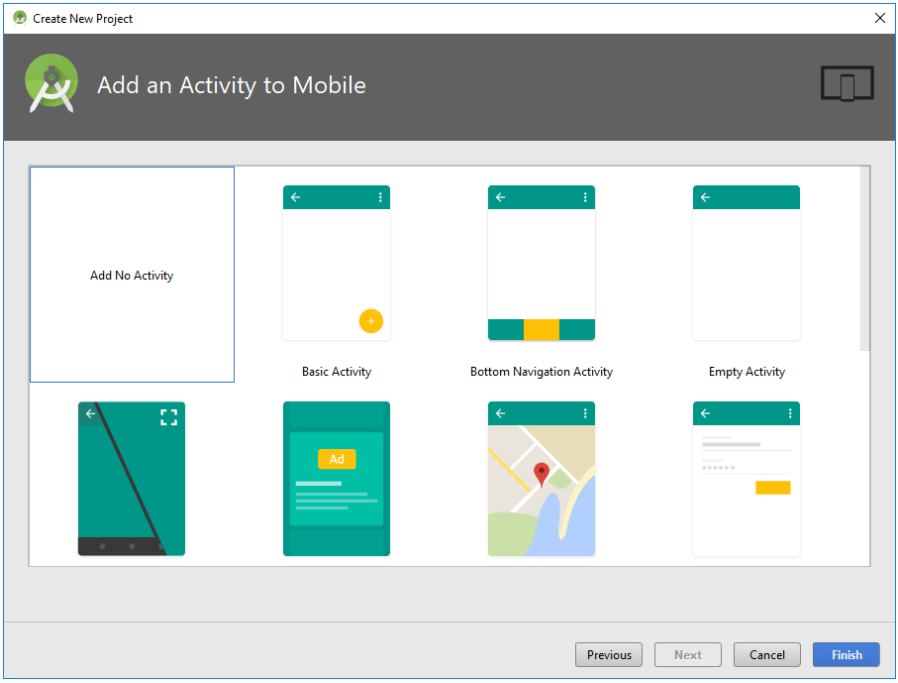

Рис. 1. Создание проекта

Добавим в файл strings.xml строки:

```
<resources>
    <string name="app_name">My
      Application</string>
    <string name="widget_name">My widget</string>
    <string name="widget_text">Hello,
     Android</string>
</resources>
```

Затем создаем layout-файл widget.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res
       /android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TextView
        android:id="@+id/tv"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#6600ff00"
        android:gravity="center"
        android:text="@string/widget_text"
        android:textColor="#000"
        android:textSize="18sp">
    </TextView>
</RelativeLayout>
```

Создаем файл метаданных res/xml/widget_metadata.xml:

```
<appwidget-provider
    xmlns:android="http://schemas.android.com/
       apk/res/android"
    android:initialLayout="@layout/widget"
    android:minHeight="40dp"
    android:minWidth="110dp"
    android:updatePeriodMillis="2400000">
</appwidget-provider>
```

initialLayout – указываем layout-файл для виджета.
minHeight, minWidth – размеры виджета.
updatePeriodMillis – интервал обновления в миллисекундах.

Создаем класс, наследующий AppWidgetProvider

```
package com.example.mywidget

import java.util.Arrays
import android.appwidget.AppWidgetManager
import android.appwidget.AppWidgetProvider
import android.content.Context
import android.util.Log

class MyWidget : AppWidgetProvider() {

    internal val LOG_TAG = "myLogs"

    override fun onEnabled(context: Context) {
        super.onEnabled(context)
        Log.d(LOG_TAG, "onEnabled")
    }

    override fun onUpdate(
        context: Context,
        appWidgetManager: AppWidgetManager,
        appWidgetIds: IntArray
    ) {
        super.onUpdate(
            context, appWidgetManager,
            appWidgetIds
        )
        Log.d(LOG_TAG, "onUpdate " +
Arrays.toString(appWidgetIds))
    }

    override fun onDeleted(
        context: Context,
        appWidgetIds: IntArray
    ) {
        super.onDeleted(context, appWidgetIds)
        Log.d(LOG_TAG, "onDeleted " +
Arrays.toString(appWidgetIds))
    }

    override fun onDisabled(context: Context) {
        super.onDisabled(context)
        Log.d(LOG_TAG, "onDisabled")
    }
}
```

onEnabled вызывается системой при создании первого экземпляра виджета

onUpdate вызывается при обновлении виджета. На вход, кроме контекста, метод получает объект AppWidgetManager и список ID экземпляров виджетов, которые обновляются. Именно этот метод обычно содержит код, который обновляет содержимое виджета.

onDeleted вызывается при удалении каждого экземпляра виджета. На вход, кроме контекста, метод получает список ID экземпляров виджетов, которые удаляются.

onDisabled вызывается при удалении последнего экземпляра виджета.

Изменим файл AndroidManifest.xml, добавив в него секцию receiver:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.mywidget">
    <application
        android:allowBackup="true"
        android:label="@string/app_name"
        android:icon="@mipmap/ic_launcher"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <receiver
            android:name="MyWidget"
            android:icon="@android:drawable/star_big_on"
            android:label="@string/widget_name">
            <intent-filter>
                <action>
                   android:name="android.appwidget.
                     action.APPWIDGET_UPDATE">
                </action>
            </intent-filter>
            <meta-data>
              android:name="android.appwidget.provider"
              android:resource="@xml/widget_metadata">
            </meta-data>
        </receiver>
    </application>
</manifest>
```

Виджет готов. Во время запуска может возникнуть ошибка “Default Activity not found”. Для ее устранения необходимо перейти в Run/Edit Configurations и в поле Launch выставить значение Nothing (рис. 2).

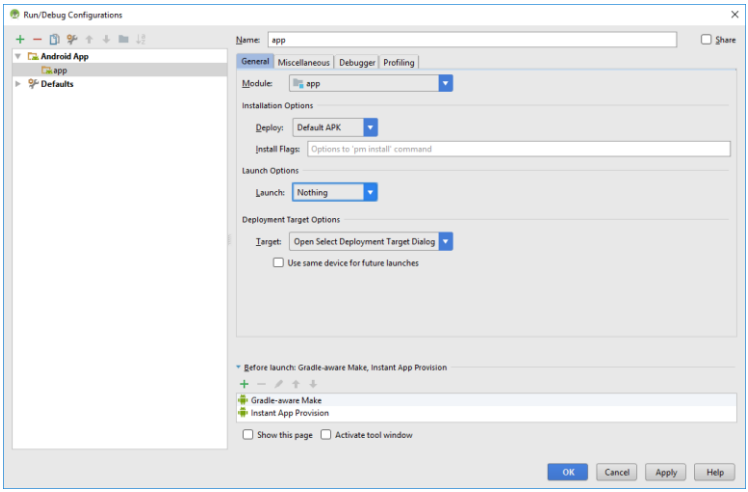

Рис. 2. Настройки запуска

Добавляем виджет на главный экран устройства (рис. 3):

Рис. 3. Добавление виджета

КОНФИГУРАЦИОННЫЙ ЭКРАН

Некоторые виджеты при размещении отображают конфигурационный экран, который позволяет настроить их. Например, у пользователя есть электронный счет на каком-либо сайте. И для этого сайта есть приложение-виджет. Чтобы виджет смог показать баланс нужного счета, он должен знать логин и пароль.

Для этих целей и существует конфигурационный экран (конфигурационное Activity). Он предложит пользователю поля для ввода и сохранит куда-либо (БД, Preferences, …) введенные данные, а при обновлении виджета эти данные будут считаны и использованы для отображения актуальной информации.

Для виджета из предыдущего примера добавим установку параметров цвета и текста, для этого создадим layout-файл config.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">
    <RadioGroup
        android:id="@+id/rgColor"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <RadioButton
            android:id="@+id/radioRed"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:checked="true"
            android:text="@string/red">
        </RadioButton>
        <RadioButton
            android:id="@+id/radioGreen"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/green">
        </RadioButton>
```

```
        <RadioButton
            android:id="@+id/radioBlue"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/blue">
        </RadioButton>
    </RadioGroup>
    <EditText
        android:id="@+id/etText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10">
        <requestFocus>
        </requestFocus>
    </EditText>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="@string/ok">
    </Button>
</LinearLayout>
```

Создаем класс Activity ConfigActivity.java:

```
import android.R
import android.app.Activity
import android.appwidget.AppWidgetManager
import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.content.SharedPreferences.Editor
import android.graphics.Color
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.EditText
import android.widget.RadioGroup

class ConfigActivity : Activity() {

    internal var widgetID =
AppWidgetManager.INVALID_APPWIDGET_ID
```

```
    internal lateinit var resultValue: Intent

    internal val LOG_TAG = "myLogs"

    override fun onCreate(savedInstanceState: Bundle?)
{
        super.onCreate(savedInstanceState)
        Log.d(LOG_TAG, "onCreate config")

        // извлекаем ID конфигурируемого виджета
        val intent = intent
        val extras = intent.extras
        if (extras != null) {
            widgetID = extras.getInt(
                AppWidgetManager.EXTRA_APPWIDGET_ID,
                AppWidgetManager.INVALID_APPWIDGET_ID
            )
        }
        // и проверяем его корректность
        if (widgetID ==
AppWidgetManager.INVALID_APPWIDGET_ID) {
            finish()
        }

        // формируем intent ответа
        resultValue = Intent()
        resultValue.putExtra(
            AppWidgetManager.EXTRA_APPWIDGET_ID,
            widgetID
        )

        // отрицательный ответ
        setResult(Activity.RESULT_CANCELED,
resultValue)
        setContentView(R.layout.config)
    }

    fun onClick(v: View) {
        val selRBColor =
(findViewById<View>(R.id.rgColor) as RadioGroup)
            .checkedRadioButtonId
        var color = Color.RED
        when (selRBColor) {
            R.id.radioRed -> color =
Color.parseColor("#66ff0000")
```

```
            R.id.radioGreen -> color =
Color.parseColor("#6600ff00")
            R.id.radioBlue -> color =
Color.parseColor("#660000ff")
        }
        val etText = findViewById<View>(R.id.etText) as
EditText

        // Записываем значения с экрана в Preferences
        val sp = getSharedPreferences(WIDGET_PREF,
Context.MODE_PRIVATE)
        val editor = sp.edit()
        editor.putString(
            WIDGET_TEXT + widgetID,
            etText.text
                .toString()
        )
        editor.putInt(WIDGET_COLOR + widgetID, color)
        editor.commit()
        val appWidgetManager =
AppWidgetManager.getInstance(this)
        MyWidget.updateWidget(this, appWidgetManager,
sp, widgetID)

        // положительный ответ
        setResult(Activity.RESULT_OK, resultValue)
        Log.d(LOG_TAG, "finish config $widgetID")
        finish()
    }

    companion object {

        val WIDGET_PREF = "widget_pref"
        val WIDGET_TEXT = "widget_text_"
        val WIDGET_COLOR = "widget_color_"
    }
}
```

Конфигурационное Activity при вызове получает Intent, в котором содержится ID создаваемого экземпляра виджета. При закрытии оно должно формировать результат методом setResult и в этом ответе передавать Intent с ID экземпляра.

В onClick мы читаем выбранный цвет и введенный в поле текст и пишем эти значения в Preferences. В имени записываемого параметра мы используем ID, чтобы можно было отличать параметры разных экземпляров друг от друга. Далее мы говорим системе, что результат работы положительный, и виджет можно создавать. Закрываем Activity.

Добавим Activity в манифест и настроим ему фильтр с action = android.appwidget.action.APPWIDGET_CONFIGURE.

Также необходимо добавить в файл метаданных (xml/widget_metadata.xml) параметр android:configure и указать в нем полный путь к классу Activity:

```
android:configure="ru.startandroid.develop.
  p1181customwidget.ConfigActivity"
```

Далее изменяем файл MyWidget.java:

```
import java.util.Arrays

import android.R
import android.appwidget.AppWidgetManager
import android.appwidget.AppWidgetProvider
import android.content.Context
import android.content.SharedPreferences
import android.content.SharedPreferences.Editor
import android.util.Log
import android.widget.RemoteViews

class MyWidget : AppWidgetProvider() {

    override fun onEnabled(context: Context) {
        super.onEnabled(context)
        Log.d(LOG_TAG, "onEnabled")
    }

    override fun onUpdate(
        context: Context,
        appWidgetManager: AppWidgetManager,
        appWidgetIds: IntArray
    ) {
        super.onUpdate(
```

```
            context, appWidgetManager,
            appWidgetIds
        )
        Log.d(LOG_TAG, "onUpdate " +
Arrays.toString(appWidgetIds))
        val sp = context.getSharedPreferences(
            ConfigActivity.WIDGET_PREF,
Context.MODE_PRIVATE
        )
        for (id in appWidgetIds) {
            updateWidget(context, appWidgetManager, sp,
id)
        }
    }

    override fun onDeleted(
        context: Context,
        appWidgetIds: IntArray
    ) {
        super.onDeleted(context, appWidgetIds)
        Log.d(LOG_TAG, "onDeleted " +
Arrays.toString(appWidgetIds))

        // Удаляем Preferences
        val editor = context.getSharedPreferences(
            ConfigActivity.WIDGET_PREF,
            Context.MODE_PRIVATE
        ).edit()
        for (widgetID in appWidgetIds) {
            editor.remove(ConfigActivity.WIDGET_TEXT +
widgetID)
            editor.remove(ConfigActivity.WIDGET_COLOR +
widgetID)
        }
        editor.commit()
    }

    override fun onDisabled(context: Context) {
        super.onDisabled(context)
        Log.d(LOG_TAG, "onDisabled")
    }

    companion object {

        internal val LOG_TAG = "myLogs"
```

```
        internal fun updateWidget(
            context: Context, appWidgetManager:
AppWidgetManager,
            sp: SharedPreferences, widgetID: Int
        ) {
            Log.d(LOG_TAG, "updateWidget $widgetID")

            // Читаем параметры Preferences
            val widgetText =
sp.getString(ConfigActivity.WIDGET_TEXT + widgetID,
null) ?: return
            val widgetColor =
sp.getInt(ConfigActivity.WIDGET_COLOR + widgetID, 0)

            // Настраиваем внешний вид виджета
            val widgetView = RemoteViews(
                context.packageName,
                R.layout.widget
            )
            widgetView.setTextViewText(R.id.tv,
widgetText)
            widgetView.setInt(
                R.id.tv, "setBackgroundColor",
                widgetColor
            )

            // Обновляем виджет
            appWidgetManager.updateAppWidget(widgetID,
widgetView)
        }
    }
}
```

В onUpdate перебираются все ID экземпляров, которые необходимо обновить, и для каждого из них вызывается метод обновления.

onDeleted вызывается, когда виджет удаляется с экрана. Если виджет удален, то нужно удалить и все настройки для него из Preferences.

Метод updateWidget обновляет конкретный экземпляр виджета, получая на вход его ID. Здесь читаются настройки, которые записало конфигурационное Activity для этого экземпляра виджета. Эти

параметры необходимо применить к view-компонентам виджета. Но прямого доступа к view-компонентам виджета нет, и нельзя вызывать методы типа setText и setBackgroundColor напрямую. Поэтому используется класс RemoteViews, он предназначен для межпроцессной работы с view.

Выполнив все предыдущие шаги, запускаем приложение и добавляем новый виджет. Перед добавлением каждого экземпляра должен появляться экран для указания цвета и текста виджета.

ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ

Программа должна быть реализована на языке высокого уровня Kotlin.

ВАРИАНТЫ ЗАДАНИЙ

1. Создать виджет со списком. Каждый элемент списка должен содержать Имя исполнителя и название композиции. (Используйте Content Provider). Предусмотреть возможность обновления виджета. Если медиа файлы были изменены или удалены, послед обновления виджета список должен измениться.
2. Создать виджет со списком. Каждый элемент списка должен содержать Название Изображения и размер файла в Мб. (Используйте Content Provider). Предусмотреть возможность обновления виджета. Если файлы изображений были изменены (название файла) или удалены, после обновления виджета список должен измениться.
3. Создать виджет со списком. Каждый элемент списка должен содержать название композиции. (Используйте Content Provider). При первичном нажатии на элемент списка вместо названия композиции должно отобразится имя исполнителя. При вторичном нажатии на тот же элемент списка должно вновь отобразится название композиции.
4. Создать виджет отображающий случайные изображения галереи, имеющиеся на устройстве. Время отображения задается программно. Предусмотреть принудительное обновление виджета. При каждом принудительном обновлении должна меняться фотография.
5. Создать виджет отображающий случайный номер телефона и имя контакта, из имеющейся на устройстве телефонной книги. Предусмотреть принудительное обновление виджета. При каждом принудительном обновлении должен меняться номер телефона и имя контакта. По согласованию с преподавателем

предусмотреть возможность отправки смс посредством функционального элемента виджета на случайный номер телефона, текст смс может быть стандартным или введен вручную.

6. Создать виджет со списком. Каждый элемент списка должен отображать название доступного датчика. При первичном нажатии на элемент списка рядом с датчиком должно отобразится его текущее значение. При вторичном нажатии на тот же элемент списка должно вновь отобразится название датчика.
7. Создать виджет отображающий текущую дату: День недели и месяц должны отображаться на русском языке. По нажатию на виджет должно открываться окно конфигурации для настройки цвета фона и шрифта. Цвет выбирается по клику на ImageButton.
8. Создать виджет отображающий текущую дату: День недели и месяц должен отображаться на русском языке. По нажатию на виджет должно открываться окно конфигурации для настройки цвета фона и шрифта. Цвет и шрифт выбирается с помощью элемента Spinner (минимальное количество вариантов для каждого параметра не менее 5).
9. Создать виджет – счетчик с текстовым полем и двумя кнопками. Текстовое поле отображает числа. При нажатии на первую кнопку значение текстового поля увеличиваются, а при нажатии на вторую кнопку, значения текстового поля уменьшаются.
10. Создать виджет, отображающий знак зодиака и его краткое описание по дате рождения. В окне конфигурации вводится дата рождения, а в элементы интерфейса виджета выводится знак зодиака и краткое описание.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Дайте определение термину «виджет».
2. Опишите процесс создания виджета.
3. Опишите назначение конфигурационного окна.
4. Опишите класс RemoteViews.
5. Опишите роль обработчика событий onEnabled.
6. Опишите роль обработчика событий onUpdate.
7. Опишите роль обработчика событий onDelete.
8. Опишите роль обработчика событий onDisable.
9. Опишите процесс добавления виджета на рабочий стол.

ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

На выполнение лабораторной работы отводится 1 занятия (2 академических часа: 1 час 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. Вологдина 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