ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

***«*САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

Институт компьютерных наук и технологий

**Отчет о прохождении учебной практики**

|  |
| --- |
| Рубша Анастасия Игоревна |

*(Ф.И.О. обучающегося)*

|  |
| --- |
| 2 курс, гр. 3530901/80003 |

|  |
| --- |
| 09.03.01 «Информатика и вычислительная техника» |

*(Направление подготовки (код и наименование)*

|  |
| --- |
| **Место прохождения практики**: Высшая школа информационных систем и суперкомпьютерных технологий (ВШИСиСТ) ИКНТ ФГАОУ ВО «СПбПУ» с использованием электронного обучения и дистанционных образовательных технологий. |

|  |
| --- |
| **Сроки практики:** с 22 июня по 18 июля 2020 г. |

|  |
| --- |
| **Руководитель практики:** |

|  |
| --- |
| Жвариков В. А., к.т.н., доцент ВШИСиСТ ИКНТ |

*(Ф.И.О., уч. степень, должность)*

|  |
| --- |
| **Оценка (зачет): \_\_\_\_\_\_\_\_\_\_\_** |

|  |
| --- |
|  |

|  |
| --- |
| Руководитель практики : Жвариков В. А. |

|  |
| --- |
| Обучающийся: Рубша А. И. |

|  |
| --- |
| Дата: |

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

***«*САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

Институт компьютерных наук и технологий

**ИНДИВИДУАЛЬНЫЙ ПЛАН (ЗАДАНИЕ И ГРАФИК)**

**ПРОВЕДЕНИЯ ПРАКТИКИ**

|  |
| --- |
| Ф.И.О. обучающегося Рубша Анастасия Игоревна |

|  |  |
| --- | --- |
| **Направление подготовки** (код/наименование): 09.03.01, «Информатика и вычислительная техника» | |
| **Профиль** (код/наименование): 09.03.01 | |
| **Вид практики:** учебная | |
| **Тип практики:** ознакомительная | |
| **Место прохождения практики**: Высшая школа информационных систем и суперкомпьютерных технологий (ВШИСиСТ) ИКНТ ФГАОУ ВО «СПбПУ» с использованием электронного обучения и дистанционных образовательных технологий. | |
| |  | | --- | |  | | Руководитель практики: Жвариков В. А., к.т.н., доцент ВШИСиСТ ИКНТ | | *(Ф.И.О., уч.степень, должность)* | |

**Рабочий график проведения учебной практики**

Сроки практики: с **22.06.2020** г. по **18.07.2020** г.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Этапы (периоды) практики | Вид работ | Сроки прохождения этапа (периода) практики |
| 1 | Организационный этап | Установочная лекция (вебинар) для разъяснения целей, задач, содержания и порядка прохождения практики, выдача сопроводительных документов по практике  Основная тема: написание приложения (судоку) для Android на языке Java | 22.06.2020 г. |
| 2 | Основной  этап | Изучение среды разработки Android Studio, а также сторонних библиотек и баз данных.  Сбор информации, обработка, систематизация и анализ фактического и теоретического материала.  Содержание практики: разработка мобильного приложения (судоку) согласно заданному ТЗ.  Планируемые результаты прохождения практики: получение навыков создания мобильных программ под Android и полностью готовое мобильное приложение. | 23.06.2020 - 15.07.2020 г. |
| 3 | Заключительный этап | Подготовка отчета | 16.07.- 17.07.2020 г. |
| Защита отчета по практике (сдача зачета) | 17.07 - 18.07.2020 г |

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Рубша А. И.

Руководитель практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Жвариков В. А.

Оглавление

[Техническое задание 4](file:///C:\Users\Настя\Desktop\NikiforovPraktikaOtchet.docx#_Toc45274919)

[Метод решения 4](file:///C:\Users\Настя\Desktop\NikiforovPraktikaOtchet.docx#_Toc45274920)

[Скриншоты работы приложения 7](file:///C:\Users\Настя\Desktop\NikiforovPraktikaOtchet.docx#_Toc45274921)

[Список](file:///C:\Users\Настя\Desktop\NikiforovPraktikaOtchet.docx#_Toc45274922) использованных источников………………………………………………………………………………10

[Приложение 10](file:///C:\Users\Настя\Desktop\NikiforovPraktikaOtchet.docx#_Toc45274923)

**Техническое задание**

**Цель работы:** освоить создание Android приложения, получить навыки работы с программой Android Studio.

Разработать на основе полученных навыков игру Судоку на языке Java.

**Правила игры судоку:**

Судоку играется на одном игровом поле, состоящем из 9 на 9 клеток, всего 81 клетка. Внутри игрового поля находятся 9 "квадратов" (состоящих из 3 x 3 клеток). Каждая горизонтальная строка, вертикальный столбец и квадрат (9 клеток каждый) должны заполняться цифрами 1-9, не повторяя никаких чисел в строке, столбце или квадрате.

Это логическая игра, поэтому не нужно гадать. Если вы не знаете, какое число поставить в определенную клетку, продолжайте сканировать другие области игрового поля, пока не увидите возможность вставить нужное число. Но не пытайтесь "форсировать" что-либо - Судоку вознаграждает за терпение, понимание и решение различных комбинаций, а не за слепое везение или угадывание.

GitHub-репозиторий: https://github.com/Valkiriya228/sudokuAndroid

# **Метод решения**

На главном меню есть кнопка Start, при нажатии на которую пользователю будет предоставлена возможность выбрать один из вариантов полей. (В моем случае Easy1 и Easy2). Также в главном меню при нажатии на кнопку About пользователь сможет ознакомиться с правилами игры Судоку.

При выборе одного из двух полей, пользователю открывается игровое поле, где он сможет увидеть поле 9 на 9 клеток. Некоторые клетки заполнены цифрами, но некоторые клетки есть пустые. По нажатию на пустую ячейку пользователю открывается маленькое окошко с возможностью выбора одной из девяти цифр. По нажатию на какую-либо из цифр, пользователь видит, что он ввел в выделенную ячейку данную цифру.

Рассмотрим методы, приведенные в классе **MainActivity**, отвечающем за отображение главного окна данной игры и открытие других окон через нажатие на кнопки, располагающиеся на данном активити.

1)findViewsById() – инициализирует названия кнопок, через поиск по их id в xml файле

2)onClick() – производит обработку нажатия на 3 кнопки (Start, About, Exit)

3)windowChoseLevel() – открытие нового окна с возможностью выбора уровня сложности

4)startGame() – метод, открывающий игровое поле с выбранным пользователем уровнем сложности

5)onCreate() – данный метод отображает главное окошко игры – т.е activity\_main

Рассмотрим методы, приведенные в классе **aboutActivity**, отвечающем за отображение окна с правилами игры для пользователя. Класс построен на about.xml файле.

1)onCreate() – единственный и главный метод, отображающий пользователю окно с правилами игры, основанный на about.xml

Рассмотрим методы, приведенные в классе **puzzleView**, отвечающем за главную отрисовку поля для судоку 9 на 9 клеток,а также возможность выделить клетку и вставить туда нужную цифру.

1)onSizeChanged() – инициализация размеров клетки и отрисовка пустоко прямоугольника с помощью метода getRect()

2)getRect() – создание пустого прямоугольника с заранее заданными координатами

3) onDraw() – пожалуй, главный метод в данном классе. Метод, рисующий линии отделяющие столбцы от строк, а также возможность выделения ячейки и отрисовка цифр в какой-либо ячейке.

4)select() – вычисление новой области выделения по заданным новым координатам

5)onTouchEvent() – метод, позволяющий пользователю осуществлять взаимодействие с полем судоку, то есть при касании пальца к экрану, программа задает координаты касания.

6)setSelectedTile() – метод изменения числа в ячейке

Рассмотрим методы, приведенные в классе **GameActivity**, отвечающем за внутреннюю логику программы (введение значений, преображение строки в массив, массив использованных значений, которые надо скрыть от пользователя для упрощения задачи)

1)onCreate() – метод, отображающий поле нарисованное в классе PuzzleView, со всеми возможностями (выделить клетку и т д)

2)gettingThePuzzle() – определение уровня сложности самого судоку и преобразование его в вид массива с помощью 3 метода (смотри ниже)

3)getAnMassivOfSudoku() – преображение строки значений судоку в их массив

4)getTileString() – по данным координатам преобразование значения клетки в формат String

5) setTileIfValid()- введение клетки при его неправильном значении

6) showKeypadOrError() – при использовании всех значений в клетках по горизонтали вертикали и в квадрате, появление окна “No moves”, что означает что ничего вставлять не нужно

7)usedCells() – вычисление массива использованных значений в клетках

8)gettingCellValue() – метод возвращающий значение клетки по ее координатам

9)setCellValue() – присваивание определенной клетке значения digit

Рассмотрим методы, приведенные в классе **keypad**, отвечающем за работу с полем при нажатии на ячейку, и выборе вставления числа в ячейку

1)onCreate() – метод отображения окна keypad.xml с возможностью некоторых скрытных кнопок.

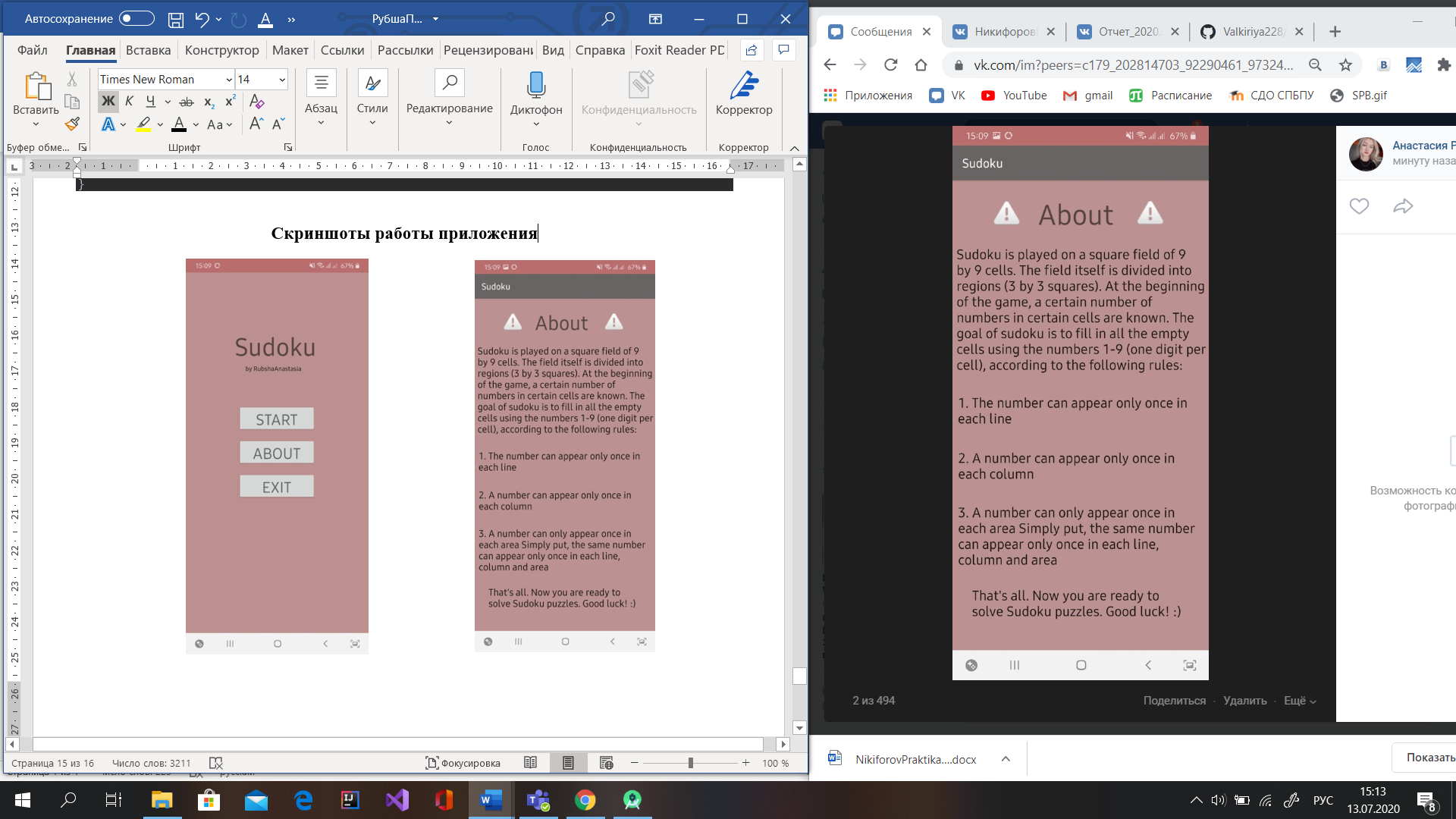
2)onKeyDown() – метод нажатия на все кнопки на данном поле

3)returnResult() – возврат выбранной пользователем цифры

4)isValid() – метод проверяющий подходит ли данная цифра для этой ячейки

5)findViewsById() – присваивание каждой кнопке и окну его id

6)setListeners() – установка обработчиков для каждой из кнопки

**Скриншоты работы приложения**

**Рис. 1-2 Отображение главного меню приложения и по нажатию на ABOUT – отображение поля с правилами игры.**

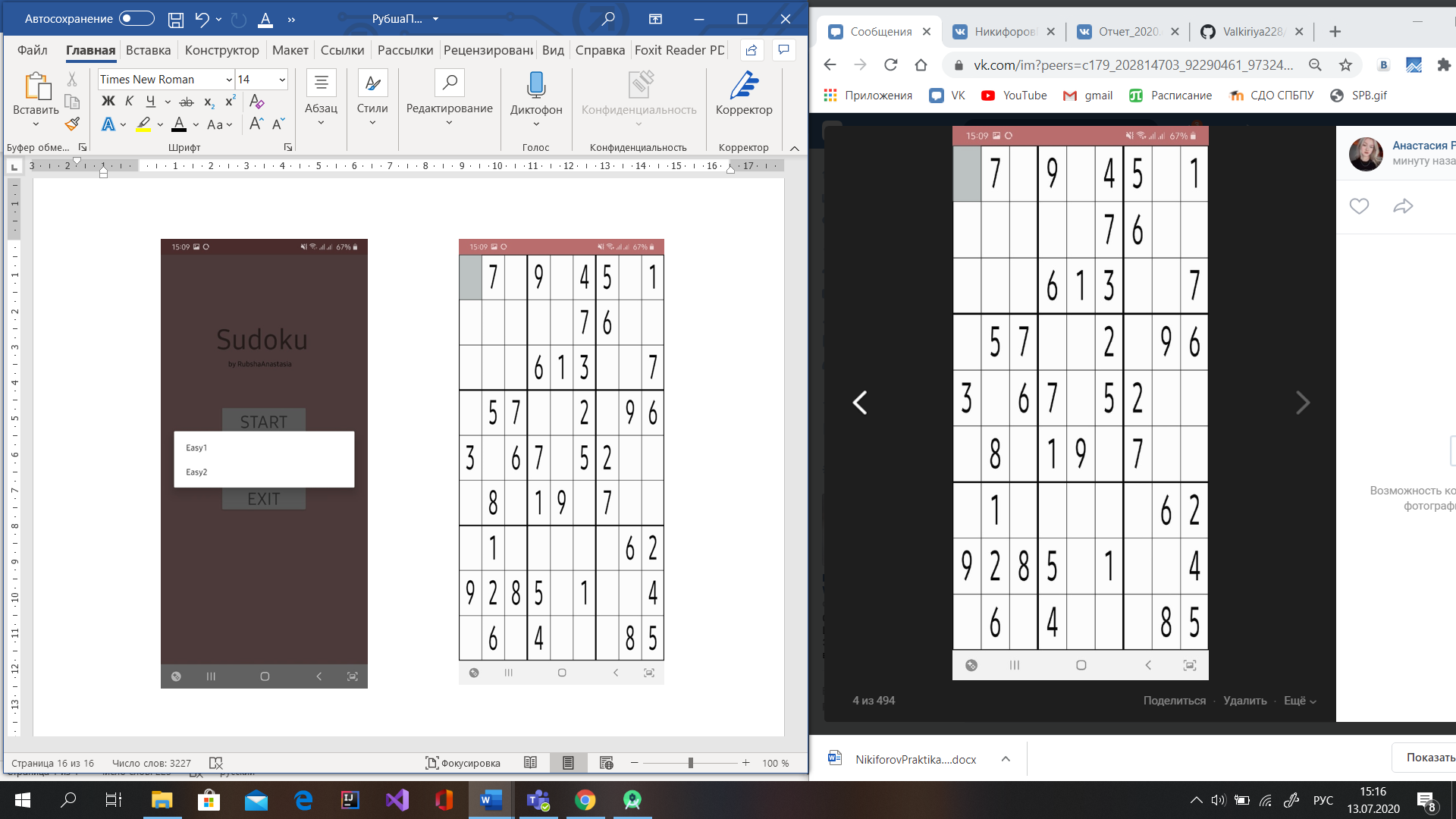
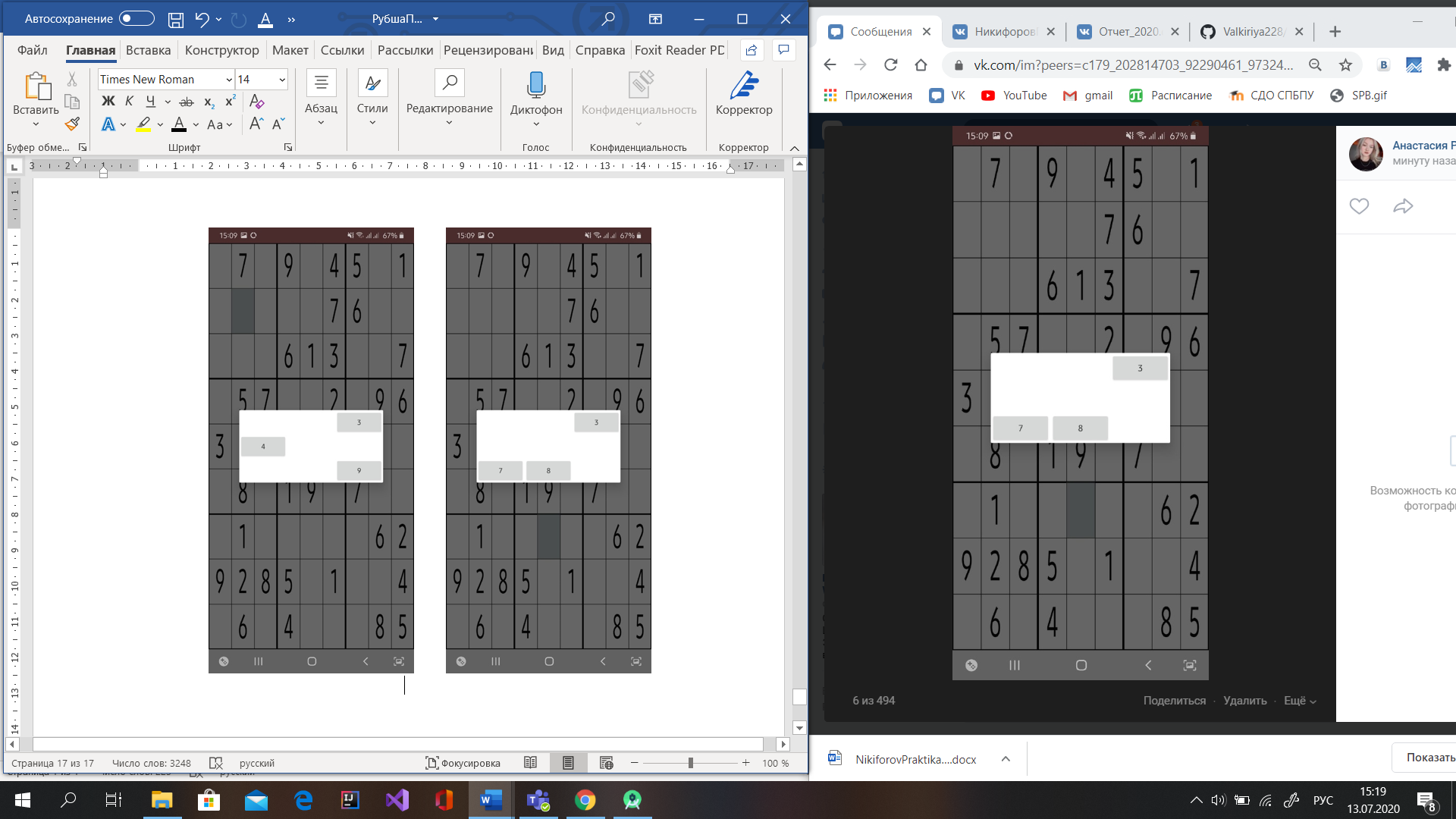


Рис. 3 – при нажатии на Start отображение выбора какого-либо из полей

Рис.4 – при выборе поля, собственно открытие данного поля (чистое судоку)

Рис 5-6 отображение поля с возможностью выбора цифр при выделении какой то определенной ячейки

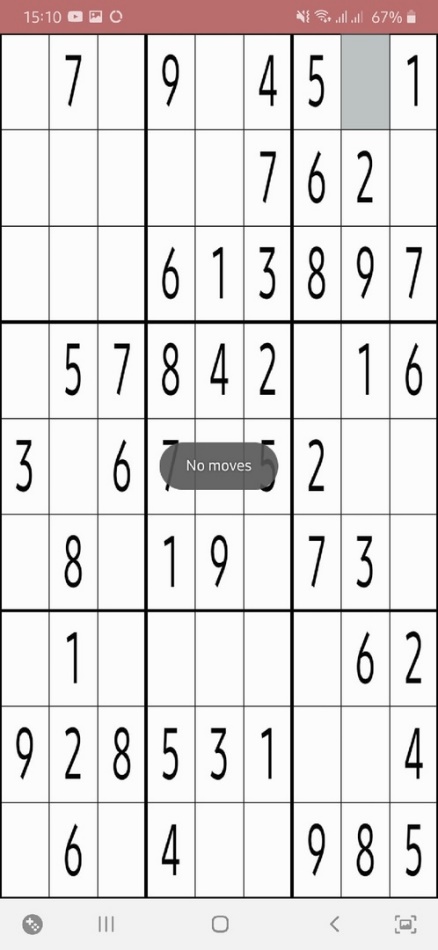


Рис 7 – при неправильных вводах цифр, наконец появляется окно No Moves – значит вы где-то допустили ошибку

**Список использованных источников.**

1. https://habr.com/ru/post/99323/
2. http://developer.alexanderklimov.ru/android/games/sudoku.php
3. http://developer.alexanderklimov.ru/android/catshop/graphics.php

**Приложение**

**1)MainActivity**

package com.example.sudoku.view;  
  
import android.app.Activity;  
import android.app.AlertDialog;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.view.View.OnClickListener;  
  
import com.example.sudoku.R;  
  
public class MainActivity extends Activity implements OnClickListener {  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*); //открытие первого главного окна с кнопками Start About Exit  
 findViewsById(); //инициализация всех кнопок и присваивание им их Id отдельным методом  
  
 }  
  
 public void findViewsById(){  
 View buttonStart = findViewById(R.id.*buttonStart*); //соответствие кнопкам их Id  
 buttonStart.setOnClickListener(this);  
 View buttonAbout = findViewById(R.id.*buttonAbout*);  
 buttonAbout.setOnClickListener(this);  
 View buttonExit = findViewById(R.id.*buttonExit*);  
 buttonExit.setOnClickListener(this);  
 }  
  
 public void onClick(View v) { // метод для нажатий на кнопки на главном окне  
 if (v.getId() == R.id.*buttonAbout*) {//  
 Intent about = new Intent(this, aboutActivity.class); //осуществляется переход в класс aboutActivity  
 startActivity(about); //отображение окна about.xml  
 } else if (v.getId() == R.id.*buttonStart*) {  
 windowChoseLevel(); // открытие окошка с возможностью выбрать уровень сложности пользователю (смотри метод ниже)  
 } else if (v.getId() == R.id.*buttonExit*) { //при нажатии на Exit осуществляется выход из приложения  
 finish(); //закрытие приложения  
 }  
 }  
 private void windowChoseLevel() { //открытие нового окна с возможностю выбора уровня сложности  
 new AlertDialog.Builder(this) //создание нового диалогового окна  
 .setItems(R.array.*difficulty*, new DialogInterface.OnClickListener() { //это диалоговое окно включает в себя 2 вида  
 // уровня сложности, с возможностью нажатия, то бишь выбора  
 public void onClick(DialogInterface dialoginterface, int i) {  
 startGame(i); //запуск метода, позволяющегооткрыть игровое поле  
 }  
 }).show();//отображение после его запуска  
 }  
  
 private void startGame(int i) { //метод, который открывает игровое поле, с выбранным уровнем сложности  
 Intent intent = new Intent(MainActivity.this, GameActivity.class); //создание окна с отсылкой к классу GameActivity,  
 // где прописывается логика  
 intent.putExtra(GameActivity.*level*, i);//инициализация сложности поля  
 startActivity(intent);//старт (открытие) окна содержимого  
 }  
  
  
  
}

**2)aboutActivity**

package com.example.sudoku.view;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.pm.ActivityInfo;  
import android.os.Bundle;  
  
import com.example.sudoku.R;  
  
public class aboutActivity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) { //открытие окна с правилами игры, при нажатии на кнопку About.  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*about*);  
 setRequestedOrientation(ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT*);  
 }  
}

**3)GameActivity**

package com.example.sudoku.view;  
  
  
import android.app.Activity;  
import android.app.Dialog;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.Gravity;  
import android.widget.Toast;  
  
import com.example.sudoku.R;  
import com.example.sudoku.keypad;  
  
  
public class GameActivity extends Activity {  
  
  
 public static final String *level* = "";  
 public static final int *easy1* = 0; //значение инта для первого поля  
 public static final int *easy2* = 1; //значение инта для второго поля  
 public static int *row* = 9; //количество строк  
 public static int *column* = 9; // количество строк  
 private int[] puzzle = new int[*row* \* *column*]; //массив с размером 9 на 9  
 private puzzleView puzzleView;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 int sudokuLevel = getIntent().getIntExtra(*level*,*easy2*);  
 puzzle = gettingThePuzzle(sudokuLevel); //присваивание puzzle в виде массива уровень который был выбран в методе gettingThePuzzle  
 usedCells();  
 puzzleView = new puzzleView(this);  
 setContentView(puzzleView);  
 puzzleView.requestFocus();  
 }  
  
  
 private int[] gettingThePuzzle(int sudokuLevel) { //определение уровня сложности и самого судоку и преобразование его в вид массива  
 String puzzle;  
 String easyFirst = "070904501000007600000613007" + //первое судоку легкого уровня  
 "057002096306705200080190700" +  
 "010000062928501004060400085";  
 String easySecond = "794080032021400590305000004" + //второе судоку легкого уровня  
 "139065020000243000040701000" +  
 "453000800067050309008300050";  
 if (sudokuLevel == *easy1*) { puzzle = easyFirst; //при выборе первого уровня сложности, присваиваем puzzle первую строку  
 } else { puzzle = easySecond;}//иначе присваиваем puzzle вторую строку  
 return *getAnMassivOfSudoku*(puzzle); //преобразование в массив  
 }  
  
  
 static protected int[] getAnMassivOfSudoku(String string) { //преображение строку значений судоку в массив  
 int[] puzzle = new int[string.length()]; //инициализация массива с размером равным размеру массивов easyFirst и easySecond  
 for (int i = 0; i < puzzle.length; i++)  
 puzzle[i] = string.charAt(i) - '0'; //если встречается "0", оно превращается в "", то бишь в пустую клетку  
 return puzzle; //возвращение чисел в судоку в виде массива  
 }  
  
 protected String getTileString(int x, int y) { //по данным координатам преобразовываем значение клетки в формат String  
 int v = gettingCellValue(x, y); //получение по координатам значения в данной клетке  
 if (v == 0) return ""; //если значение равно 0, присваиваем пустое значение клетке  
 else return String.*valueOf*(v); //иначе преобразовываем значение в формат String  
 }  
  
  
 protected boolean setTileIfValid(int x, int y, int value) { //введение значения если неправильное значение  
 int[] tiles = getUsedTiles(x, y); //массив использованных клеток по координате  
 if (value != 0) {  
 for (int tile : tiles) {  
 if (tile == value)  
 return false; //возврат false если введено значение которое уже есть в массиве использованных значений  
 }  
 }  
 setCellValue(x, y, value); // иначе присваивание клетке значение value  
 usedCells();//вычисление массива использованных клеток  
 return true;  
 }  
  
 protected void showKeypadOrError(int x, int y) {  
 int[] cells = getUsedTiles(x, y);  
 if (cells.length == 9) { //если использованы все значения в клетках, то появляется окно, что вставлять ничего не нужно  
 Toast toast = Toast.*makeText*(this,  
 R.string.*no\_moves\_label*, Toast.*LENGTH\_SHORT*);  
 toast.setGravity(Gravity.*CENTER*, 0, 0); //расположение окошка по середине  
 toast.show(); //показ данного окна пользователю  
 } else {  
 Dialog v = new keypad(this, cells, puzzleView); //иначе открытие окна с возможностью ввести число в клетку  
 v.show(); //показ данного окна пользователю  
 }  
 }  
  
 protected int[] getUsedTiles(int x, int y) { //возврат используемых клеток судя по координатам  
 return used[x][y];  
 }  
  
 private final int[][][] used = new int[9][9][]; //трехмерный массив с использованными клетками  
  
 private void usedCells() { //вычисление массива использованных клеток  
 for (int x = 0; x < *row*; x++) {  
 int y = 0;  
 while (y < *column*) {  
 used[x][y] = usedCells(x, y);  
 y++;  
 }  
 }  
 }  
  
  
 private int[] usedCells(int x, int y) {  
 int[] massive = new int[*row*]; //создаем пустой массив в котором будут значения которые уже используются  
 for (int i = 0; i < *row*; i++) { //проход по строкам судоку  
 if (i == y) continue;  
 int t = gettingCellValue(x, i); // присваивание значению t значение клетки по координатам x, i  
 if (t != 0) massive[t - 1] = t; //если это значение не равно 0, то добавляем это значение в массив, с индексом "значение - 1"  
 }  
 for (int i = 0; i < *column*; i++) { //проход по колоннам судоку  
 if (i == x) continue;  
 int t = gettingCellValue(i, y); // присваивание значению t значение клетки по координатам i, y  
 if (t != 0) massive[t - 1] = t; //если это значение не равно 0, то добавляем это значение в массив, с индексом "значение - 1"  
 }  
 int startx = (x / 3) \* 3; // стартовая клетка в каждой тройке каждой строки  
 int starty = (y / 3) \* 3; //стартовая клетка в каждой тройке каждой колонны  
 for (int i = startx; i < startx + 3; i++) { //проходим массивом в данном квадрате 3 на 3  
 for (int j = starty; j < starty + 3; j++) {  
 if (i == x && j == y) continue;  
 int t = gettingCellValue(i, j); // присваивание значению t значение клетки по координатам i, j  
 if (t != 0) massive[t - 1] = t; //если клетка не равна 0 то присваиваем массиву с индексом "значение - 1" значение t  
 }  
 }  
 int array = 0;  
 for (int t : massive) {  
 if (t != 0) array++; //если элемент в массиве не равен 0, инкрементируем array  
 }  
 int[] massiv1 = new int[array]; //создание нового массивас размером, равным полученному array  
 array = 0; //обнуляем данную переменную  
 for (int t : massive) {  
 if (t != 0) massiv1[array++] = t;  
 }  
 return massiv1;  
 }  
  
 private int gettingCellValue(int x, int y) { //метод возвращающий значение клетки по ее координатам  
 return puzzle[y \* 9 + x];  
 }  
  
 private void setCellValue(int x, int y, int digit) { //присваивание определенной клетке значения digit  
 puzzle[y \* 9 + x] = digit;  
 }  
}

**4)puzzleView**

package com.example.sudoku.view;  
  
import android.annotation.SuppressLint;  
import android.content.Context;  
import android.graphics.Canvas;  
import android.graphics.Paint;  
import android.graphics.Rect;  
import android.graphics.Paint.FontMetrics;  
import android.graphics.Paint.Style;  
import android.util.Log;  
import android.view.MotionEvent;  
import android.view.View;  
  
import com.example.sudoku.R;  
  
public class puzzleView extends View {  
 private static final String *TAG* = "Sudoku";  
 private final GameActivity game;  
 private float width; // ширина одной клетки судоку  
 private float height; // высота одной клетки судоку  
 private int selX; // координата Х выделенной клетки  
 private int selY; // координата У выделенной клетки  
 private final Rect selRect = new Rect();  
  
 public puzzleView(Context context) {  
 super(context);  
 this.game = (GameActivity) context;  
 setFocusableInTouchMode(true); // функция используется для включения просмотра фокуса в режиме сенсорного режима.  
 }  
  
 @Override  
 protected void onSizeChanged(int w, int h, int oldw, int oldh) { //инициализация рамеров клетки и рисование пустого прямоугольника через getRect  
 width = w / 9f; //деление всего экрана на 9 столбиков с шириной width  
 height = h / 9f; //деление экрана на 9 строк с высотой height  
 getRect(selX, selY, selRect); //переход к методу getRect для отрисовки прямоугольника  
 super.onSizeChanged(w, h, oldw, oldh);  
 }  
  
 private void getRect(int x, int y, Rect rect) { //создаем пустой прямоугольник и задаем координаты точек  
 rect.set((int) (x \* width), (int) (y \* height), (int) (x \* width + width), (int) (y \* height + height));  
 }  
 //инициализация цветов для метода onDraw (отрисовка)  
 private Paint lines = new Paint(); //отрисовка обычных линий  
 private Paint linesDiv3 = new Paint(); //отрисовка линий, разделяющих поле на 9 квадратов 3 на 3  
 private Paint background = new Paint(); //инициализация цвета фона поля  
 private Paint foreground = new Paint(Paint.*ANTI\_ALIAS\_FLAG*);//отрисовка цифр, вводимых в ячейку  
 private Paint selected = new Paint();//инициализация цвета выделенной ячейки  
 @Override  
 protected void onDraw(Canvas canvas) { //метод с отрисовкой границ, столбцов, строк, выделения ячейки  
 background.setColor(getResources().getColor(R.color.*colorwhite*)); //присваивание белого цвета фону поля  
 canvas.drawRect(0, 0, getWidth(), getHeight(), background); //закрашивание белым цветом всего поля по координатам  
  
 lines.setColor(getResources().getColor(R.color.*colorlightblack*)); //присваивание черного цвета обычным линиям  
 lines.setStrokeWidth(1F);  
  
 linesDiv3.setColor(getResources().getColor(R.color.*colorblack*));//присваивание черного цвета линиямДив3  
 linesDiv3.setStrokeWidth(5F);//присваивание данным линиям толщины, для различия с отсальными линиями  
 //отрисовка всех строк и столбцов обычным черным  
 for (int i = 0; i < 10; i++) {  
 canvas.drawLine(0, i \* height, getWidth(), i \* height, lines);  
 canvas.drawLine(0, i \* height + 1, getWidth(), i \* height + 1, lines);  
 canvas.drawLine(i \* width, 0, i \* width, getHeight(), lines);  
 canvas.drawLine(i \* width + 1, 0, i \* width + 1, getHeight(), lines);  
 }  
 //проход по полю и отрисовка жирным черным цветом только каждую третью строку и столбец  
 for (int i = 0; i < 10; i++) {  
 if (i % 3 != 0) //проверка условия, третья ли это линия  
 continue;  
 canvas.drawLine(0, i \* height, getWidth(), i \* height, linesDiv3);  
 canvas.drawLine(0, i \* height + 1, getWidth(), i \* height + 1, linesDiv3);  
 canvas.drawLine(i \* width, 0, i \* width, getHeight(), linesDiv3);  
 canvas.drawLine(i \* width + 1, 0, i \* width + 1, getHeight(), linesDiv3);  
 }  
 //отрисовка цифр и присваивание вводимо  
  
 foreground.setColor(getResources().getColor(R.color.*colorblack*));//присваивание черного цвета цвету текста (цифр)  
 foreground.setStyle(Style.*FILL*);//присваивания стиля шрифта  
 foreground.setTextSize(height \* 0.75f); //размер цифры в ячейке  
 foreground.setTextScaleX(width / height);//позволяет растянуть/сжать текст  
 foreground.setTextAlign(Paint.Align.*CENTER*);//расположение текста по центру ячейки  
 // отрисовка цифры в центре ячейки  
 FontMetrics fm = foreground.getFontMetrics();  
 // центровка по координате Х  
 float x = width / 2;  
 // центровка по У  
 float y = height / 2 - (fm.ascent + fm.descent) / 2;  
 for (int i = 0; i < 9; i++) {  
 for (int j = 0; j < 9; j++) {  
 canvas.drawText(this.game.getTileString(i, j), i \* width + x, j \* height + y, foreground);  
 }  
 }  
 selected.setColor(getResources().getColor(R.color.*colorlightgrey*)); //инициализация цвета для выделенной пользователем ячейки  
 selected.setAlpha(127);//присваивание цвету выделения прозрачности, чтобы видеть цифру, находящуюся в выделенной ячейке  
 canvas.drawRect(selRect, selected);//отрисовка прямоугольника(квадрата) размером с ячейку и залитие цветом, выбранным выше  
 }  
  
 private void select(int x, int y) {//новые координаты + вычисление новой области выделения  
 invalidate(selRect);  
 selX = Math.*min*(Math.*max*(x, 0), 8);//вычисляем новые координаты х и у выделения  
 selY = Math.*min*(Math.*max*(y, 0), 8);  
 getRect(selX, selY, selRect);//вычисление нового прямоугольника выделения  
 invalidate(selRect);  
 }  
  
 @SuppressLint("ClickableViewAccessibility")  
 @Override  
 public boolean onTouchEvent(MotionEvent event) {  
 if (event.getAction() != MotionEvent.*ACTION\_DOWN*)  
 return super.onTouchEvent(event);  
 select((int) (event.getX() / width), (int) (event.getY() / height));  
 game.showKeypadOrError(selX, selY);  
 return true;  
 }  
  
 public void setSelectedTile(int tile) { //изменение числа в ячейке  
 if (game.setTileIfValid(selX, selY, tile)) {  
 invalidate();// можно изменить подсказки  
 } else {  
 //число не подходит для данной ячейки  
 Log.*d*(*TAG*, "setSelectedTile: invalid: " + tile);  
  
 }  
 }  
}

**5)keypad**

package com.example.sudoku;  
  
import android.app.Dialog;  
import android.content.Context;  
import android.os.Bundle;  
import android.view.KeyEvent;  
import android.view.View;  
  
import com.example.sudoku.view.puzzleView;  
  
public class keypad extends Dialog {  
  
 private final View[] keys = new View[9]; //массив содержащий все кнопки от 1 до 9  
 private View keypad;  
 private final int[] useds; //массив использованных цифр  
 private final puzzleView puzzleView;  
  
 public keypad(Context context, int[] useds, com.example.sudoku.view.puzzleView puzzleView) {  
 super(context);  
 this.useds = useds;  
 this.puzzleView = puzzleView;  
 }  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setTitle(R.string.*keypad\_title*);  
 setContentView(R.layout.*keypad*); //открытие окна с цифрами  
 findViewsById(); //метод с присваиванием id  
 for (int element : useds) {  
 if (element != 0)  
 keys[element - 1].setVisibility(View.*INVISIBLE*); //кнопка скрывается, и пользователь ее не видит  
 }  
 setListeners();//устанавливает обработчики для каждой кнопки  
 }  
  
  
 @Override  
 public boolean onKeyDown(int keyCode, KeyEvent event) { //метод нажатий на кнопки  
 int tile = 0;  
 switch (keyCode) { //при нажатии на какую либо из кнопок ей присваиваются ее значение которое пользователь хочет ввести в ячейку  
 case KeyEvent.*KEYCODE\_1*: tile = 1; break;  
 case KeyEvent.*KEYCODE\_2*: tile = 2; break;  
 case KeyEvent.*KEYCODE\_3*: tile = 3; break;  
 case KeyEvent.*KEYCODE\_4*: tile = 4; break;  
 case KeyEvent.*KEYCODE\_5*: tile = 5; break;  
 case KeyEvent.*KEYCODE\_6*: tile = 6; break;  
 case KeyEvent.*KEYCODE\_7*: tile = 7; break;  
 case KeyEvent.*KEYCODE\_8*: tile = 8; break;  
 case KeyEvent.*KEYCODE\_9*: tile = 9; break;  
 default:  
 return super.onKeyDown(keyCode, event);  
 }  
 if (isValid(tile)) returnResult(tile);  
 return true;  
 }  
  
 private void returnResult(int tile) { //возврат выбранной цифры  
 puzzleView.setSelectedTile(tile);  
 dismiss();  
 }  
 private boolean isValid(int tile) { //метод проверяющий подходит ли данная цифра для этой ячейки  
 for (int t : useds) {  
 if (tile == t) return false;  
 }  
 return true;  
 }  
  
 private void findViewsById() { //присваиваение каждой кнопке и окну с цифрами его id  
 keypad = findViewById(R.id.*keypad*);  
 keys[0] = findViewById(R.id.*keypad\_1*);  
 keys[1] = findViewById(R.id.*keypad\_2*);  
 keys[2] = findViewById(R.id.*keypad\_3*);  
 keys[3] = findViewById(R.id.*keypad\_4*);  
 keys[4] = findViewById(R.id.*keypad\_5*);  
 keys[5] = findViewById(R.id.*keypad\_6*);  
 keys[6] = findViewById(R.id.*keypad\_7*);  
 keys[7] = findViewById(R.id.*keypad\_8*);  
 keys[8] = findViewById(R.id.*keypad\_9*);  
 }  
 private void setListeners() { //установка обработчиков для каждой из кнопки  
 for (int i = 0; i < keys.length; i++) {  
 final int t = i + 1;  
 keys[i].setOnClickListener(new View.OnClickListener(){  
 public void onClick(View v) {  
 returnResult(t);  
 }});  
 }  
 keypad.setOnClickListener(new View.OnClickListener(){  
 public void onClick(View v) {  
 returnResult(0);  
 }});  
 }  
}