Санкт-Петербургский государственный политехнический университет

Институт компьютерных наук и технологий

**Кафедра «Компьютерные системы и программные технологии»**

**КУРСОВОЙ ПРОЕКТ**

**Игра «2048»**

по дисциплине «Программирование под Android»

Выполнила

студентка гр.33531/2 Е.Ю.Рязанова

Руководитель А.О.Алексюк

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Санкт-Петербург

2018

Санкт-Петербургский политехнический университет Петра Великого

**ЗАДАНИЕ**

**НА ВЫПолнение курсовоГО ПРОЕКТА**

студентке группы 33531/2 Рязановой Евгении Юрьевне

***1. Тема проекта (работы):*** Программирование игры «2048» под Андроид.

***2. Срок сдачи законченного проекта (работы)*** 06.02.19 г.

***3. Содержание пояснительной записки***: введение, основная часть (раскрывается структура основной части), заключение.

***4. Техническое задание***: реализовать игру «2048» для ОС Android в программе для разработки приложений под ОС Android - AndroidStudio. Язык программирования, используемый для реализации – Java или Kotlin, на выбор.

***Дата получения задания***: «21» сентября 2018 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.О.Алексюк

*(подпись)*

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_ Е.Ю.Рязанова

*(подпись студента)*

06.02.2019 г.

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| Введение . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 4 |
| Структура проекта. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 5 |
| Главный экран. MainActivity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 5 |
| Основной экран. GameActivity.. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 5 |
| Логическая реализация. Класс Matrix . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 8 |
| Сервис MusicService. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 10 |
| OnSwipeTouchListener. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 10 |
| ImageAdapter. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . | 10 |
| RulesActivity. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. | 11 |
| Ресурсы. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 11 |
| Тестирование. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 11 |
| Заключение . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 12 |
| Список использованных источников . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 13 |

**ВВЕДЕНИЕ**

Разработка мобильных приложений - одно из наиболее популярных направлений в программировании. И это неудивительно, с учетом повсеместного использования гаджетов. Андроид – мобильная операционная система, которая используется повсеместно. Цель данной курсовой работы познакомиться с данной платформой и получить навыки разработки под нее. Для этого будет написана игра «2048», в которой пользователь взаимодействует с игровым полем 4x4, заполненным числами {2,4,8,…2048} , для победы ему необходимо получить число 2048. Для удобства пользователя в приложении помимо окна с игровым полем будут добавлены главный экран, правила, а так же фоновая музыка. Чтобы сделать проект отличающимся от оригинала, поле будет заполнено не обычными числами, а фотографиями биатлонистов.

**Структура проекта**

1. **Главный экран. MainActivity**

Первое, что видит пользователь при запуске - меню приложения. Взаимодействие осуществляется через кнопки: начать игру, правила, выход. *MainActivity* :

**import** android.content.Intent;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
  
**public class** MainActivity **extends** AppCompatActivity **implements** View.OnClickListener {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 Button startButton = findViewById(R.id.***startButton***);  
 Button rulesButton = findViewById(R.id.***rulesButton***);  
 Button exitButton = findViewById(R.id.***exitButton***);  
 startButton.setOnClickListener(**this**);  
 rulesButton.setOnClickListener(**this**);  
 exitButton.setOnClickListener(**this**);  
  
 }  
  
 **public void** onClick(View view){  
 **switch** (view.getId()) {  
 **case** R.id.***startButton***: {  
 Intent gameIntent = **new** Intent(MainActivity.**this**, GameActivity.**class**);  
 startActivity(gameIntent);  
 **break**;  
 }  
 **case** R.id.***rulesButton***: {  
 Intent rulesIntent = **new** Intent(MainActivity.**this**, RulesActivity.**class**);  
 startActivity(rulesIntent);  
 **break**;  
 }  
 **case** R.id.***exitButton***: {  
 finish();  
 **break**;  
 }  
 }  
  
 }  
  
  
}

Данный класс содержит в себе метод *onCreate()*, в котором создаются кнопки, обработчик нажатий и намерения для перехода на следующую активити*.* Метод *onClick* описывает поведение при нажатии на кнопки.

*activity\_main –* layout, расположенный в ресурсах ( в двух вариантах – portrait и land).

**2. Основной экран. GameActivity**

**package** com.example.eugenia.game\_2048;  
  
**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.widget.AdapterView;  
**import** android.widget.GridView;  
**import** android.widget.TextView;  
  
**public class** GameActivity **extends** Activity {  
 Matrix **matr**;  
 GridView **gridView** = **null**;  
 TextView **scoreView** ;  
 **public** Integer **score**;  
 **private final int CELLS\_NUMBER** = 4;  
 **private final int INIT\_CELLS** = 2;  
 **private static final** String ***KEY\_COUNT*** = **"COUNT"**;  
 **private static final** String ***KEY\_ARRAY*** = **"ARRAY"**;  
 **private static final** String ***KEY\_VICTORY*** = **"VICTORY"**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_game***);  
 Intent serviceIntent = **new** Intent(GameActivity.**this**, MusicService.**class**);  
 startService(serviceIntent);  
 **gridView** = (GridView) findViewById(R.id.***gridview***);  
 **scoreView** = (TextView) findViewById(R.id.***scoreView***);  
 **score** = 0;  
 **matr** = **new** Matrix();  
 **for** (**int** i = 0; i < **INIT\_CELLS**; i++) { *//Создаем 2 начальные ячейки* **matr**.newStep();  
 }  
 **score** = **matr**.**score**;  
 **scoreView**.setText(getString(R.string.***result***, **score**));  
 drawOnView(fill\_tab());  
 **gridView**.setOnTouchListener(**new** OnSwipeTouchListener(GameActivity.**this**){  
 @Override  
 **public void** onSwipeTop() {  
 **if**(**matr**.m\_up()) {  
 drawOnView(fill\_tab());  
 **score** = **matr**.**score**;  
 **scoreView**.setText(getString(R.string.***result***, **score**));  
 **if** (**matr**.**victory**) showEndGame();  
 }  
 **else** showEndGame();  
 }  
  
 @Override  
 **public void** onSwipeBottom() {  
 **if**(**matr**.m\_down()) {  
 drawOnView(fill\_tab());  
 **score** = **matr**.**score**;  
 **scoreView**.setText(getString(R.string.***result***, **score**));  
 **if** (**matr**.**victory**) showEndGame();  
 }  
 **else** showEndGame();  
 }  
  
 @Override  
 **public void** onSwipeLeft() {  
 **if**(**matr**.m\_left()) {  
 drawOnView(fill\_tab());  
 **score** = **matr**.**score**;  
 **scoreView**.setText(getString(R.string.***result***, **score**));  
 **if** (**matr**.**victory**) showEndGame();  
 }  
 **else** showEndGame();  
 }  
  
 @Override  
 **public void** onSwipeRight() {  
 **if**(**matr**.m\_right()) {  
 drawOnView(fill\_tab());  
 **score** = **matr**.**score**;  
 **scoreView**.setText(getString(R.string.***result***, **score**));  
 **if** (**matr**.**victory**) showEndGame();  
 }  
 **else** showEndGame();  
 }  
 });  
  
 }  
  
 @Override  
 **protected void** onStart() {  
 **super**.onStart();  
 startService(**new** Intent(GameActivity.**this**, MusicService.**class**));  
 }  
  
 @Override  
 **protected void** onStop() {  
 **super**.onStop();  
 stopService(**new** Intent(GameActivity.**this**, MusicService.**class**));  
 }  
  
 **void** drawOnView(Integer[] picsArray) {  
 **gridView**.setAdapter(**new** ImageAdapter(**this**, picsArray ));  
 **gridView**.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {  
 @Override  
 **public void** onItemClick(AdapterView<?> parent, View view, **int** position, **long** id) {  
  
 }  
 });  
 }  
  
 *//поиск нужной картинки* Integer findPic(**int** cellVal) {  
 **int** picId = 0;  
 **switch** (cellVal) {  
 **case** 2:  
 picId = R.drawable.***pic2***;  
 **break**;  
 **case** 4:  
 picId = R.drawable.***pic4***;  
 **break**;  
 **case** 8:  
 picId = R.drawable.***pic8***;  
 **break**;  
 **case** 16:  
 picId = R.drawable.***pic16***;  
 **break**;  
 **case** 32:  
 picId = R.drawable.***pic32***;  
 **break**;  
 **case** 64:  
 picId = R.drawable.***pic4***;  
 **break**;  
 **case** 128:  
 picId = R.drawable.***pic128***;  
 **break**;  
 **case** 256:  
 picId = R.drawable.***pic256***;  
 **break**;  
 **case** 512:  
 picId = R.drawable.***pic512***;  
 **break**;  
 **case** 1024:  
 picId = R.drawable.***pic1024***;  
 **break**;  
 **case** 2048:  
 picId = R.drawable.***pic2048***;  
 **break**;  
 **default**:  
 picId = R.drawable.***pic0***; *// Свободна* }  
 **return** picId;  
 }  
  
 *//заполняем массив id картинок для передачи в адаптер* Integer[] fill\_tab() {  
 Integer[] picsId = **new** Integer[**CELLS\_NUMBER** \* **CELLS\_NUMBER**];  
 **int** c = 0;  
 **for** (**int** i = 0; i < **CELLS\_NUMBER**; i++) {  
 **for** (**int** j = 0; j < **CELLS\_NUMBER**; j++) {  
 **int** cellValue = **matr**.get(i, j);  
 picsId[c] = findPic(cellValue);  
 c++;  
 }  
  
 }  
 **return** picsId;  
 }  
 *//создание диалогового окна* **void** showEndGame() {  
 AlertDialog.Builder endDialog = **new** AlertDialog.Builder(GameActivity.**this**);  
 endDialog.setTitle(R.string.***dialogTitle***);  
 **if**(**matr**.**victory**)  
 endDialog.setMessage(getString(R.string.***victoryMessage***,getString(R.string.***result***, **score**)));  
 **else** endDialog.setMessage(getString(R.string.***failMessage***,getString(R.string.***result***, **score**)));  
 endDialog.setPositiveButton(R.string.***newGame***, **new** DialogInterface.OnClickListener() {  
 **public void** onClick(DialogInterface dialog, **int** arg1) {  
 Intent gameIntent = **new** Intent(GameActivity.**this**, GameActivity.**class**);  
 startActivity(gameIntent);  
 }  
 });  
 endDialog.setNegativeButton(R.string.***backToMenu***, **new** DialogInterface.OnClickListener(){  
 **public void** onClick(DialogInterface dialog, **int** arg1) {  
 Intent menuIntent = **new** Intent(GameActivity.**this**, MainActivity.**class**);  
 startActivity(menuIntent);  
 }  
 });  
 endDialog.setCancelable(**false**);  
 endDialog.show();  
 }  
  
 @Override  
 **protected void** onSaveInstanceState(Bundle outState) {  
 **super**.onSaveInstanceState(outState);  
 outState.putBoolean(***KEY\_VICTORY***, **matr**.**victory**);  
 outState.putInt(***KEY\_COUNT***, **score**);  
 outState.putIntegerArrayList(***KEY\_ARRAY***,**matr**.**allList**);  
  
 }  
  
 @Override  
 **protected void** onRestoreInstanceState(Bundle savedInstanceState) {  
 **super**.onRestoreInstanceState(savedInstanceState);  
 **matr**.**victory** = savedInstanceState.getBoolean(***KEY\_VICTORY***);  
 **score** = savedInstanceState.getInt(***KEY\_COUNT***);  
 **matr**.**score** = savedInstanceState.getInt(***KEY\_COUNT***);  
 **scoreView**.setText(getString(R.string.***result***, **score**));  
 **matr**.**allList** = savedInstanceState.getIntegerArrayList(***KEY\_ARRAY***);  
 **int** c =0;  
 **for** (**int** i = 0; i < **CELLS\_NUMBER**; i++) {  
 **for** (**int** j = 0; j < **CELLS\_NUMBER**; j++) {  
 **matr**.**m**[i][j]=**matr**.**allList**.get(c);  
 c++;  
 }  
 drawOnView(fill\_tab());  
  
 }  
 }  
}

Активити содержит layout *activity\_game*, в котором расположены GridView –реализует поле игры и TextView – для отображения набранных очков. Жизненный цикл GameActivity описан в методах *onCreate(), onStart(), onStop().* Методы *onSaveInstanceState()* и *onRestoreInstanceState()* нужны для сохранения данных о текущем состоянии поля и количестве очков, если активность будет разрушена(например, при повороте экрана). Методы *fill\_tab()*, *findPic()* нужны для формирования данных, которые будут переданы в адаптер в методе *drawOnView()*. Метод *showEndGame()* нужен для создания диалогового окна при завершении игры. В зависимости от результата – поражения или победы отображается сообщение и 2 кнопки. Пользователь может вернуться в главное меню или начать заново.

**3. Логическая реализация. Класс Matrix**

Логика игры описана в классе Matrix[4]. Игровое поле хранится в двумерном массиве, методы *m\_up(), m\_down(), m\_right(), m\_left()* обрабатывают изменение матрицы при сдвигах.

**public class** Matrix {  
 **private final int CELLS\_NUMBER** = 4;  
 **private final int INIT\_CELLS** = 2;  
 Point **p**; *// p.x = j p.y = i (индексы матрицы m[][])* **int m**[][];  
 ArrayList<Integer> **allList**; *// одномерный массив - матрица m[][] в виде одномерного массива* Random **rnd**;  
 **boolean victory**;  
 **int score** =0;  
 **public** Matrix() {  
 **m** = **new int**[][]{ {0, 0, 0, 0 },  
 {0, 0, 0, 0 },  
 {0, 0, 0, 0 },  
 {0, 0, 0, 0 } };  
  
  
 **allList** = **new** ArrayList<>();  
  
 **for**( **int** i=0; i<**m**.**length**; i++)  
 **for**( **int** j=0; j<**m**[i].**length**; j++ )  
 setAllList(i,j,**m**[i][j]);  
 **rnd** = **new** Random();  
 }  
  
 *// занести в одномерный массив allList число, по индексу N, соответствующее m[i][j]* **void** setAllList(**int** i, **int** j, **int** value) {  
 **int** N; *// index allList* N = i\* **CELLS\_NUMBER** +j;  
 **allList**.add(N, value);  
 }  
  
 *// получить по индексу N массива allList индексы для матрицы m[][]* Point getAllList( **int** N) {  
 **int** i = N/ **CELLS\_NUMBER**;  
 **int** k = i\* **CELLS\_NUMBER**;  
 **int** j = N-k;  
 Point p = **new** Point(j,i);  
 **return** p;  
 }  
  
 *// добавить новое число 2* **boolean** newStep() {  
 **if**( !**allList**.contains(0)) *// если нет свободных мест* **return false**;  
 *// массив индексов свободных мест в allList (для быстрого поиска рандома)* ArrayList<Integer> index0 = **new** ArrayList<>();  
  
 **for**( **int** i=0; i<**allList**.size(); i++) {  
 **if**( **allList**.get(i).compareTo(0) == 0) *// место свободное* index0.add(i);  
 }  
  
 **int** freeIndex = **rnd**.nextInt(index0.size()); *// берем рандомный свободный индекс* **int** indexInList= index0.get(freeIndex);  
 **allList**.set(indexInList, **INIT\_CELLS**); *// заносим на свободное место 2  
 // заносим и в матрицу m[][] эту 2* Point p = getAllList( indexInList);  
 **m**[p.**y**][p.**x**] = **INIT\_CELLS**;  
  
 **return true**;  
 }  
  
 **boolean** m\_down( ) {  
 **for**( **int** i=0; i<**m**.**length**; i++) {  
 **int**[] arg = getColumn(i);  
 **int**[] rab= **new int**[arg.**length**]; *//меняем порядок элементов на обратный* **for**(**int** a=0; a<rab.**length**;a++){  
 rab[a]=arg[rab.**length**-a-1];  
 }  
 arg=rab;  
 **int**[] result= moveLine(arg); *//сдвигаем* **for**(**int** a=0; a<rab.**length**;a++){ *//возвращаем исходный порядок* rab[a]=result[rab.**length**-a-1];  
 }  
 result=rab;  
 setColumn(i,result);  
 }  
 **allList**.clear(); *// чистим одномерный массив* **for**( **int** i=0; i<**m**.**length**; i++) *// записываем новые значения* **for**( **int** j=0; j<**m**[i].**length**; j++ )  
 setAllList(i,j,**m**[i][j]);  
 **return** newStep(); *// добавляем (если можно) новую 2*...  
  
 *//возвращает массив ячеек i-ой строки* **int**[] getRow(**int** i){  
 **return m**[i];  
 }  
 *//меняет i-ую строку* **void** setRow(**int** i,**int**[] newRow){  
 **m**[i]=newRow;  
 }  
 *//Возвращает i столбец* **int**[] getColumn(**int** i){  
 **int**[] res= **new int**[**CELLS\_NUMBER**];  
 **for**(**int** j = 0; j< **CELLS\_NUMBER**; j++){  
 res[j]=**m**[j][i];  
 }  
 **return** res;  
 }  
  
 *//Меняет сотояние ячеек i столбца* **void** setColumn(**int** i,**int**[] newColumn){  
 **for**(**int** j = 0; j< **CELLS\_NUMBER**; j++){  
 **m**[j][i]=newColumn[j];  
 }  
 }  
  
 **private final int INCREASE\_SCORE** = 2;  
 **private final int CELL\_2048** = 2048;  
  
 **private int**[] moveLine(**int**[]oldLine){  
  
 **int**[]noZero= **new int**[oldLine.**length**]; *//убираем нули* **int** k=0;  
 **for** (**int** i=0; i<oldLine.**length**;i++){  
 **if**(oldLine[i]!=0){  
 noZero[k]=oldLine[i];  
 k++;  
 }  
 }  
 **for**(**int** i=k; i<noZero.**length**; i++){ *// чтобы массив не содержал null, заполним его нулями* noZero[i]=0;  
 }  
  
 **int**[] newLine=**new int**[noZero.**length**];  
 **int** f=0;  
 **int** i=0;  
 **while** (i<noZero.**length**){  
 **if**((noZero[i]!=0)&&(i+1<noZero.**length**)&&(noZero[i]==noZero[i+1])){  
 newLine[f]=noZero[i]\***INCREASE\_SCORE**;  
 **score** +=noZero[i]\***INCREASE\_SCORE**;  
 *// если получили 2048 завершить игру* **if**(newLine[f]==**CELL\_2048**) **victory** = **true**;  
 i++;  
 } **else**{  
 newLine[f]=noZero[i];  
 }  
 i++;  
 f++;  
  
 }  
  
 **for**(**int** j=f; j<newLine.**length**; j++){  
 newLine[f]=0;  
 }  
  
  
  
 **return** newLine;  
 }

**4. Сервис MusicService**

Сервис для проигрывания музыки, вызывается в GameActivity.

**public class** MusicService **extends** Service {  
 **private** MediaPlayer **musicPlayer**;  
 @Override  
 **public** IBinder onBind(Intent intent) {  
 **return null**;  
 }  
 @Override  
 **public void** onCreate() {  
 **super**.onCreate();  
 **musicPlayer** = MediaPlayer.*create*(**this**,R.raw.***adagio***);  
 **musicPlayer**.setLooping(**true**);  
 }  
  
 @Override  
 **public int** onStartCommand(Intent intent, **int** flags, **int** startId) {  
 **musicPlayer**.start();  
 **return super**.onStartCommand(intent, flags, startId);  
 }  
  
 @Override  
 **public void** onDestroy() {  
 **super**.onDestroy();  
 **musicPlayer**.stop();  
 }  
}

**5. OnSwipeTouchListener**

Класс, описывающий обработку касания экрана – swipe. Был взят с GitHub[5].

**6. ImageAdapter**

Класс, реализующий адаптер, необходимый для отображения картинок на экране.

**public class** ImageAdapter **extends** BaseAdapter {  
 Context **mContext**;  
 Integer [] **picsId**;  
 **public** ImageAdapter(Context c, Integer[] list) {  
 **mContext** = c;  
 **picsId** = list;  
 }  
 @Override  
 **public int** getCount() {  
  
 **return picsId**.**length**;  
 }  
  
 @Override  
 **public** Object getItem(**int** position) {  
  
 **return picsId**[position];  
 }  
  
 @Override  
 **public long** getItemId(**int** position) {  
  
 **return** 0;  
 }  
  
 @Override  
 **public** View getView(**int** position, View convertView, ViewGroup parent) {  
 ImageView imageView;  
 **if** (convertView == **null**) {  
 imageView = **new** ImageView(**mContext**);  
 imageView.setLayoutParams(**new** ViewGroup.LayoutParams(ViewGroup.LayoutParams.***WRAP\_CONTENT***,ViewGroup.LayoutParams.***WRAP\_CONTENT***));  
 imageView.setScaleType(ImageView.ScaleType.***CENTER\_CROP***);  
  
 } **else** {  
 imageView = (ImageView) convertView;  
  
 }  
 imageView.setImageResource(**picsId**[position]);  
 **return** imageView;  
 }  
  
  
}

**7. RulesActivity**

Отображает layout *activity\_ rules,* который содержит текстовое поле с описанием игры.

**public class** RulesActivity **extends** Activity {  
 @Override  
 **protected void** onCreate( Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_rules***);  
 }  
}

**8. Ресурсы**

Используемые ресурсы:

* layout : (*res – layout)*

*activity\_main, activity\_game, activity\_ rules.* Содержат альтернативные ресурсы – портретную и альбомную ориентации.

* цвета (*res – values – colors*)
* строки (*res – values – strings)*
* музыка (*res – raw –adagio.mp3);*
* изображения (*res – drawable);*

**9. Тестирование**

Unit тесты для функций сдвига матрицы находятся в классе ExampleUnitTest.

В классе *MainActivityInstrumentedTest* тестируется отображение кнопок. Также тестируется корректность отображения кнопки *выход* после поворота экрана и завершение активности при нажатии *back*.

В классе *GameActivityInstrumentedTest*  содержатся тесты, которые проверяют правильность отображения очков, а также их сохранение при повороте экрана. Еще один тест проверяет корректность вызова функции *showEndGame()* и появление диалога на экране.

**ЗАКЛЮЧЕНИЕ**

В ходе курсовой работы были получены навыки разработки приложений под Android. Рассмотрены основные компоненты, такие как Service и Activity. Изучен жизненный цикл их и особенности взаимодействия с ними. Освоена работа с Adapter и AlertDialog.

Полный код проекта выложен в соответствующем репозитории на GitHub:

*https://github.com/KatPierce/AndroidProject*

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Официальная документация Android.

URL: https://developer.android.com/guide/

1. Сайт Александра Климова.

URL: http://developer.alexanderklimov.ru/android/theory/

1. Уроки по Android // https://startandroid.ru. Дата последнего обновления: 01.12.2017.

URL: https://startandroid.ru/ru/uroki.html

1. Игра 2048. Рязанова Евгения. Дата последнего обновления: 30.06.2017.

<URL:https://github.com/KatPierce/task-3>

1. Репозиторий GitHub пользователя amit-spaceo. Дата последнего обновления: 18.05.2018.

последнего обновления: 30.06.2017.

<URL:https://github.com/spaceotech/SO_GestureDetect/blob/master/app/src/main/java/com/spaceo/gesturedetectdemo/OnSwipeTouchListener.java>