Санкт-Петербургский государственный политехнический университет

Институт компьютерных наук и технологий

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

**КУРСОВОЙ ПРОЕКТ**

**Игра «Эрудит»**

по дисциплине «Программирование под Android»

Выполнил

студент гр.33531/2 Д.О.Архиреев

Руководитель А.О.Алексюк

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2018 г.

Санкт-Петербург

2018

Санкт-Петербургский политехнический университет Петра Великого

**ЗАДАНИЕ**

**НА ВЫПолнение курсовоГО ПРОЕКТА**

студент группы 33531/2 Архиреев Дмитрий Олегович

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

***2. Срок сдачи законченного проекта (работы)*** 28.12.18 г.

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

***4. Техническое задание***: реализовать игру «Эрудит» 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***: «01» октября 2018 г.

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

*(подпись)*

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_ Д.О.Архиреев

*(подпись студента)*

28.12.2018г.

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| Введение . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 4 |
| Цель проекта. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 5 |
| Задачи. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 5 |
| 1. Правила игры «Эрудит». Описание дальнейшей реализации. . . . . . . . | 5 |
| 2. Основная часть работы . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 6 |
| 3.Ресурсы. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 12 |
| 4. Тестирование и инспектор. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ……….. | 13 |
| 5. Результат работы готового приложения . . . . . . . . . . . . . . . . . . . . . . . ……. | 18 |
| Заключение . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 19 |
| Список использованных источников . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 20 |

**ВВЕДЕНИЕ**

В данном курсовом проекте по дисциплине «Программирование под Android» мной будет написана игра «Эрудит», реализующая полученное мной индивидуальное задание.

В работе будут использовано в качестве базового компонента Activity. Проверка слов на корректность вместо подгрузки большого словаря будет осуществляться с помощью Yandex словарей.

**Цель проекта**

Цель данного проекта научиться использовать базовые компоненты Android для создания своего работающего приложения. Цель будет достигнута в виде игры реализованной на языке Java «Эрудит»

**Задачи**

Для достижения поставленной цели мной будет решена задача написания, работающего Android-приложения, в котором будет реализована игра «Эрудит» и получены базовые навыки работы в среде AndroidStudio.

**1. Правила игры «Эрудит». Описание дальнейшей реализации**

В игре эрудит принимают участие 2 человека и более. Побеждает тот, кто первым наберет обусловленное количество очков. Слова составляются только по вертикали или горизонтали и должны читаться слева-направо или сверху-вниз. За один ход можно составить несколько слов из имеющихся 7 фишек. Новые слова составлять на свободном поле без увязки с предыдущими словами нельзя. В начале игры фишки выкладываются оборотной стороной и перемешиваются. Каждый игрок берет 7 фишек. Первое составленное слово располагают горизонтально или вертикально так чтобы одна из его букв пришлась на центральную клетку поля. Если игрок не может или не желает в свой ход составлять слова, он меняет фишки (от 1 до 7) и пропускает ход. Сумма очков каждого хода состоит из суммы очков составленных букв, а также премий, получаемых за размещение букв на премиальных клетках. Премиальные клетки для букв: очки буквы, расположенной на зеленой клетке, удваиваются, на желтой - утраиваются. Премиальные клетки для слов: Если одна из букв слова расположена на синей клетке, сумма очков всего слова удваивается, на красной – утраивается. Если буквы слова одновременно расположены на премиальных клетках букв и слов, сначала подсчитывается премия за букву, потом - за слово. Премия за букву и слово засчитывается лишь тому игроку, который первым сумел занять премиальные клетки. Каждый игрок, использовавший за один ход все 7 фишек, получается дополнительную премию в 15 очков

Структура приложения:

1. Главное меню
2. Ввод имен игроков
3. Об игре
4. Самая игра

**2. Основная часть работы**

Рассмотрим некоторые основные части игры

У приложения разделены графическая часть и логика игры, поэтому для начала покажу структуру приложения и выпишу все реализованные классы и опишу их задачи

1. MainActivity – собственно главное меню игры, содержит в себе две кнопки
2. GameStart – активити с двумя полями для ввода имен игроков и кнопкой начала игры
3. Game – активити где происходит сама игра. Содержит в себе frameLayout и 4 кнопки управления игрой

Листинг:

public class Game extends Activity {

public Player player1;

public Player player2;

public GameField gameField;

private TextView textView;

private FrameLayout frameLayout;

private boolean orientation = false;

private Point size;

private int sizeCell;

private final int CELLS\_IN\_ROW = 15;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_game);

textView = (TextView) findViewById(R.id.namePlayer);

initGameField();

resetGameField(savedInstanceState);

}

private void initGameField() {

Intent initIntent = getIntent();

player1 = new Player(initIntent.getStringExtra("firstPlayer"));

player2 = new Player(initIntent.getStringExtra("secondPlayer"));

size = new Point();

Display display = getWindowManager().getDefaultDisplay();

display.getSize(size);

if (this.getResources().getConfiguration().orientation == Configuration.ORIENTATION\_LANDSCAPE) {

//Округление в сторону нуля

sizeCell = ((size.y / CELLS\_IN\_ROW) / 10) \* 10;

orientation = true;

}

else if (this.getResources().getConfiguration().orientation == Configuration.ORIENTATION\_PORTRAIT) {

//Округление в сторону 0

sizeCell = ((size.x / CELLS\_IN\_ROW) / 10) \* 10;

orientation = false;

}

gameField = new GameField(this, player1, player2, sizeCell);

frameLayout = (FrameLayout) findViewById(R.id.frameGame);

gameField.setOrientation(orientation);

gameField.setSizeScreen(size);

gameField.setStartPoint();

frameLayout.addView(gameField);

}

private void resetGameField(Bundle savedInstanceState){

if (savedInstanceState != null){

gameField.resetGameField(savedInstanceState);

if(gameField.getGames().getFirstTurn()) {

if (!gameField.getGames().getTurn()) {

String str = gameField.getGames().getFirstPlayer().getName() +

"\n" +

gameField.getGames().getFirstPlayer().getScore();

textView.setText(str);

} else {

String str = gameField.getGames().getSecondPlayer().getName() +

"\n" +

gameField.getGames().getSecondPlayer().getScore();

textView.setText(str);

}

}

}

}

@Override

protected void onSaveInstanceState(Bundle saveState) {

super.onSaveInstanceState(saveState);

//Сохранение игроков

saveState.putString("firstPlayerName", gameField.getGames().getFirstPlayer().getName());

saveState.putString("secondPlayerName", gameField.getGames().getSecondPlayer().getName());

saveState.putInt("secondPlayerScore", gameField.getGames().getSecondPlayer().getScore());

saveState.putInt("firstPlayerScore", gameField.getGames().getFirstPlayer().getScore());

if(gameField.getGames().getFirstTurn()){

saveState.putCharArray("firstPlayerLetter",gameField.getGames().getFirstPlayer().getLetter());

saveState.putCharArray("secondPlayerLetter",gameField.getGames().getSecondPlayer().getLetter());

}

//сохрание переменных поля

saveState.putBoolean("firstTurn", gameField.getGames().getFirstTurn());

saveState.putBoolean("turn", gameField.getGames().getTurn());

saveState.putBoolean("nextTurn", gameField.getGames().getNextTurn());

//Сохранение поля

saveState.putCharArray("letterInBoard", gameField.getGames().getLetterInBoard());

saveState.putIntegerArrayList("indexLetterInBoard", gameField.getGames().getIndexLetterInBoadr());

//Сохранение размеров ячейки

saveState.putInt("sizeCell", gameField.getSizeCell());

saveState.putInt("sizeX", gameField.getSizeScreen().x);

saveState.putInt("sizeY", gameField.getSizeScreen().y);

//Сохранение оставшихся букв

saveState.putIntegerArrayList("freeLetter", gameField.getGames().getGameRule().getCountFreeLetter());

}

public void onClickNextTurn(View view) {

if (!gameField.getGames().getTurn()) {

String str = gameField.getGames().getSecondPlayer().getName() +

"\n" +

gameField.getGames().getSecondPlayer().getScore();

textView.setText(str);

gameField.changeTurn();

gameField.getGames().resetTurn();

} else {

String str = gameField.getGames().getFirstPlayer().getName() +

"\n" +

gameField.getGames().getFirstPlayer().getScore();

textView.setText(str);

gameField.changeTurn();

gameField.getGames().resetTurn();

}

}

public void onClickRecycle(View view) {

if (gameField.isTempListNumCells()) {

if (!gameField.getGames().getTurn()) {

String str = gameField.getGames().getSecondPlayer().getName() +

"\n" +

gameField.getGames().getSecondPlayer().getScore();

textView.setText(str);

gameField.changeTurnResetLetter();

gameField.getGames().resetTurn();

} else {

String str = gameField.getGames().getFirstPlayer().getName() +

"\n" +

gameField.getGames().getFirstPlayer().getScore();

textView.setText(str);

gameField.changeTurnResetLetter();

gameField.getGames().resetTurn();

}

}

}

public void onClickNewWord(View view) {

gameField.newWord();

textView.setText(gameField.getGames().tc);

}

public void onClickStartGame(View view) {

StringBuilder str = new StringBuilder();

if (!gameField.getGames().getFirstTurn()){

gameField.start();

str.append(gameField.getGames().getFirstPlayer().getName());

str.append("\n");

str.append(gameField.getGames().getFirstPlayer().getScore());

textView.setText(str.toString());

}

}

}

Методы onClick.. отвечают за управление игрой и вызов методов отображения изменения состояний поля и игроков на экране .

1. AboutActivity – содержит в себе информацию о версии игры и об авторе
2. GameFiled – отвечает за отрисовку Canvas на экране

Пример кода основных методов:

public void setStartPoint(){

if (!orientation){

int startX = (sizeScreen.x - sizeCell\*NUMBER\_CELL\_IN\_ROW) / 2;

games.setStartPoint( new Point(startX,indent), new Point(indent, sizeScreen.y - indent));

games.setIncPointX(startX);

games.setIncPointY(sizeCell);

}

else {

games.setStartPoint(new Point(0, indentY),new Point(sizeCell \* NUMBER\_CELL\_IN\_ROW + indent,

indentY\*2));

games.setIncPointX(0);

games.setIncPointY(sizeCell);

}

setPictures();

initBitmapListLetter();

games.setSizeCell(sizeCell);

games.startGame();

}

public void start(){

if (!games.getFirstTurn()){

games.addLetters();

games.setListCellLetter();

games.setFirstTurn(true);

}

invalidate();

}

Эти методы отвечают за старт игры определением размера ячеек и поля

Непосредственно сам метод onDraw который отвечает за постоянно отрисовывание игры:

@Override

public void onDraw(Canvas canvas){

super.onDraw(canvas);

try {

for (int i = 0; i < COUNT\_LETTER; i++){

paint.setStyle(Paint.Style.STROKE);

canvas.drawPath(games.getListCellLetter().get(i).getPath(),paint);

matrix.setTranslate(games.getListCellLetter().get(i).getStartPoint().x - 5,

games.getListCellLetter().get(i).getStartPoint().y - 5);

List<Character> list = games.getNextTurn() ? games.getFirstPlayer().getListLetter() : games.getSecondPlayer().getListLetter();

if (list.get(i) != ' ')

canvas.drawBitmap(bitmapListLetter.get(games.findLetter(list.get(i))),matrix,paint);

}

}catch (IndexOutOfBoundsException | NullPointerException e){

}

for (int i = 0; i < games.getWorkListCells().size(); i++){

matrix.setTranslate(games.getWorkListCells().get(i).getStartPoint().x,

games.getWorkListCells().get(i).getStartPoint().y);

paint.setStyle(Paint.Style.STROKE);

canvas.drawPath(games.getWorkListCells().get(i).getPath(), paint);

if (games.getWorkListCells().get(i).getLetter() != ' '){

canvas.drawBitmap(bitmapListLetterBoard.get(games.findLetter(games.getWorkListCells().get(i).getLetter())),matrix,paint);

}

else {

switch (games.getWorkListCells().get(i).getState()) {

case DEF\_CELL: {

canvas.drawBitmap(defpict, matrix, paint);

break;

}

case START\_POSITION: {

canvas.drawBitmap(startpict, matrix, paint);

break;

}

case X2LETTER: {

canvas.drawBitmap(x2Letterpict, matrix, paint);

break;

}

case X3LETTER: {

canvas.drawBitmap(x3Letterpict, matrix, paint);

break;

}

case X2WORD: {

canvas.drawBitmap(x2Wordpict, matrix, paint);

break;

}

case X3WORD: {

canvas.drawBitmap(x3Wordpict, matrix, paint);

break;

}

}

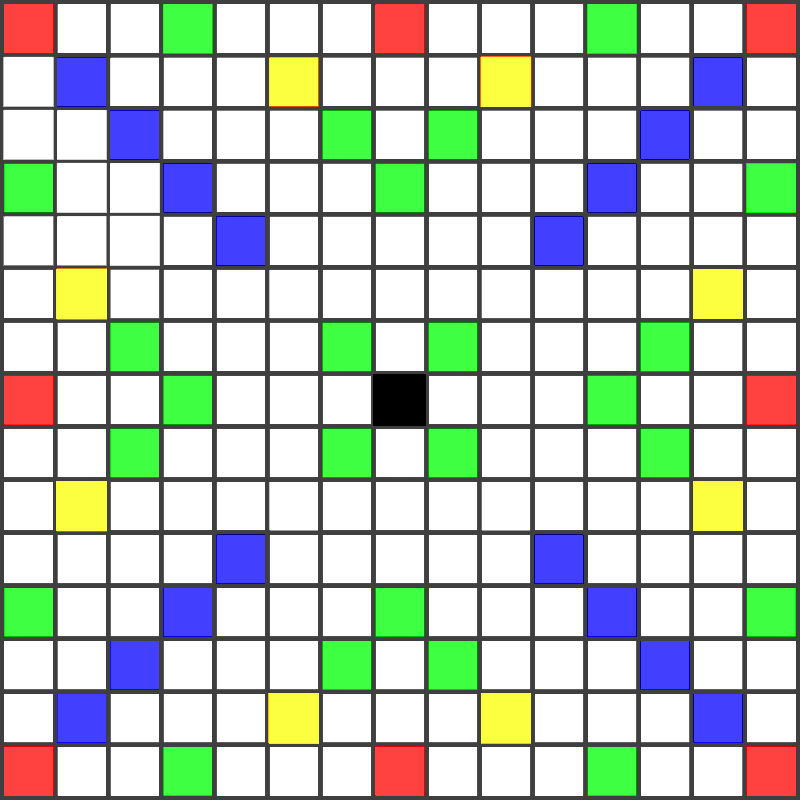
}

}

}

1. Cell – вспомогательный класс который отвечает за отдельную ячейку поля
2. СellForLetter – класс которы отвечает за контейнер для букв игрока
3. Letter – класс который содержит в себе символ буквы, их количество и стоимость
4. Player – класс в котором храниться вся информация об игроке
5. GameMechanic – класс где правила описанные в другом класс(о нем будет сказано позже) соединяются в логику приложения.

Одна из задач класса сформировать поле такого вида:



Заполнение поля осуществляется с помощью функции которая вызывает три шаблона инициализации:

public void startGame(){

templateOne();

templateTwo();

templateThree();

}

И эти методы определяют каждую ячейку одним из состояний:

enum State{

DEF\_CELL, X2WORD, X2LETTER, X3WORD, X3LETTER,START\_POSITION

}

DEF\_CELL – свободная пустая ячейка

X2WORD – ячейка бонус: умножение слова на 2

X2LETTER – ячейка бонус: умножение буквы на 2

X3WORD – ячейка бонус: умножение слова на 3

X3LETTER – ячейка бонус: умножение буквы на 3

START\_POSITION – центральная стартовая клетка .

1. GameRules – основной класс где описана вся логика игры – содержит в себе методы создания и проверки корректности слова, а также сброса букв и восстановлением количества букв игрока после хода
2. WordChecker – класс отвечающийза взаимодействие с Yandex API

package name.mizunotlt.eruditkurs;

import org.json.simple.JSONArray;

import org.json.simple.JSONObject;

import org.json.simple.parser.JSONParser;

import org.json.simple.parser.ParseException;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.MalformedURLException;

import java.net.URL;

import javax.net.ssl.HttpsURLConnection;

import static java.net.HttpURLConnection.HTTP\_OK;

interface WordCheckerInterface {

String KEY = "dict.1.1.20181002T210630Z.48eeb6b8a8579498.d722e12836ce5ecfbd5ff8822c5bfd2088b1723d";

String COMMAND = "lookup";

String LANG = "ru-ru";

String urlApi = "https://dictionary.yandex.net/api/v1/dicservice.json/";

boolean checkCorrectWord(JSONObject jsobj);

}

class WordChecker implements WordCheckerInterface {

private String word;

WordChecker(String word){

this.word = word;

}

private final JSONParser jsonParser = new JSONParser();

public URL buildUrl() throws MalformedURLException {

String str = urlApi +

COMMAND +

"?key=" +

KEY +

"&lang=" +

LANG +

"&text=" +

this.word;

return new URL(str);

}

public JSONObject connectYandexDicApi(URL forApi) throws IOException {

JSONObject jsobj = new JSONObject();

HttpsURLConnection conn = (HttpsURLConnection) forApi.openConnection();

conn.setRequestMethod("POST");

conn.setDoOutput(true);

conn.setDoInput(true);

BufferedReader in = new BufferedReader(

new InputStreamReader(conn.getInputStream()));

if (conn.getResponseCode() == HTTP\_OK) {

try {

try {

jsobj = (JSONObject) jsonParser.parse(in.readLine());

} catch (ParseException e) {

e.printStackTrace();

}

} catch (IOException e) {

return null;

}

} else {

return null;

}

return jsobj;

}

@Override

public boolean checkCorrectWord(JSONObject jsobj) {

boolean result = false;

JSONArray jsArray = (JSONArray)jsobj.get("def");

if (jsArray.size() == 0)

return false;

JSONObject obj = (JSONObject) jsArray.get(0);

Object text = obj.get("text");

String pos = obj.get("pos").toString();

if ((text != null) && (pos.toLowerCase().equals("noun"))){

result = true;

}

return result;

}

}

На этом рассмотрение основных методов использованных для реализации приложения заканчиваем и переходим к ресурсам

**3. РЕСУРСЫ:**

В AndroidStudio ресурсы содержатся в специальной папке *res.* В моём проекте были добавлены такие ресурсы, как:

* layout-ы : (*res – layout)*

а) *activity\_main.xml:*

б) *activity\_game xml:*

в) activity\_game\_start

*г) activity\_about*

*д)* (*res – layout-land –* *activity\_game.xml):*

*е)* (*res – layout-land –* *activity\_main.xml):*

* строки (*res – values – strings):*

<resources>

<string name="app\_name">EruditKurs</string>

<string name="title\_activity\_start\_game">EruditKurs</string>

<string name="button">Button</string>

<string name="newGame">Новая игра</string>

<string name="exit">Выход</string>

<string name="aboutButton">Об Игре</string>

<string name="startGame">Старт игры</string>

<string name="nextTurn">Конец хода</string>

<string name="newWord">Новое слово</string>

<string name="about">Игра эрудит!!

Версия игры: 1.0

Автор: Архиреев Дмитрий

P.S ОЧЕНЬ УСТАЛ</string>

</resources>

* Размеры

<resources>

<dimen name="fab\_margin">16dp</dimen>

<dimen name="gamePortraitTextViewWidthHeight">80dp</dimen>

<dimen name="gamePortraitMarginBottom">144dp</dimen>

<dimen name="gamePortraitMarginEndStart">8dp</dimen>

<dimen name="gamePortraitButtonWidth">145dp</dimen>

<dimen name="gamePortraitButtonHeight">40dp</dimen>

<dimen name="gamePortraitButtonRecycleWidth">45dp</dimen>

<dimen name="gamePortraitButtonRecycleHeight">55dp</dimen>

<dimen name="gamePortraitMarginBottomNextTurn">188dp</dimen>

<dimen name="gamePortraitMarginEndNextTurn">8dp</dimen>

<dimen name="gamePortraitMarginStartNextTurn">252dp</dimen>

<dimen name="gamePortraitMarginBottomNewWord">124dp</dimen>

<dimen name="gamePortraitMarginEndNewWord">8dp</dimen>

<dimen name="gamePortraitMarginStartNewWord">8dp</dimen>

<dimen name="gamePortraitMarginBottomStartGame">84dp</dimen>

<dimen name="gamePortraitMarginEndStartGame">8dp</dimen>

<dimen name="gamePortraitMarginStartStartGame">252dp</dimen>

<dimen name="gamePortraitMarginBottomRecycle">8dp</dimen>

<dimen name="gamePortraitMarginEndRecycle">8dp</dimen>

<dimen name="gamePortraitMarginStartRecycle">8dp</dimen>

<dimen name="gameLandTextViewWidthHeight">80dp</dimen>

<dimen name="gameLandTextViewMarginBottom">76dp</dimen>

<dimen name="gameLandTextViewMarginEndStart">8dp</dimen>

<dimen name="gameLandMarginBottom">144dp</dimen>

<dimen name="gameLandMarginEndStart">8dp</dimen>

<dimen name="gameLandButtonWidth">145dp</dimen>

<dimen name="gameLandButtonHeight">40dp</dimen>

<dimen name="gameLandButtonRecycleWidth">45dp</dimen>

<dimen name="gameLandButtonRecycleHeight">55dp</dimen>

<dimen name="gameLandMarginBottomNextTurn">16dp</dimen>

<dimen name="gameLandMarginEndNextTurn">8dp</dimen>

<dimen name="gameLandMarginStartNextTurn">252dp</dimen>

<dimen name="gameLandMarginBottomNewWord">120dp</dimen>

<dimen name="gameLandMarginEndNewWord">8dp</dimen>

<dimen name="gameLandMarginStartNewWord">8dp</dimen>

<dimen name="gameLandMarginBottomStartGame">68dp</dimen>

<dimen name="gameLandMarginEndStartGame">8dp</dimen>

<dimen name="gameLandMarginStartStartGame">252dp</dimen>

<dimen name="gameLandMarginBottomRecycle">16dp</dimen>

<dimen name="gameLandMarginEndRecycle">8dp</dimen>

<dimen name="gameLandMarginStartRecycle">8dp</dimen>

<dimen name="mButtonWidth">150dp</dimen>

<dimen name="mButtonHeight">55dp</dimen>

<dimen name="mMarginStart">8dp</dimen>

<dimen name="mMarginEnd">8dp</dimen>

<dimen name="mMarginTop">8dp</dimen>

<dimen name="mMarginBottom">8dp</dimen>

<dimen name="mLandMarginStart">8dp</dimen>

<dimen name="mLandMarginEnd">8dp</dimen>

<dimen name="mLandMarginTop">240dp</dimen>

<dimen name="mLandMarginBottom">8dp</dimen>

<dimen name="textEditMargin">8dp</dimen>

<dimen name="gameStartButtonWidthStartGame">150dp</dimen>

<dimen name="gameStartButtonHeightStartGame">55dp</dimen>

<dimen name="gameStartMarginStartGame">8dp</dimen>

<dimen name="textEditMarginBottomSecond">80dp</dimen>

<dimen name="textEditMarginBottomFirst">160dp</dimen>

</resources>

* изображения (*res – drawable). (см гитхаб)*

**4. Тестирование и инспектор**

Для того чтобы проверить корректность работы моего приложения, мной были написаны соответствующие Unit и Android тесты.

В классе *EruditKursInstrumentedTest* тестируется корректность работы самой игры. Здесь тестируется корректная и некорректная постановка слова на поле, а так же проверяется правильно ли работает проверка слова. Так проверено замена букв игрока.

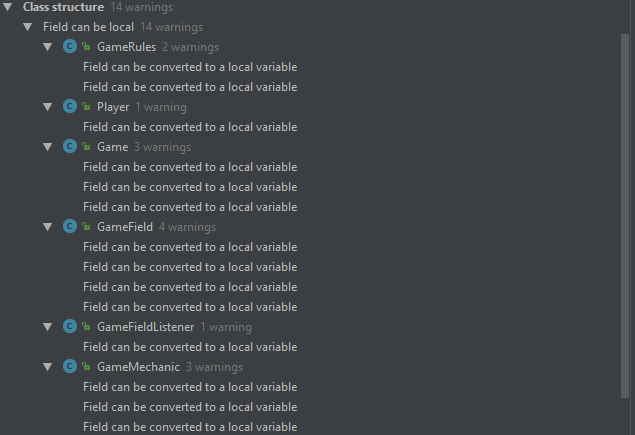
В классе *MainActivityInstrumentalTest* я тестирую работу главного меню, а именно переход на другие активити

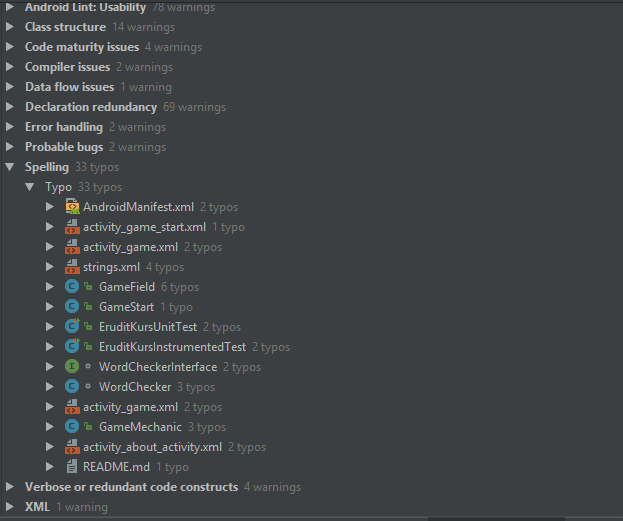
В классе StartGameActivityInstrumentalTest – проверяется проверка заполения двух тестовых полей и корректного нажатия на кнопку

В классе EruditKursUnitTest проиходит проверка основных методов бизнес логики приложения

Все тесты успешно проходят, что подтверждает корректность работы написанной программы.

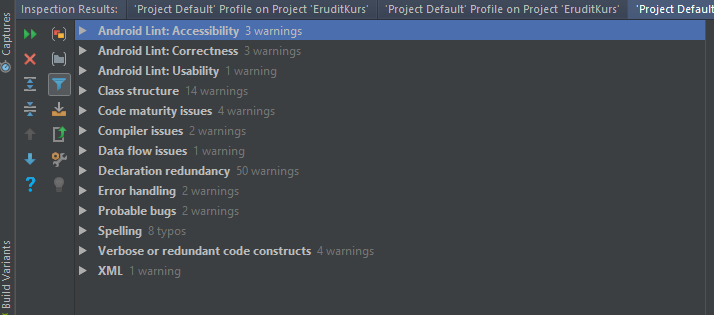
После написания тестов мной был запущен анализатор кода, который отобразил все нарушения, которые есть в моем коде. Приведу некоторые из них





В основном анализатор предупреждал об неиспользованных переменных, опечатках, изменении структуры класса, и в целом упрощении кода.

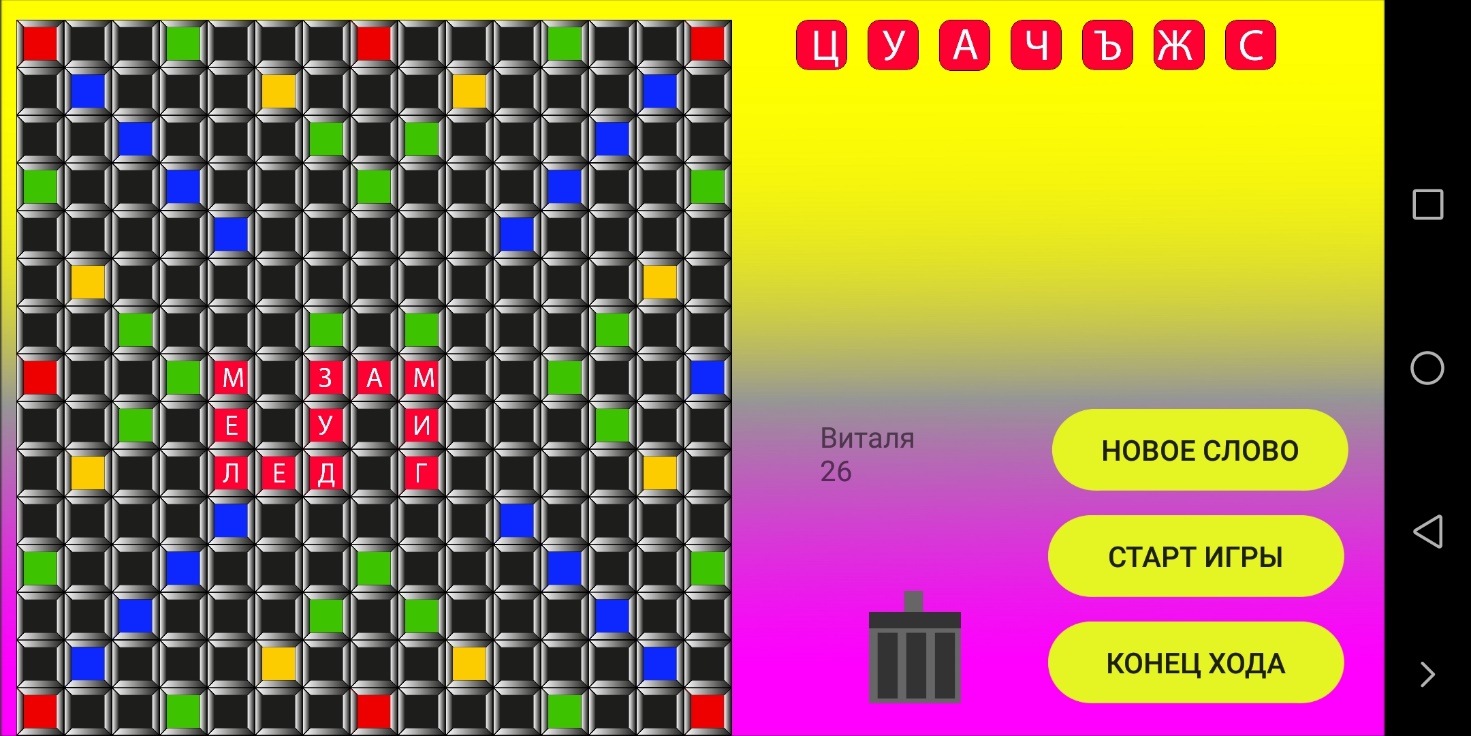
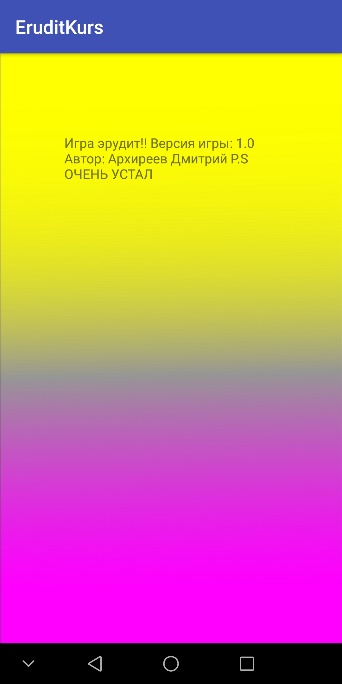
После упрощения кода, и исправления ошибок результат работы анализатора такой



Количество предупреждения стало существенно меньше

1. **Результат работы**

В итоге моё приложение, запущенное на устройстве, выглядит следующим образом:



**ЗАКЛЮЧЕНИЕ**

В данном курсовом проекте мной была игра «Эрудит» для ОС Android. В ходе выполнения я ознакомился со средой разработки Android Studio, и научился проектировать и разрабатывать мобильные приложения. Так же были получены навыки тестирования приложения, и в очередной раз отточены навыки тестирования бизнес-логики. Цель данной курсовой работы, поставленная мной, я считаю была достигнута.

Полный программный код проекта выложен в соответствующем репозитории на GitHub:

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

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. КСПТ: Курс «Программирование под Android». // http://kspt.icc.spbstu.ru. Дата последнего обновления: 06.03.2017.

URL: http://kspt.icc.spbstu.ru/course/android

1. Espresso bacics. // developer.android.com. Дата последнего обновления: 20.12.2018.

URL: https://developer.android.com/training/testing/espresso/basics

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

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