

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ПЕТРА ВЕЛИКОГО

Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчет
по курсовой работе
«SudOKu»

Дисциплина
«Программирование под Андроид»

выполнила:
Васильева В.В
группа: 33531/2
преподаватель:
Кузнецов А.Н.

Санкт-Петербург

2018

Введение	3
Реализация	4
Активити	4
Java классы	6
Другие ресурсы	19
Трудности	19
Инспекция кода	20
Тесты	20
Ссылка на проект	21
Выводы	21

Введение

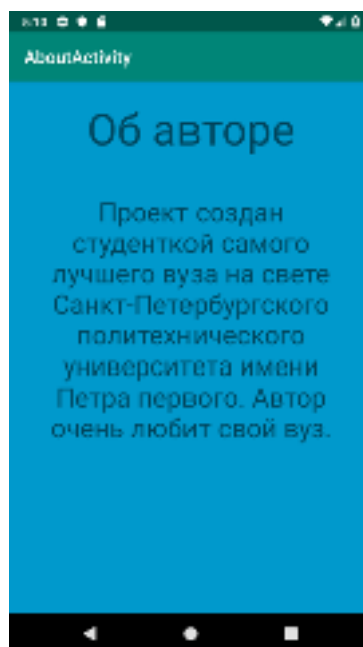
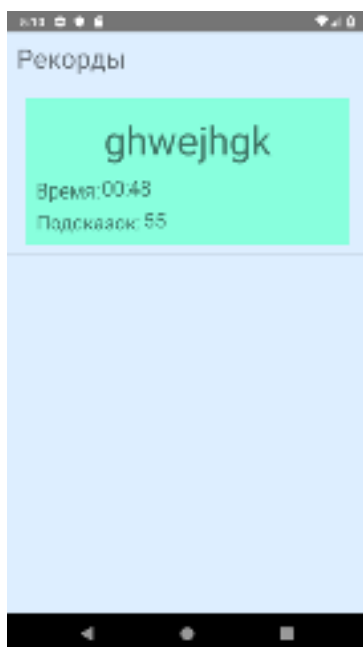
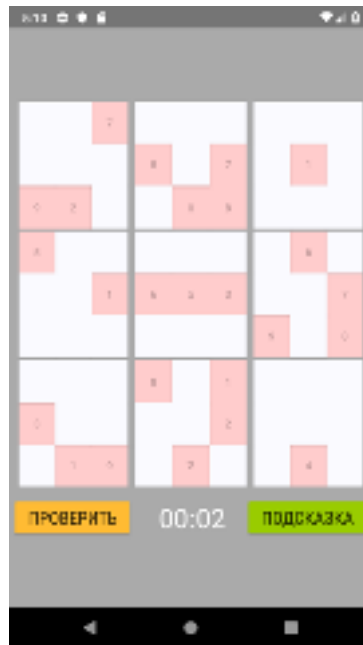
В данной работе с помощью среды разработки Android Studio реализован проект в формате приложения (игры) для смартфона. Тема проекта - sudoku. Основной язык, используемый для разработки - Java.

Реализация

Активности

Приложение состоит из четырех активити:

- меню игры,
- сама игра,
- таблица рекордов,
- страничка с информацией об авторе;



Рассмотрим содержание самого сложного активити - активити игры

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/darker_gray">
```

```

tools:context=".GameActivity"
android:id="@+id/activity_game_portrait">

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">

    <GridLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:id="@+id/a_game_sudokuGrid"
        android:layout_margin="10dp">
    </GridLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center">
        <Button
            android:id="@+id/a_game_check"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="2dp"
            android:backgroundTint="@android:color/holo_orange_light"
            android:fontFamily="sans-serif-condensed"
            android:layout_weight="2"
            android:text="@string/checkButton"
            android:textSize="20sp" />
        <Space
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="4"/>
        <Chronometer
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="30sp"
            android:textColor="@android:color/white"
            android:id="@+id/a_game_chrono"
            android:layout_weight="0"
            android:gravity="center"/>
        <Space
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="3"/>
        <Button
            android:id="@+id/a_game_help"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="2dp"
            android:backgroundTint="@android:color/holo_green_light"
            android:fontFamily="sans-serif-condensed"
            android:layout_weight="2"
            android:text="@string/tip"
            android:textSize="20sp"/>
    </LinearLayout>

```

В основе лежит `frameLayout`, в котором находятся два `linearLayout`-а. Первый отображает экран загрузки в то время, как логическая часть приложения генерируется. После генерации этот `layout` становится прозрачным и пользователь видит второй `layout`. На нем уже располагаются поле sudoku, расположенное в `gridLayout`, кнопки и таймер (`chronometer`). Кнопки и хронометр лежат в отдельном `linearLayout`-е и разделены объектом `Space`.

Для `landscape` ориентации экрана существует отдельная активность.



Устроено оно очень похоже на активность для `portrait` ориентации за исключением того, что основной `linearLayout` имеет горизонтальную ориентацию и благодаря этому кнопки с таймером находятся не под полем судаку, а справа от него. Таким же образом изменение ориентации `layout`-а, в котором лежат кнопки и хронометр, позволяет отобразить их вертикально относительно друг-друга, а не горизонтально.

Java классы

MainMenuActivity

```
public class MainMenuActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main_menu);

        Button startButton = findViewById(R.id.a_menu_PLAY);
        startButton.setOnClickListener(v -> goToGame());

        Button recButton = findViewById(R.id.a_menu_RECORDS);
        recButton.setOnClickListener(v -> goToRecords());

        Button aboutButton = findViewById(R.id.a_menu_ABOUT);
        aboutButton.setOnClickListener(v -> goToAbout());

        Button exitButton = findViewById(R.id.a_menu_EXIT);
        exitButton.setOnClickListener(v -> exit());
    }
}
```

```

private void exit() {
    finish();
}

private void goToAbout() {
    Intent intent = new Intent(this, AboutActivity.class);
    startActivity(intent);
}

private void goToRecords() {
    Intent intent = new Intent(this, RecordsActivity.class);
    startActivity(intent);
}

private void goToGame() {
    Intent intent = new Intent(this, GameActivity.class);
    startActivity(intent);
}
}

```

Данный класс представляет собой реализацию главного меню игры. В методе onCreate на к каждой кнопке экрана добавляется onClickListener. При нажатии на кнопку вызывается соответствующий этой кнопке метод, который перенаправляет приложение в новое активити.

AboutActivity

```

public class AboutActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_about);
    }
}

```

Простой класс с установкой layout.

RecordsActivity

```

public class RecordsActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_records);

        List<Record> records = DBConnector.getAllRecords(this);
        if (records != null){
            RecordAdapter adapter = new RecordAdapter(this, records);

            ListView lvMain = findViewById(R.id.a_recs_listView);
            lvMain.setAdapter(adapter);
        }
    }
}

```

Класс для отображения рекордов.

DataBaseConnector

```
public class DBConnector {

    public static void addRecord(Context context, Record record) {
        DBHelper helper = new DBHelper(context);
        SQLiteDatabase database = helper.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(KEY_NAME, record.getName());
        values.put(KEY_TIME, record.getTime());
        values.put(KEY_TIPS, record.getTips());
        database.insert(TABLE_NAME, null, values);
        database.close();
        helper.close();
    }

    public static List<Record> getAllRecords(Context context) {
        DBHelper helper = new DBHelper(context);
        SQLiteDatabase database = helper.getReadableDatabase();

        Cursor cursor = database.query(TABLE_NAME, null, null, null, null, null, KEY_TIME + " DESC");
        int indexName = cursor.getColumnIndex(KEY_NAME);
        int indexTime = cursor.getColumnIndex(KEY_TIME);
        int indexTips = cursor.getColumnIndex(KEY_TIPS);

        if (cursor.moveToFirst()) {
            List<Record> records = new ArrayList<>();
            do {
                records.add(new Record(
                    cursor.getString(indexName),
                    cursor.getString(indexTime),
                    cursor.getInt(indexTips)
                ));
            } while (cursor.moveToNext());
            database.close();
            helper.close();
            return records;
        }
        database.close();
        helper.close();
        return null;
    }
}
```

Класс, реализующих взаимодействие с базой данных SQLite. В первом методе реализовано добавление одного рекорда. Во втором методе все рекорды возвращаются в виде списка.

DataBase Helper

```
public class DBHelper extends SQLiteOpenHelper {

    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "Slava_sudoku_records";
    public static final String TABLE_NAME = "Sudoku_Records";
    public static final String KEY_ID = "_id";
    public static final String KEY_NAME = "name";
    public static final String KEY_TIME = "time";
    public static final String KEY_TIPS = "tips";
```



```

public DBHelper(@Nullable Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
}

@Override
public void onCreate(SQLiteDatabase db) {
    String sql = "create table " + TABLE_NAME + "(" +
        KEY_ID + " integer primary key autoincrement," +
        KEY_NAME + " text not null," +
        KEY_TIME + " text not null," +
        KEY_TIPS + " integer not null" + ")";
    db.execSQL(sql);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    String sql = "drop table if exists " + TABLE_NAME;
    db.execSQL(sql);
    onCreate(db);
}
}

```

Класс, который необходимо реализовать для работы с базой данных SQLite. В этом классе необходимо переопределить два метода onCreate и onUpgrade. В первом описана структура базы данных при ее создании, а во втором ее обновление.

Record

```

class Record {
    private final String name;
    private final String time;
    private final int tips;

    public Record(String name, String time, int tips) {
        this.name = name;
        this.time = time;
        this.tips = tips;
    }

    public String getName() {
        return name;
    }

    public String getTime() {
        return time;
    }

    public int getTips() {
        return tips;
    }
}

```

Класс для удобного хранения и использования рекордов.

RecordAdapter

```

public class RecordAdapter extends BaseAdapter {

    Context ctx;
    LayoutInflater lInflater;
    List<Record> objects;
}

```

```

RecordAdapter(Context context, List<Record> Records) {
    ctx = context;
    objects = Records;
    inflater = (LayoutInflater) ctx
        .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
}

@Override
public int getCount() {
    return objects.size();
}

@Override
public Object getItem(int position) {
    return objects.get(position);
}

@Override
public long getItemId(int position) {
    return position;
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {
    View view = convertView;
    if (view == null) {
        view = inflater.inflate(R.layout.list_item_record, parent, false);
    }

    Record record = getRecord(position);
    ((TextView) view.findViewById(R.id.recordName)).setText(record.getName());
    ((TextView) view.findViewById(R.id.recordTime)).setText(record.getTime());
    ((TextView) view.findViewById(R.id.recordTips)).setText(String.valueOf(record.getTips()));

    return view;
}

Record getRecord(int position) {
    return ((Record) getItem(position));
}
}

```

Класс используется для удобного отображения рекордов в listView. Рекорды в нем хранятся в списке.

GameActivity

```

public class GameActivity extends AppCompatActivity {
    boolean loaded = false;
    Chronometer chronometer;
    List<SudokuButton> gameButtons;
    Sudoku game;
    Button check;
    Button help;
    int tips = 0;
    GridLayout gameGrid;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_game);
    }
}

```

```

gameGrid = findViewById(R.id.a_game_sudokuGrid);
chronometer = findViewById(R.id.a_game_chrono);
chronometer.stop();

LinearLayout loading = findViewById(R.id.a_game_loadingScreen);

check = findViewById(R.id.a_game_check);
check.setOnClickListener(v -> checkGame());

help = findViewById(R.id.a_game_help);
help.setOnClickListener(v -> help());

if (savedInstanceState == null || !savedInstanceState.containsKey("sudokuGame")) {
    loading.setAlpha(1f);
    generateNewStuff();
} else {
    loaded = true;
    loading.setAlpha(0f);
    loadSavedGame(savedInstanceState);
}
}

@Override
protected void onSaveInstanceState(Bundle saveState) {
    super.onSaveInstanceState(saveState);

    if (!loaded)
        return;

    saveState.putSerializable("sudokuGame", game);
    saveState.putLong("chronoBase", chronometer.getBase());
    saveState.putInt("tipsCounter", tips);
}

private void loadSavedGame(Bundle savedInstanceState) {
    game = (Sudoku) savedInstanceState.getSerializable("sudokuGame");
    tips = savedInstanceState.getInt("tipsCounter");
    chronometer.setBase(savedInstanceState.getLong("chronoBase"));
    fillGrid();
    chronometer.start();
}

private void help() {
    List<Integer> notFinished = new ArrayList<>();
    for (int i = 0; i < 81; i++) {
        if (game.getNumber(i % 9, i / 9) == 0) {
            notFinished.add(i);
        }
    }
    if (notFinished.isEmpty())
        return;
    Collections.shuffle(notFinished);
    int helpingIndex = notFinished.get(0);
    int solution = game.getSolutionForHelp(helpingIndex % 9, helpingIndex / 9);
    gameButtons.get(helpingIndex).setHelped(solution);
    tips++;
}

```

```

private void generateNewStuff() {
    AsyncTask generatingMap = new AsyncTask() {
        @Override
        protected Object doInBackground(Object[] objects) {
            generateAsync();
            return null;
        }

        @Override
        protected void onPostExecute(Object o) {
            super.onPostExecute(o);
            notifyGameCreated();
        }
    };
    generatingMap.execute();
}

private void generateAsync() {
    game = new Sudoku();
    SudokuButton.game = game;
}

private void checkGame() {
    for (int i = 0; i < 81; i++) {
        if (!checkPoint(i % 9, i / 9))
            return;
    } win();
}

private void win() {
    chronometer.stop();
    AlertDialog.Builder alertDialog = new AlertDialog.Builder(this);
    alertDialog.setTitle("Поздравляем!");
    alertDialog.setMessage("Введите имя игрока:");
    final EditText input = new EditText(this);
    LinearLayout.LayoutParams lp = new LinearLayout.LayoutParams(
        LinearLayout.LayoutParams.MATCH_PARENT,
        LinearLayout.LayoutParams.MATCH_PARENT);
    lp.setMargins(20, 20, 20, 20);
    input.setLayoutParams(lp);
    alertDialog.setView(input);
    alertDialog.setPositiveButton("Готово", (dialog, which) -> {
        int tips = this.tips;
        String name = input.getText().length() == 0 ? "[Игрок]" : String.valueOf(input.getText());
        String time = String.valueOf(chronometer.getText());
        Record record = new Record(name, time, tips);
        DBConnector.addRecord(getApplicationContext(), record);
        finish();
    });
    alertDialog.show();
}

private boolean checkPoint(int x, int y) {
    if (game.getNumber(x, y) == 0) {
        makeToast("Не все поля заполнены!");
        return false;
    }
    if (!game.isCheckValid(x, y)) {
        makeToast("Не все поля корректные!");
        return false;
    }
    return true;
}

```

```

private void makeToast(String msg) {
    Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();
}

private void fillGrid() {

    gameButtons = new ArrayList<>();
    gameGrid.setColumnCount(11);

    int counter = 0;
    int second_counter = 0;

    for (int i = 0; i < 81; i++) {
        SudokuButton button = new SudokuButton(this, i, 0);
        int actualy = game.getNumber(i % 9, i / 9);
        if (actualy != 0) {
            if (game.isHelped(i % 9, i / 9))
                button.setHepled(actualy);
            else if (game.isInitial(i % 9, i / 9)) {
                button.setContent(actualy);
                button.makeStatic();
            } else
                button.setContent(actualy);
        }
        gameButtons.add(button);
        gameGrid.addView(button);
        counter++;
        second_counter++;
        if (second_counter == 27 && !(i > 60)) {
            for (int j = 0; j < 11; j++) {
                Space space = new Space(this);
                GridLayout.LayoutParams doubleLayoutParams = new GridLayout.LayoutParams();
                doubleLayoutParams.width = 0;
                doubleLayoutParams.height = 8;
                doubleLayoutParams.rowSpec = GridLayout.spec(GridLayout.UNDEFINED, 1f);
                doubleLayoutParams.columnSpec = GridLayout.spec(GridLayout.UNDEFINED, 1f);
                space.setLayoutParams(doubleLayoutParams);
                gameGrid.addView(space);
            }
            counter = 0;
            second_counter = 0;
            continue;
        }
        if (counter == 3 || counter == 6) {
            Space space = new Space(this);
            GridLayout.LayoutParams doubleLayoutParams = new GridLayout.LayoutParams();
            doubleLayoutParams.width = 0;
            doubleLayoutParams.rowSpec = GridLayout.spec(GridLayout.UNDEFINED, 1f);
            doubleLayoutParams.columnSpec = GridLayout.spec(GridLayout.UNDEFINED, 1f);
            space.setLayoutParams(doubleLayoutParams);
            gameGrid.addView(space);
        }
        if (counter == 9)
            counter = 0;
    }
}

```

```

public void notifyGameCreated() {
    loaded = true;
    fillGrid();
    LinearLayout loading = findViewById(R.id.a_game_loadingScreen);
    Animation fadeOut = new AlphaAnimation(1, 0);
    fadeOut.setDuration(1000);
    fadeOut.setAnimationListener(new Animation.AnimationListener() {
        @Override
        public void onAnimationStart(Animation animation) {
            //nothing
        }

        @Override
        public void onAnimationRepeat(Animation animation) {
            //nothing
        }

        @Override
        public void onAnimationEnd(Animation animation) {
            loading.setAlpha(0f);
        }
    });
    loading.setAnimation(fadeOut);
    fadeOut.start();
    chronometer.setBase(SystemClock.elapsedRealtime());
    chronometer.start();
}
}

```

Основной класс игры. В методе onCreate происходит проверка: существует ли сохраненное состояние у игры. Это позволяет не потерять информацию при смене ориентации девайса. Само сохранение состояния происходит в методе onSaveInstanceState. Если сохраненных состояний у приложения нет, то экран загрузки становится непрозрачным и начинается генерация поля sudoku. Для генерации используется AsyncTask. Логика генерации поля sudoku прописана в классе Sudoku. Когда генерация поля завершена вызывается метод notifyGameCreated. В нем экран загрузки становится прозрачным, запускается таймер и вызывается метод fillGrid. В этом методе gridView заполняется кастомными кнопками. Метод checkGame вызывается при нажатии на соответствующую кнопку. Если не все поля заполнены или заполнены, но не верно, то вызывается метод makeToast с соответствующим сообщением. Если же все поля заполнены верно, то вызывается метод win. В этом методе останавливается таймер и «собирается» всплывающее окно, в котором предлагается ввести имя игрока, из данных собирается рекорд и кладется в базу данных.

SudokuButton

```

public class SudokuButton extends android.support.v7.widget.AppCompatButton {

    public static Sudoku game;
    boolean isConstant = false;
    int value;
    final int position;

    public SudokuButton(Context context, int position, int initValue) {
        super(context);
        this.position = position;

        GridLayout.LayoutParams doubleLayoutParams = new GridLayout.LayoutParams();
        doubleLayoutParams.width = 0;
    }
}

```

```

        if (context.getResources().getConfiguration().orientation == ORIENTATION_LANDSCAPE)
            doubleLayoutParams.height = 0;
        doubleLayoutParams.rowSpec = GridLayout.spec(GridLayout.UNDEFINED, 5f);
        doubleLayoutParams.columnSpec = GridLayout.spec(GridLayout.UNDEFINED, 5f);
        doubleLayoutParams.setGravity(Gravity.FILL);
        setLayoutParams(doubleLayoutParams);

        this.setTextSize(TypedValue.COMPLEX_UNIT_SP, 12);
        setBackground(getResources().getDrawable(R.drawable.button_background,
getContext().getTheme()));
        setValue(initValue);
        setOnClickListener(v -> clicked());
    }

    private void updateContent() {
        if (value == 0)
            setText("");
        else
            setText(String.valueOf(value)); }

    private void clicked() {
        if (isConstant)
            return;
        setValue((value + 1) % 10);
        game.setNumber(position % 9, position / 9, value);
    }

    private void setValue(int value) {
        this.value = value;
        updateContent();
    }

    public void makeStatic() {
        isConstant = true;
        setEnabled(false);
    }

    public void setHepled(int solution) {
        setValue(solution);
        makeStatic();
        setBackground(getResources().getDrawable(R.drawable.button_hepled, getContext().getTheme()));
    }

    public void setContent(int actualy) {
        setValue(actualy);
    }
}

```

Кастомная кнопка, которой задаются layout параметры. Прописан механизм смены значения при нажатии на кнопку. Прописаны состояния, если кнопка является подсказкой и если она была «открыта» при генерации поля.

Sudoku

```

public class Sudoku implements Serializable {
    private int[][] solution;    // Generated solution.
    private int[][] game;       // Generated game with user input.
    private boolean[][] helped; // If cell is true, this one was a hint
    private boolean[][] initial; // If true, the cell was opened at beginning
    private boolean[][] check;  // Holder for checking validity of game.
    private boolean help;       // Help turned on or off.
}

```

```

public Sudoku() {
    check = new boolean[9][9];
    helped = new boolean[9][9];
    initial = new boolean[9][9];
    newGame();
    helped[3][3] = false;
    help = true;
}

public Sudoku(int[][] solution, int[][] game) {
    this.solution = solution;
    this.game = game;
    helped = new boolean[9][9];
    initial = new boolean[9][9];
    helped[3][3] = false;
    help = true;
}

public void newGame() {
    solution = generateSolution(new int[9][9], 0);
    game = generateGame(copy(solution));
    for (int i = 0; i < 81; i++){
        if (game[i % 9][i / 9] != 0)
            initial[i % 9][i / 9] = true;
    }
}

public void checkGame() {
    for (int y = 0; y < 9; y++) {
        for (int x = 0; x < 9; x++)
            check[y][x] = game[y][x] == solution[y][x];
    }
}

public void setNumber(int x, int y, int number) {
    game[y][x] = number;
    checkGame();
}

public int[][] getGame(){
    return game;
}

public int[][] getSolution(){
    return solution;
}

public int getNumber(int x, int y) {
    return game[y][x];
}

public boolean isCheckValid(int x, int y) {
    return check[y][x];
}

```



```

boolean isPossibleX(int[][] game, int y, int number) {
    for (int x = 0; x < 9; x++) {
        if (game[y][x] == number)
            return false;
    }
    return true;
}

boolean isPossibleY(int[][] game, int x, int number) {
    for (int y = 0; y < 9; y++) {
        if (game[y][x] == number)
            return false;
    }
    return true;
}

boolean isPossibleBlock(int[][] game, int x, int y, int number) {
    int x1 = x < 3 ? 0 : x < 6 ? 3 : 6;
    int y1 = y < 3 ? 0 : y < 6 ? 3 : 6;
    for (int yy = y1; yy < y1 + 3; yy++) {
        for (int xx = x1; xx < x1 + 3; xx++) {
            if (game[yy][xx] == number)
                return false;
        }
    }
    return true;
}

private int getNextPossibleNumber(int[][] game, int x, int y, List<Integer> numbers) {
    while (numbers.size() > 0) {
        int number = numbers.remove(0);
        if (isPossibleX(game, y, number) && isPossibleY(game, x, number) && isPossibleBlock(game, x, y,
number))
            return number;
    }
    return -1;
}

private int[][] generateSolution(int[][] game, int index) {
    if (index > 80)
        return game;

    int x = index % 9;
    int y = index / 9;

    List<Integer> numbers = new ArrayList<Integer>();
    for (int i = 1; i <= 9; i++) numbers.add(i);
    Collections.shuffle(numbers);

    while (numbers.size() > 0) {
        int number = getNextPossibleNumber(game, x, y, numbers);
        if (number == -1)
            return null;

        game[y][x] = number;
        int[][] tmpGame = generateSolution(game, index + 1);
        if (tmpGame != null)
            return tmpGame;
        game[y][x] = 0;
    }
}

```

```

private int[][] generateGame(int[][] game) {
    List<Integer> positions = new ArrayList<Integer>();
    for (int i = 0; i < 81; i++)
        positions.add(i);
    Collections.shuffle(positions);
    return generateGame(game, positions);
}

private int[][] generateGame(int[][] game, List<Integer> positions) {
    while (positions.size() > 0) {
        int position = positions.remove(0);
        int x = position % 9;
        int y = position / 9;
        int temp = game[y][x];
        game[y][x] = 0;
        if (!isValid(game))
            game[y][x] = temp;
    } return game; }

private boolean isValid(int[][] game) { return isValid(game, 0, new int[] { 0 }); }

private boolean isValid(int[][] game, int index, int[] numberOfSolutions) {
    if (index > 80) return ++numberOfSolutions[0] == 1;
    int x = index % 9;
    int y = index / 9;
    if (game[y][x] == 0) {
        List<Integer> numbers = new ArrayList<Integer>();
        for (int i = 1; i <= 9; i++) numbers.add(i);
        while (numbers.size() > 0) {
            int number = getNextPossibleNumber(game, x, y, numbers);
            if (number == -1) break;
            game[y][x] = number;
            if (!isValid(game, index + 1, numberOfSolutions)) {
                game[y][x] = 0;
                return false; }
            game[y][x] = 0;
        } } else if (!isValid(game, index + 1, numberOfSolutions))
            return false;
    return true; }

private int[][] copy(int[][] game) {
    int[][] copy = new int[9][9];
    for (int y = 0; y < 9; y++) {
        for (int x = 0; x < 9; x++)
            copy[y][x] = game[y][x];
    } return copy; }

public int getSolutionForHelp(int x, int y) {
    setNumber(x, y, solution[y][x]);
    helped[y][x] = true;
    return solution[y][x];
}

public boolean isHelped(int x, int y) { return helped[y][x]; }

public boolean isInitial(int x, int y) { return initial[y][x]; }
}

```

Класс, отвечающий за логику игры.

Другие ресурсы

Большинство значений вынесены в папку values

Например цвета

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="colorPrimary">#008577</color>
  <color name="colorPrimaryDark">#00574B</color>
  <color name="colorAccent">#D81B60</color>

  <color name="black_overlay">#66000000</color>
  <color name="button_enabled">#FAFAFF</color>
  <color name="button_disabled">#FFCCCC</color>
  <color name="button_pressed">#74748e</color>
  <color name="button_helped">#CCFFCC</color>
  <color name="total_black">#000000</color>
  <color name="loading_text">#111119</color>
  <color name="loading_background">#FAEEFF</color>
  <color name="list_element">#88FFDD</color>
  <color name="list_background">#DDEEFF</color>
</resources>
```

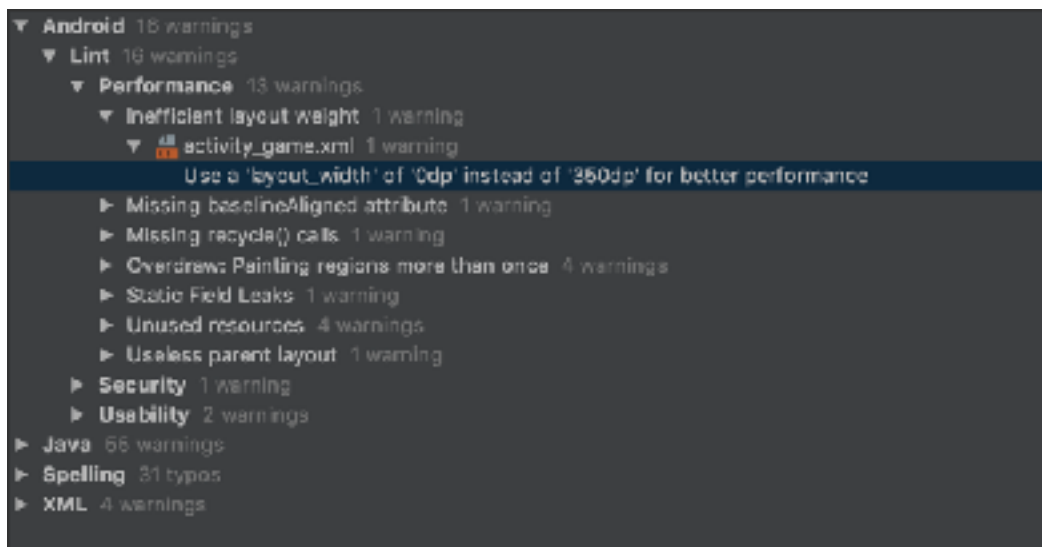
Строки

```
<resources>
  <string name="app_name">sudOKu</string>
  <string name="dummy_button">Dummy Button</string>
  <string name="dummy_content">DUMMY\nCONTENT</string>
  <string name="title_activity_game">GameActivity</string>
  <string name="title_activity_records">RecordsActivity</string>
  <string name="title_activity_about">AboutActivity</string>
  <string name="checkButton">Проверить</string>
  <string name="tip">Подсказка</string>
  <string name="load">Загрузка...</string>
  <string name="aboutAuthor">Об авторе</string>
  <string name="aboutActivityContent">Проект создан студенткой самого лучшего вуза на свете
  Санкт-Петербургского политехнического университета имени Петра первого. Автор очень любит
  свой вуз.</string>
  <string name="sudoku">sudOKu</string>
  <string name="play">Играть</string>
  <string name="records">Рекорды</string>
  <string name="exit">Выход</string>
  <string name="time">Время:</string>
  <string name="tipsCount">Подсказок:</string>
</resources>
```

Трудности

Изначально поле sudoku стояло не из кнопок, а из текстовых полей и для их заполнения использовалась экранная клавиатура. Это оказалось неудобным так как клавиатура занимала много места и ее приходилось постоянно прятать и вызывать.

Инспекция кода



Выполненные исправления:

- This 'Cursor' should be freed up after use with '#close()'
- Unused resources
- AllowBackup/FullBackupContent Problems
- Unnecessary semicolon
- " statement can be simplified
- Declaration access can be weaker
- Declaration can have final modifier
- Unused import
- Simplifiable JUnit assertion
- Explicit type can be replaced with \diamond
- Manual array copy
- XML tag empty body



Остальные предупреждения были незначительными.

Тесты

В проекте представлены два вида тестов: тесты логики и JUnit тесты.

Пример первого теста

```
assertFalse(sudoku.isPossibleX(sudoku.getGame(), 1, 3));
assertTrue(sudoku.isPossibleX(sudoku.getGame(), 1, 2));
```

проверка на возможность хода

Пример второго

```
onView(withId(R.id.a_menu_PLAY))  
    .perform(click());  
  
onView(withId(R.id.a_game_sudokuGrid))  
    .check(matches(isDisplayed()));
```

имитация нажатия и проверка на нахождение на экране

Ссылка на проект

<https://github.com/oxovu/AndroidProject>

Выводы

При выполнении данного курсового проекта были получены навыки работы со средой разработки AndroidStudio. Благодаря выполнению задания были изучены такие сопутствующие инструменты как: база данных, работа с поворотом экрана и альтернативными ресурсами, простейшая анимация, создание своих классов, наследующих стандартные андроид классы. Был получен навык написания JUnit тестов. Был изучен инструмент инспекции кода, позволяющий удобно исправлять предупреждения в коде.