Санкт-Петербургский Политехнический Университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Программирование

Отчет по курсовой работе «Шашки»

Студент:
Булгакова Маргарита Васильевна
23501/4
Преподаватель:
Вылегжанина К.Д.

Санкт-Петербург

2017

Оглавление

Игра «Шашки»

Задание

У нас имеется прямоугольное поле размером 8x8, которое окрашено в чёрный и белый цвет. На этом поле расставленны шашки. Два игрока по очереди передвигают свои шашки. Целью игры является уничтожение всех шашек соперника. У кого шашек не осталось, тот и считается проигравшим.

Правила работы программы

Всё действие происходит на поле размером 8x8. Во время партии каждому игроку принадлежат шашки одного цвета: чёрного или белого. Цель игры — лишить противника возможности хода путём взятия или запирания всех его шашек. Все шашки, участвующие в партии, выставляются перед началом игры на доску. Далее они передвигаются по полям доски и могут быть сняты с неё в случае боя шашкой противника.Брать шашку, находящуюся под боем, обязательно.Существует только два вида шашек: простые и дамки. В начале партии все шашки простые. Простая шашка может превратиться в дамку, если она достигнет последнего противоположного горизонтального ряда доски (дамочного поля).Простые шашки ходят только вперёд на следующее поле. Дамки могут ходить и вперёд и назад.

Концепция

Готовый проект должен моделировать игру между двумя игроками в шашки. Пользователь должен иметь возможность наблюдать за текущим состоянием игры, передвигать шашки и бить шашки противника.

Минимально работоспособный продукт

Минимально работоспособный продукт должен уметь: предоставить пользователю информацию о текущем состоянии игры, давать игроку возможность передвигать свои шашки, уничтожать шашки противника.

Диаграмма прецедентов использования

На основе разработанной концепции была составлена UML диаграмма прецедентов использования.(рис.1)

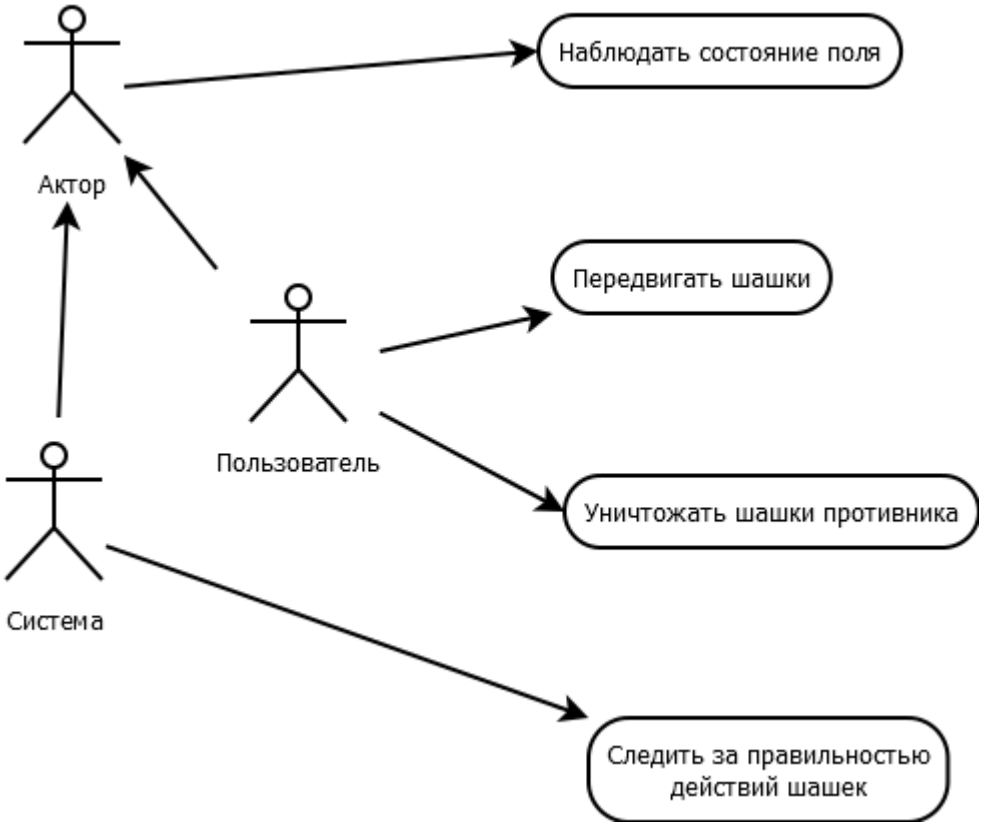

Рис.1 Диаграмма прецедентов использования

Проектирование приложения, реализующего игру «Шашки»

Компоненты

На основе анализа концепции и выделенных прецедентов использования было принято решение выделить два основных компонента, которые будут входить в состав продукта:

1. **Библиотека**
   При написании проекта, была создана библиотека. В ней находятся все необходимые классы для создания и работы игры. Один из классов (api) создан для предоставления всех действий над моделью.
2. **Графическое приложение**
   Графически визуализирует игровую модель, предоставляет пользователю графический интерфейс для взаимодействия с ней и выполнения остальный действий предусмотренных в реализации библиотеки.

Выводы

Таким образом, была разработана концепция приложения, что позволило определить внешний вид продукта и выделить его основные компоненты.

Реализация игры «Шашки»

Версия программ

Операционная система: Windows 7, среда разработки: Android Studio 2.3.3, компилятор: Gradle 3.0, Java 1.8.0.120.

Библиотека

Основные классы, выделенные в библиотеке

- Класс Draughts. Реализует шашку. Содержит координаты шашки, её цвет и тип. Присутствуют методы, возвращающие и задающие координаты и состояние шашки, проверяющий может ли данная шашка сделать заданный ход и ещё один метод, проверяющий достигла ли шашка конца поля.
- Класс Field. Класс представляет поле игры. Содержит двумерный массив клеток и цвет текущего хода. Присутствуют методы, возвращающие и задающие двумерном массиве и цвет текущего хода, также есть методы: проверяющий свободна ли данная клетка и проверяющий может ли она уничтожить данную шашку противника.
- Класс Api. Класс, предоставляющий все методы, доступные над игрой. Позволяет сделать ход шашкой, уничтожить шашку противника, сохранить поле в файл, загрузить поле из файла, получить данные о текущем состоянии поля и метод, узнающий может ли шашка уничтожить кого-нибудь вокруг себя

Андроид приложение

Андроид приложение позволяет играть на устройствах под управлением ОС Андроид.

- Класс About. Класс, который выводит информацию о правилах игры.
- Класс Draught. Главное меню приложение. В этом классе расположены кнопки: "Новая игра", "Помощь", "Выход".
- Класс DraughtView. Класс отвечающий за отрисовку поля. Все действия над полем происходят именно в этом классе.
- Класс Game. Класс, который вызывает отрисовку поля. Также он выводит сообщения о неправильных действиях игрока.

На рис.2 представлено главное окно приложения. В нём пользователю можно начать играть, открыть помощь или выйти.

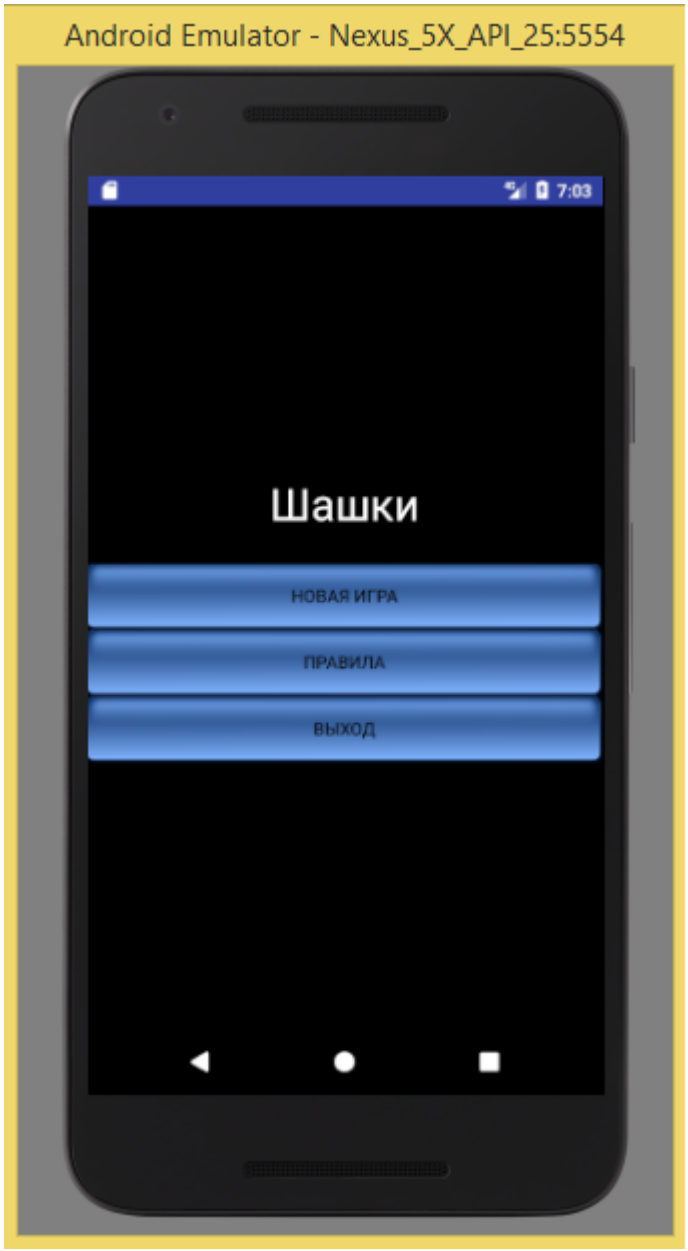

Рис. 2 Главное меню

На рис.3 – окно с полем. В центре расположено поле с шашками, внизу текущие данные об игре для каждой из сторон и информация о том, чей ход в данный момент.

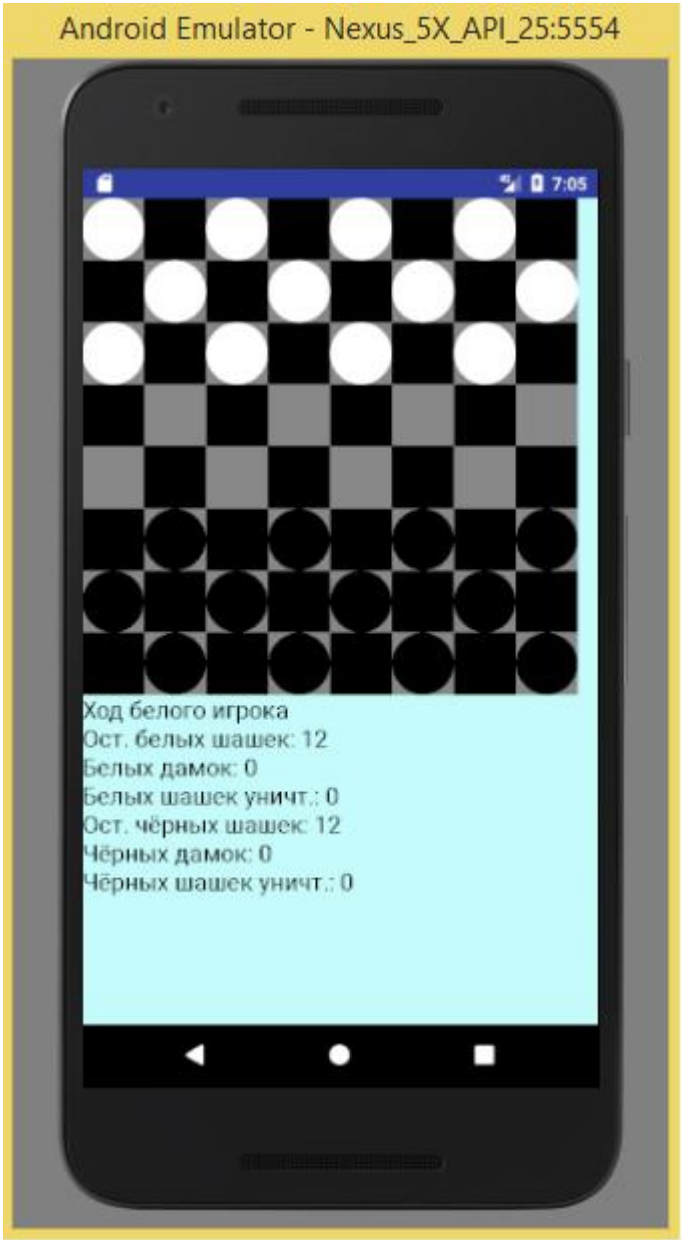

Рис.3 Игровое поле

На рис.4 – окно с правилами, по которым происходит игра Шашки.

Рис.4 Окно с правилами игры

Тестирование

Приложение тестировалось вручную. После добавления каждой новой функции в приложение, она была протестирована на работоспособность. После теста при возникновении ошибок они все были исправлены.

Вывод

По окончании семестра получены навыки написания программы на языке Java для операционной системы Android, делать графический интерфейс с помощью Swing, а также был получен опыт работы с большими проектами на Java, содержащими много классов и имеющих как консольное приложение, так и графическое.

Листинги

Библиотека

```
package com.example.rita.draughts;


import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class Api {

    public static void move_draught(Field field, int x_draught, int
y_draught, int x_place, int y_place) {
        Draughts draught = field.get_draught(x_draught, y_draught);
        if (check_destruction_around(field)){
            throw new IllegalArgumentException();
        }
        if (field.get_color() == draught.get_color()) {
            if (!draught.check_of_move(x_place, y_place)) {
                throw new IllegalArgumentException();
            } else {
                if (!field.check_free(x_place, y_place)) {
                    throw new IllegalArgumentException();
                } else {
                    field.set_null(x_draught, y_draught);
                    draught.set_x(x_place);
                    draught.set_y(y_place);
                    field.set_draught(draught);
                    if (draught.check_of_type()) {
                        draught.set_type(true);
                    }
                }
            }
        }else {
            throw new IllegalArgumentException();
        }
        field.set_color(!field.get_color());
    }

    public static void destroy_draught(Field field, int x_selected, int
y_selected, int x_destroyed, int y_destroyed) {
        Draughts draught = field.get_draught(x_selected, y_selected);
        if (field.get_color() == draught.get_color()) {
            if (!field.check_destruction(x_selected, y_selected, x_destroyed,
y_destroyed)) {
                throw new IllegalArgumentException();
            } else {
                if (!draught.get_type()) {
                    field.set_null(x_selected, y_selected);
                    field.set_null(x_destroyed, y_destroyed);
                    draught.set_x(x_selected + 2 * (x_destroyed -
x_selected));
                    draught.set_y(y_selected + 2 * (y_destroyed -
y_selected));
                    field.set_draught(draught);
                    draught.check_of_type();
                    if (draught.check_of_type()) {
```

```
                        draught.set_type(true);
                    }
                }else {
                    field.set_null(x_selected, y_selected);
                    field.set_null(x_destroyed, y_destroyed);
                    if ((x_destroyed - x_selected < 0) && (y_destroyed -
y_selected < 0)){
                        draught.set_x(x_destroyed - 1);
                        draught.set_y(y_destroyed - 1);
                        field.set_draught(draught);
                    }
                    if ((x_destroyed - x_selected < 0) && (y_destroyed -
y_selected > 0)){
                        draught.set_x(x_destroyed - 1);
                        draught.set_y(y_destroyed + 1);
                        field.set_draught(draught);
                    }
                    if ((x_destroyed - x_selected > 0) && (y_destroyed -
y_selected < 0)){
                        draught.set_x(x_destroyed + 1);
                        draught.set_y(y_destroyed - 1);
                        field.set_draught(draught);
                    }
                    if ((x_destroyed - x_selected > 0) && (y_destroyed -
y_selected > 0)){
                        draught.set_x(x_destroyed + 1);
                        draught.set_y(y_destroyed + 1);
                        field.set_draught(draught);
                    }
                }
            }
        }
        if (!check_destruction_around(field)) {
            field.set_color(!field.get_color());
        }
    }

    public static void save_file(Field field) throws IOException {
        File file = new File("savefile.txt");
        if(!file.exists()){
            file.createNewFile();
        }
        FileWriter wrt = new FileWriter(file);
        String lineSeparator = System.getProperty("line.separator");
        for (int i = 0; i <= 7; ++i){
            for (int j = 0; j <= 7; ++j){
                Draughts draught = field.get_draught(i,j);
                if (draught == null){
                    wrt.append("nl ");
                }else {
                    if ((draught.get_color()) && (!draught.get_type())) {
                        wrt.append("tf ");
                    }
                    if ((draught.get_color()) && (draught.get_type())) {
                        wrt.append("tt ");
                    }
                    if ((!draught.get_color()) && (!draught.get_type())) {
                        wrt.append("ff ");
                    }
                    if ((!draught.get_color()) && (draught.get_type())) {
                        wrt.write("ft ");
                    }
                }
```

```
            }
            wrt.append(lineSeparator);
        }
        if (field.get_color()){
            wrt.append("tr");
        }else {
            wrt.append("fl");
        }
        wrt.close();
    }

    public static Field load_file() throws FileNotFoundException {
        File file = new File("savefile.txt");
        Scanner s = new Scanner(file);
        String string;
        Field field = new Field();
        for (int i = 0; i <= 7; ++i){
            for (int j = 0; j <= 7; ++j){
                string = s.next();
                if (string.equals("nl")){
                    field.set_null(i,j);
                }
                if (string.equals("tt")){
                    field.set_draught(new Draughts(i,j,true,true));
                }
                if (string.equals("tf")){
                    field.set_draught(new Draughts(i,j,true,false));
                }
                if (string.equals("ff")){
                    field.set_draught(new Draughts(i,j,false,false));
                }
                if (string.equals("ft")){
                    field.set_draught(new Draughts(i,j,false,true));
                }
            }
        }
        string = s.next();
        if (string.equals("tr")){
            field.set_color(true);
        }else {
            field.set_color(false);
        }
        return field;
    }

    public static int[] get_statistics(Field field){
        int[] statistics = {0,0,0,0,0,0};
        for (int i = 0; i <= 7; ++i){
            for (int j = 0; j <= 7; ++j){
                Draughts draught = field.get_draught(i,j);
                if (draught != null) {
                    if (draught.get_color()) {
                        ++statistics[0];
                        if (draught.get_type()) {
                            ++statistics[1];
                        }
                    }else {
                        ++statistics[3];
                        if (draught.get_type()) {
                            ++statistics[4];
                        }
                    }
                }
```

```
            }
        }
        statistics[2] = 12 - statistics[0];
        statistics[5] = 12 - statistics[3];
        return statistics;
    }

    private static boolean check_destruction_around(Field field){
        for (int i = 0; i <= 7; ++i) {
            for (int j = 0; j <= 7; ++j) {
                if ((field.get_draught(i,j) != null) &&
(field.get_draught(i,j).get_color() == field.get_color())) {
                    if ((i + 1 <= 7) && (i + 1 >= 0) && (j + 1 <= 7) && (j +
1 >= 0) &&
                            (field.check_destruction(i, j, i + 1, j + 1))) {
                        return true;
                    }
                    if ((i - 1 <= 7) && (i - 1 >= 0) && (j + 1 <= 7) && (j +
1 >= 0) &&
                            (field.check_destruction(i, j, i - 1, j + 1))) {
                        return true;
                    }
                    if ((i + 1 <= 7) && (i + 1 >= 0) && (j - 1 <= 7) && (j -
1 >= 0) &&
                            (field.check_destruction(i, j, i + 1, j - 1))) {
                        return true;
                    }
                    if ((i - 1 <= 7) && (i - 1 >= 0) && (j - 1 <= 7) && (j -
1 >= 0) &&
                            (field.check_destruction(i, j, i - 1, j - 1))) {
                        return true;
                    }
                }
            }
        }
        return false;
    }
}
```

```java
package com.example.rita.draughts;

import java.lang.Math;

public class Draughts {
    private int x;
    private int y;
    private boolean color;
    private boolean type;

    public Draughts(int x, int y, boolean color, boolean type){
        this.x = x;
        this.y = y;
        this.color = color;
        this.type = type;
    }

    public boolean check_of_move(int x, int y){
        if (type == false) {
            if (this.color == true) {
                if ((x - this.x == 1) && (Math.abs(y - this.y) == 1) && (x >=
0) && (x <= 7) && (y >= 0) && (y <= 7)) {
                    return true;
                } else {
                    return false;
                }
            } else {
                if ((this.x - x == 1) && (Math.abs(y - this.y) == 1) && (x >=
0) && (x <= 7) && (y >= 0) && (y <= 7)) {
                    return true;
                } else {
                    return false;
                }
            }
        }else{
            if ((Math.abs(this.x - x) == Math.abs(this.y - y)) && (x >= 0) &&
(x <= 7) && (y >= 0) && (y <= 7)) {
                return true;
            } else {
                return false;
            }
        }
    }

    public boolean check_of_type(){
        if (color == true){
            if (x == 7) {
                return true;
            }else{
                return false;
            }
        }else{
            if (x == 0) {
                return true;
            }else{
                return false;
            }
        }
    }

    public boolean get_type() {return type;}

    public int get_x() {return x;}
```

```
    public int get_y() {return y;}

    public void set_type(boolean type) {this.type = type;}

    public boolean get_color() { return color; }

    public void  set_x(int x) {this.x = x;}

    public void  set_y(int y) {this.y = y;}
}
```

```
package com.example.rita.draughts;


public class Field {
    private Draughts[][] draughts;
    private boolean color = true;

    public Field() {
        draughts = new Draughts[8][8];
        int i = 0;
        while (i <= 7){
            draughts[0][i] = new Draughts(0,i,true,false);
            i = i + 2;
        }
        i = 1;
        while (i <= 7){
            draughts[1][i] = new Draughts(1,i,true,false);
            i = i + 2;
        }
        i = 0;
        while (i <= 7){
            draughts[2][i] = new Draughts(2,i,true,false);
            i = i + 2;
        }
        i = 1;
        while (i <= 7){
            draughts[5][i] = new Draughts(5,i,false,false);
            i = i + 2;
        }
        i = 0;
        while (i <= 7){
            draughts[6][i] = new Draughts(6,i,false,false);
            i = i + 2;
        }
        i = 1;
        while (i <= 7) {
            draughts[7][i] = new Draughts(7,i,false,false);
            i = i + 2;
        }
        color = true;
    }

    public boolean check_free(int x, int y){
        if ((x <= 7) && (y <= 7) && (x >= 0) && (y >= 0)) {
            if (draughts[x][y] == null) {
                return true;
            } else {
                return false;
            }
        }else{
            return false;
        }
    }

    public boolean check_destruction(int x_selected, int y_selected, int
x_destroyed, int y_destroyed){
        if (!draughts[x_selected][y_selected].get_type()) {
            if (!draughts[x_selected][y_selected].get_color()) {
                if ((draughts[x_destroyed][y_destroyed] != null) &&
(Math.abs(draughts[x_selected][y_selected].get_x() -
draughts[x_destroyed][y_destroyed].get_x()) == 1) &&
                        (Math.abs(draughts[x_selected][y_selected].get_y() -
draughts[x_destroyed][y_destroyed].get_y()) == 1) &&
```

```
                        (draughts[x_destroyed][y_destroyed].get_color() !=
draughts[x_selected][y_selected].get_color()) &&
                        (check_free(x_selected + 2 * (x_destroyed -
x_selected), y_selected + 2 * (y_destroyed - y_selected))) &&
                        (x_selected + 2 * (x_destroyed - x_selected) >= 0) &&
                        (x_selected + 2 * (x_destroyed - x_selected) <= 7) &&
                        (y_selected + 2 * (y_destroyed - y_selected) >= 0) &&
                        (y_selected + 2 * (y_destroyed - y_selected) <= 7)) {
                    return true;
                } else {
                    return false;
                }
            } else {
                if ((draughts[x_destroyed][y_destroyed] != null) &&
(Math.abs(draughts[x_selected][y_selected].get_x() -
draughts[x_destroyed][y_destroyed].get_x()) == 1) &&
                        (Math.abs(draughts[x_selected][y_selected].get_y() -
draughts[x_destroyed][y_destroyed].get_y()) == 1) &&
                        (draughts[x_destroyed][y_destroyed].get_color() !=
draughts[x_selected][y_selected].get_color()) &&
                        (check_free(x_selected + 2 * (x_destroyed -
x_selected), y_selected + 2 * (y_destroyed - y_selected))) &&
                        (x_selected + 2 * (x_destroyed - x_selected) >= 0) &&
                        (x_selected + 2 * (x_destroyed - x_selected) <= 7) &&
                        (y_selected + 2 * (y_destroyed - y_selected) >= 0) &&
                        (y_selected + 2 * (y_destroyed - y_selected) <= 7)) {
                    return true;
                } else {
                    return false;
                }
            }
        }else {
            if ((x_destroyed - x_selected > 0) && (y_destroyed - y_selected >
0)) {
                if
((draughts[x_selected][y_selected].check_of_move(x_destroyed, y_destroyed))
                        && (draughts[x_destroyed][y_destroyed] != null)
                        && (draughts[x_destroyed][y_destroyed].get_color() !=
draughts[x_selected][y_selected].get_color())
                        && check_free(x_destroyed + 1, y_destroyed + 1) &&
                        (x_destroyed + 1 >= 0) &&
                        (x_destroyed + 1 <= 7) &&
                        (y_destroyed + 1 >= 0) &&
                        (y_destroyed + 1 <= 7)) {
                    return true;
                } else {
                    return false;
                }
            }
            if ((x_destroyed - x_selected > 0) && (y_destroyed - y_selected <
0)) {
                if
((draughts[x_selected][y_selected].check_of_move(x_destroyed, y_destroyed))
                        && (draughts[x_destroyed][y_destroyed] != null)
                        && (draughts[x_destroyed][y_destroyed].get_color() !=
draughts[x_selected][y_selected].get_color())
                        && check_free(x_destroyed + 1, y_destroyed - 1) &&
(x_destroyed + 1 >= 0)
                        && (x_destroyed + 1 <= 7)
                        && (y_destroyed - 1 >= 0)
                        && (y_destroyed - 1 <= 7)) {
                    return true;
                } else {
```

```
                    return false;
                }
            }
            if ((x_destroyed - x_selected < 0) && (y_destroyed - y_selected >
0)) {
                if
((draughts[x_selected][y_selected].check_of_move(x_destroyed, y_destroyed))
                        && (draughts[x_destroyed][y_destroyed] != null)
                        && (draughts[x_destroyed][y_destroyed].get_color() !=
draughts[x_selected][y_selected].get_color())
                        && check_free(x_destroyed - 1, y_destroyed + 1)
                        && (x_destroyed - 1 >= 0)
                        && (x_destroyed - 1 <= 7)
                        && (y_destroyed + 1 >= 0)
                        && (y_destroyed + 1 <= 7)) {
                    return true;
                } else {
                    return false;
                }
            }
            if ((x_destroyed - x_selected < 0) && (y_destroyed - y_selected <
0)) {
                if
((draughts[x_selected][y_selected].check_of_move(x_destroyed, y_destroyed))
                        && (draughts[x_destroyed][y_destroyed] != null)
                        && (draughts[x_destroyed][y_destroyed].get_color() !=
draughts[x_selected][y_selected].get_color())
                        && check_free(x_destroyed - 1, y_destroyed - 1)
                        && (x_destroyed - 1 >= 0)
                        && (x_destroyed - 1 <= 7)
                        && (y_destroyed - 1 >= 0)
                        && (y_destroyed - 1 <= 7)) {
                    return true;
                } else {
                    return false;
                }
            }
            return false;
        }
    }

    public  Draughts get_draught(int x, int y){return draughts[x][y];}

    public void set_draught(Draughts draught){
        draughts[draught.get_x()][draught.get_y()] = draught;
    }

    public void  set_null(int x, int y){
        draughts[x][y] = null;
    }

    public void set_color(boolean color){this.color = color;}

    public boolean get_color(){return color;}
}
```

```
package com.example.rita.draughts;

import android.app.Activity;
import android.os.Bundle;
import android.widget.Toast;

public class Game extends Activity {
    Field field;
    DraughtView draughtView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        field = new Field();
        draughtView = new DraughtView(this, field, this);
        setContentView(draughtView);
        draughtView.requestFocus();
    }

    @Override
    protected void onResume() {
        super.onResume();
        Music.play(this, R.raw.game);
    }

    @Override
    protected void onPause() {
        super.onPause();
        Music.stop(this);
    }

    public void wrong_move(){
        Toast error = Toast.makeText(getApplicationContext(),
R.string.wrong_move_text, Toast.LENGTH_LONG);
        error.show();
    }

    public void empty_cell (){
        Toast error = Toast.makeText(getApplicationContext(),
R.string.empty_cell_text, Toast.LENGTH_LONG);
        error.show();
    }

    public void wrong_color(){
        Toast error = Toast.makeText(getApplicationContext(),
R.string.wrong_color_text, Toast.LENGTH_LONG);
        error.show();
    }
}
```

Графическое приложение

```
package com.example.rita.draughts;

import android.app.Activity;
import android.os.Bundle;

public class About extends Activity{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.about);
    }
}
```

```java
package com.example.rita.draughts;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class Draught extends Activity implements View.OnClickListener {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        View newButton = findViewById(R.id.new_button);
        newButton.setOnClickListener(this);
        View aboutButton = findViewById(R.id.about_button);
        aboutButton.setOnClickListener(this);
        View exitButton = findViewById(R.id.exit_button);
        exitButton.setOnClickListener(this);
    }

    @Override
    protected void onResume() {
        super.onResume();
        Music.play(this, R.raw.main);
    }

    @Override
    protected void onPause() {
        super.onPause();
        Music.stop(this);
    }

    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.new_button:
                startGame();
                break;
            case R.id.about_button:
                Intent i = new Intent(this, About.class);
                startActivity(i);
                break;
            case R.id.exit_button:
                finish();
                break;

        }
    }

    private void startGame(){
        Intent intent = new Intent(Draught.this, Game.class);
        startActivity(intent);
    }
}
```

```
package com.example.rita.draughts;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.view.MotionEvent;
import android.view.View;

public class DraughtView extends View {
    Game game;
    private static final int cell_size = 130;
    Field field;
    private boolean click = false;
    private int x_first_click = -10;
    private int y_first_click = -10;
    private int x_second_click;
    private int y_second_click;
    private int fontSize = 50;

    public DraughtView(Context context, Field field, Game game) {
        super(context);
        this.field = field;
        this.game = game;
        setFocusable(true);
        setFocusableInTouchMode(true);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        Paint paint_backgroung = new Paint();

paint_backgroung.setColor(getResources().getColor(R.color.background_field));
        canvas.drawRect(0, 0, getWidth(), getHeight(), paint_backgroung);
        Paint paint_white_cell = new Paint();
        Paint paint_black_cell = new Paint();

paint_white_cell.setColor(getResources().getColor(R.color.white_cell));

paint_black_cell.setColor(getResources().getColor(R.color.black_cell));
        for (int i = 0; i <= 7; i = i + 2){
            canvas.drawRect(i * cell_size, 0, i * cell_size + cell_size,
cell_size, paint_white_cell);
            canvas.drawRect((i + 1) * cell_size, 0, (i + 1) * cell_size +
cell_size, cell_size, paint_black_cell);
            canvas.drawRect((i + 1) * cell_size, cell_size, (i + 1) *
cell_size + cell_size, 2 * cell_size, paint_white_cell);
            canvas.drawRect(i * cell_size, cell_size, i * cell_size +
cell_size, 2 * cell_size, paint_black_cell);
            canvas.drawRect(i * cell_size, 2 * cell_size, i * cell_size +
cell_size, 3 * cell_size, paint_white_cell);
            canvas.drawRect((i + 1) * cell_size, 2 * cell_size, (i + 1) *
cell_size + cell_size, 3 * cell_size, paint_black_cell);
            canvas.drawRect((i + 1) * cell_size, 3 * cell_size, (i + 1) *
cell_size + cell_size, 4 * cell_size, paint_white_cell);
            canvas.drawRect(i * cell_size, 3 * cell_size, i * cell_size +
cell_size, 4 * cell_size, paint_black_cell);
            canvas.drawRect(i * cell_size, 4 * cell_size, i * cell_size +
cell_size, 5 * cell_size, paint_white_cell);
            canvas.drawRect((i + 1) * cell_size, 4 * cell_size, (i + 1) *
cell_size + cell_size, 5 * cell_size, paint_black_cell);
            canvas.drawRect((i + 1) * cell_size, 5 * cell_size, (i + 1) *
```

```
cell_size + cell_size, 6 * cell_size, paint_white_cell);
            canvas.drawRect(i * cell_size, 5 * cell_size, i * cell_size +
cell_size, 6 * cell_size, paint_black_cell);
            canvas.drawRect(i * cell_size, 6 * cell_size, i * cell_size +
cell_size, 7 * cell_size, paint_white_cell);
            canvas.drawRect((i + 1) * cell_size, 6 * cell_size, (i + 1) *
cell_size + cell_size, 7 * cell_size, paint_black_cell);
            canvas.drawRect((i + 1) * cell_size, 7 * cell_size, (i + 1) *
cell_size + cell_size, 8 * cell_size, paint_white_cell);
            canvas.drawRect(i * cell_size, 7 * cell_size, i * cell_size +
cell_size, 8 * cell_size, paint_black_cell);
        }
        Paint paint_white_draught = new Paint();
        Paint paint_black_draught = new Paint();
        Paint paint_super_draught = new Paint();

paint_white_draught.setColor(getResources().getColor(R.color.white_draught));

paint_black_draught.setColor(getResources().getColor(R.color.black_draught));

paint_super_draught.setColor(getResources().getColor(R.color.super_draught));
        Path polygon = new Path();
        for (int i = 0; i <= 7; ++i){
            for (int j = 0; j <= 7; ++j){
                Draughts draught = field.get_draught(i,j);
                if ((!field.check_free(i,j)) && (draught.get_color())) {
                    canvas.drawCircle(j * cell_size + cell_size / 2, i *
cell_size + cell_size / 2, cell_size / 2, paint_white_draught);
                }
                if ((!field.check_free(i,j)) && (!draught.get_color())) {
                    canvas.drawCircle(j * cell_size + cell_size / 2, i *
cell_size + cell_size / 2, cell_size / 2, paint_black_draught);
                }
                if ((!field.check_free(i,j)) && (draught.get_type())) {
                    polygon.moveTo(j * cell_size + cell_size / 2 - 45, i *
cell_size + cell_size / 2 + 40);
                    polygon.lineTo(j * cell_size + cell_size / 2 - 45, i *
cell_size + cell_size / 2 - 40);
                    polygon.lineTo(j * cell_size + cell_size / 2 - 23, i *
cell_size + cell_size / 2);
                    polygon.lineTo(j * cell_size + cell_size / 2, i *
cell_size + cell_size / 2 - 40);
                    polygon.lineTo(j * cell_size + cell_size / 2 + 23, i *
cell_size + cell_size / 2);
                    polygon.lineTo(j * cell_size + cell_size / 2 + 45, i *
cell_size + cell_size / 2 - 40);
                    polygon.lineTo(j * cell_size + cell_size / 2 + 45, i *
cell_size + cell_size / 2 + 40);
                    canvas.drawPath(polygon, paint_super_draught);
                }
            }
        }
        Paint paint_select_draught = new Paint();

paint_select_draught.setColor(getResources().getColor(R.color.select_draught)
);
        canvas.drawCircle(x_first_click * cell_size + cell_size / 2,
y_first_click * cell_size + cell_size / 2, cell_size / 2,
paint_select_draught);
        Paint font = new Paint();
        font.setColor(Color.BLACK);
        font.setTextSize(fontSize);
        font.setStyle(Paint.Style.FILL);
```

```
        if (field.get_color()){
            canvas.drawText("Ход белого игрока", 0, 1090, font);
        }else {
            canvas.drawText("Ход чёрного игрока", 0, 1090, font);
        }
        int[] statistics = Api.get_statistics(field);
        canvas.drawText("Ост. белых шашек: " + statistics[0], 0, 1150, font);
        canvas.drawText("Белых дамок: " + statistics[1], 0, 1210, font);
        canvas.drawText("Белых шашек уничт.: " + statistics[2], 0, 1270,
font);
        canvas.drawText("Ост. чёрных шашек: " + statistics[3], 0, 1330,
font);
        canvas.drawText("Чёрных дамок: " + statistics[4], 0, 1390, font);
        canvas.drawText("Чёрных шашек уничт.: " + statistics[5], 0, 1450,
font);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if ((event.getAction() != MotionEvent.ACTION_DOWN) || ((event.getY()
> 8 * cell_size) || (event.getX() > 8 * cell_size)))
            return super.onTouchEvent(event);
        click = !click;
        if (click) {
            x_first_click = (int) event.getX() / 130;
            y_first_click = (int) event.getY() / 130;
            if (!field.check_free(y_first_click,x_first_click)) {
                if (field.get_color() ==
field.get_draught(y_first_click,x_first_click).get_color()) {
                    postInvalidate();
                }else {
                    x_first_click = -10;
                    y_first_click = -10;
                    click = !click;
                    game.wrong_color();
                }
            }else{
                x_first_click = -10;
                y_first_click = -10;
                click = !click;
                game.empty_cell();
            }
        }else {
            x_second_click = (int) event.getX() / 130;
            y_second_click = (int) event.getY() / 130;
            try {
                Api.move_draught(field,
y_first_click,x_first_click,y_second_click,x_second_click);
                postInvalidate();
            }catch (IllegalArgumentException ex){
                try {

Api.destroy_draught(field,y_first_click,x_first_click,y_second_click,x_second
_click);
                    postInvalidate();
                }catch (IllegalArgumentException e){
                    x_first_click = -10;
                    y_first_click = -10;
                    postInvalidate();
                    game.wrong_move();
                }
            }
            x_first_click = -10;
```

```
                y_first_click = -10;
            }
            return true;
        }
    }
```