Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

# Программирование

Отчет по курсовой работе
"Шашки"

**Работу выполнил:**
Корсков А.В.
Группа:
23501/4
**Преподаватель:**
Вылегжанина К.Д.

Санкт-Петербург
2017

# Содержание

# 1 Игра Шашки

## 1.1 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.

## 1.2 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.

## 1.3 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.

## 1.4 Минимально работоспособный продукт

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

## 1.5 Диаграмма прецедентов использования

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

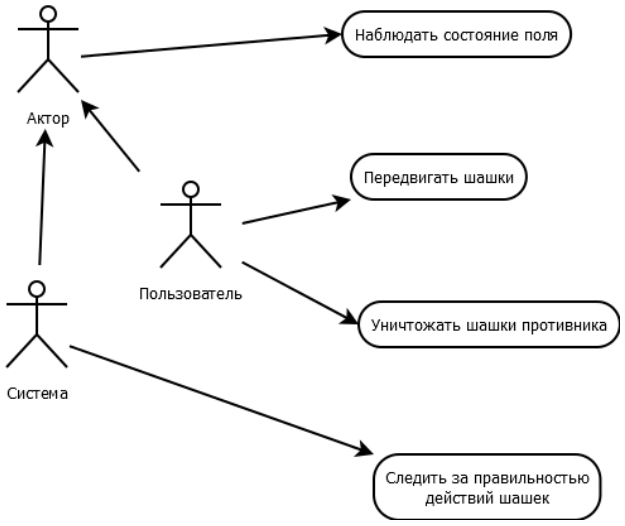


Рис. 1: Диаграмма прецедентов использования

# 2 Проектрование приложения, реализующего игру Шашки

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.

## 2.1 Выводы

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

# 3 Реализация игры Шашки

## 3.1 Версии программ

Операционная система: Windows 8, среда разработки: Android Studio 2.3, компилятор: Gradle 3.0, Java 1.8.0.120.

## 3.2 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 концаа 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

## 3.3 Андроид приложение

Андроид приложение позволяет играть на устройствах под управлением ОС Андроид.
Основные классы, выделенные в Андроид приложении.

- 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.
- Класс Music. Класс отвечающий за воспроизводство музыки. В нём можно запускать и останавливать проигрывание аудио файла.

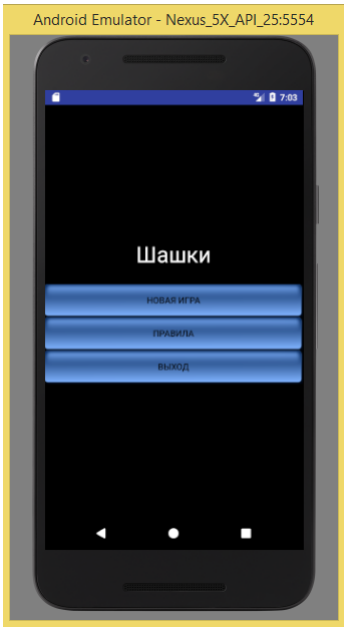


Рис. 2: Главное меню графического приложения

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

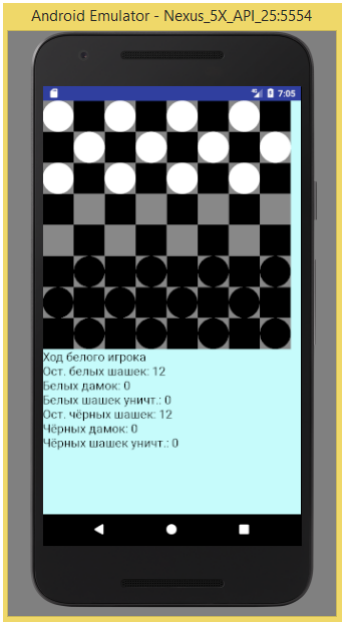


Рис. 3: Представление поля в графическом приложении

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. 4: Окно с правилами игры Шашки

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

# 4 Процесс обеспечения качества и тестирование

## 4.1 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.

## 4.2 Демонстрации

Во время создания приложения была проведена 1 демонстрация. По результату которой не было найдено серьёзных замечаний и недоработок к проекту.

# 5 Вывод

По окончании семестра автор проекта научился писать программы на языке Java для операционной системы Android, делать графический интерфейс с помощью Swing, а также получил опыт работы с большими проектами на Java, содержащими много классов и имеющих как консольное приложение, так и графическое.

# 6 Приложение 1. Листинги кода

## 6.1 Библиотека

```
1  package com.example.aleksey.draughts;
2
3  import java.lang.Math;
4
5  public class Draughts {
6      private int x;
7      private int y;
8      private boolean color;
9      private boolean type;
10
11     public Draughts(int x, int y, boolean color, boolean type){
12         this.x = x;
13         this.y = y;
14         this.color = color;
15         this.type = type;
16     }
17
18     public boolean check_of_move(int x, int y){
19         if (type == false) {
20             if (this.color == true) {
21                 if ((x - this.x == 1) && (Math.abs(y - this.y) ==
   ↪ 1) && (x >= 0) && (x <= 7) && (y >= 0) && (y <= 7)) {
22                     return true;
23                 } else {
24                     return false;
25                 }
26             } else {
27                 if ((this.x - x == 1) && (Math.abs(y - this.y) ==
   ↪ 1) && (x >= 0) && (x <= 7) && (y >= 0) && (y <= 7)) {
28                     return true;
29                 } else {
30                     return false;
31                 }
32             }
33         }else{
34             if ((Math.abs(this.x - x) == Math.abs(this.y - y)) && (
   ↪ x >= 0) && (x <= 7) && (y >= 0) && (y <= 7)) {
35                 return true;
36             } else {
37                 return false;
38             }
39         }
40     }
41
42     public boolean check_of_type(){
43         if (color == true){
44             if (x == 7) {
45                 return true;
46             }else{
47                 return false;
48             }
49         }else{
50             if (x == 0) {
51                 return true;
52             }else{
53                 return false;
54             }
55         }
```

```
56      }
57
58      public boolean get_type() {return type;}
59
60      public int get_x() {return x;}
61
62      public int get_y() {return y;}
63
64      public void set_type(boolean type) {this.type = type;}
65
66      public boolean get_color() { return color; }
67
68      public void  set_x(int x) {this.x = x;}
69
70      public void  set_y(int y) {this.y = y;}
71  }
```

```
1  package com.example.aleksey.draughts;
2
3
4  public class Field {
5      private Draughts[][] draughts;
6      private boolean color = true;
7
8      public Field() {
9          draughts = new Draughts[8][8];
10         int i = 0;
11         while (i <= 7){
12             draughts[0][i] = new Draughts(0,i,true,false);
13             i = i + 2;
14         }
15         i = 1;
16         while (i <= 7){
17             draughts[1][i] = new Draughts(1,i,true,false);
18             i = i + 2;
19         }
20         i = 0;
21         while (i <= 7){
22             draughts[2][i] = new Draughts(2,i,true,false);
23             i = i + 2;
24         }
25         i = 1;
26         while (i <= 7){
27             draughts[5][i] = new Draughts(5,i,false,false);
28             i = i + 2;
29         }
30         i = 0;
31         while (i <= 7){
32             draughts[6][i] = new Draughts(6,i,false,false);
33             i = i + 2;
34         }
35         i = 1;
36         while (i <= 7) {
37             draughts[7][i] = new Draughts(7,i,false,false);
38             i = i + 2;
39         }
40         color = true;
41     }
42
43     public boolean check_free(int x, int y){
44         if ((x <= 7) && (y <= 7) && (x >= 0) && (y >= 0)) {
45             if (draughts[x][y] == null) {
46                 return true;
47             } else {
48                 return false;
49             }
50         }else{
51             return false;
52         }
53     }
54
55     public boolean check_destruction(int x_selected, int y_selected
   ↪ , int x_destroyed, int y_destroyed){
56         if (!draughts[x_selected][y_selected].get_type()) {
57             if (!draughts[x_selected][y_selected].get_color()) {
58                 if ((draughts[x_destroyed][y_destroyed] != null) &&
   ↪  (Math.abs(draughts[x_selected][y_selected].get_x() -
   ↪ draughts[x_destroyed][y_destroyed].get_x()) == 1) &&
59                         (Math.abs(draughts[x_selected][y_selected].
```

```
    ↪ get_y() - draughts[x_destroyed][y_destroyed].get_y()) == 1)
    ↪ &&
60                      (draughts[x_destroyed][y_destroyed].
    ↪ get_color() != draughts[x_selected][y_selected].get_color())
    ↪  &&
61                      (check_free(x_selected + 2 * (x_destroyed -
    ↪  x_selected), y_selected + 2 * (y_destroyed - y_selected)))
    ↪ &&
62                      (x_selected + 2 * (x_destroyed - x_selected
    ↪ ) >= 0) &&
63                      (x_selected + 2 * (x_destroyed - x_selected
    ↪ ) <= 7) &&
64                      (y_selected + 2 * (y_destroyed - y_selected
    ↪ ) >= 0) &&
65                      (y_selected + 2 * (y_destroyed - y_selected
    ↪ ) <= 7)) {
66                  return true;
67              } else {
68                  return false;
69              }
70          } else {
71              if ((draughts[x_destroyed][y_destroyed] != null) &&
    ↪  (Math.abs(draughts[x_selected][y_selected].get_x() -
    ↪ draughts[x_destroyed][y_destroyed].get_x()) == 1) &&
72                      (Math.abs(draughts[x_selected][y_selected].
    ↪ get_y() - draughts[x_destroyed][y_destroyed].get_y()) == 1)
    ↪ &&
73                      (draughts[x_destroyed][y_destroyed].
    ↪ get_color() != draughts[x_selected][y_selected].get_color())
    ↪  &&
74                      (check_free(x_selected + 2 * (x_destroyed -
    ↪  x_selected), y_selected + 2 * (y_destroyed - y_selected)))
    ↪ &&
75                      (x_selected + 2 * (x_destroyed - x_selected
    ↪ ) >= 0) &&
76                      (x_selected + 2 * (x_destroyed - x_selected
    ↪ ) <= 7) &&
77                      (y_selected + 2 * (y_destroyed - y_selected
    ↪ ) >= 0) &&
78                      (y_selected + 2 * (y_destroyed - y_selected
    ↪ ) <= 7)) {
79                  return true;
80              } else {
81                  return false;
82              }
83          }
84       }else {
85          if ((x_destroyed - x_selected > 0) && (y_destroyed -
    ↪ y_selected > 0)) {
86              if ((draughts[x_selected][y_selected].check_of_move
    ↪ (x_destroyed, y_destroyed))
87                      && (draughts[x_destroyed][y_destroyed] !=
    ↪ null)
88                      && (draughts[x_destroyed][y_destroyed].
    ↪ get_color() != draughts[x_selected][y_selected].get_color())
89                      && check_free(x_destroyed + 1, y_destroyed
    ↪ + 1) &&
90                      (x_destroyed + 1 >= 0) &&
91                      (x_destroyed + 1 <= 7) &&
92                      (y_destroyed + 1 >= 0) &&
93                      (y_destroyed + 1 <= 7)) {
94                  return true;
```

```
95                  } else {
96                      return false;
97                  }
98              }
99              if ((x_destroyed - x_selected > 0) && (y_destroyed -
    ↪ y_selected < 0)) {
100                 if ((draughts[x_selected][y_selected].check_of_move
    ↪ (x_destroyed, y_destroyed))
101                         && (draughts[x_destroyed][y_destroyed] !=
    ↪ null)
102                         && (draughts[x_destroyed][y_destroyed].
    ↪ get_color() != draughts[x_selected][y_selected].get_color())
103                         && check_free(x_destroyed + 1, y_destroyed
    ↪ - 1) && (x_destroyed + 1 >= 0)
104                         && (x_destroyed + 1 <= 7)
105                         && (y_destroyed - 1 >= 0)
106                         && (y_destroyed - 1 <= 7)) {
107                     return true;
108                 } else {
109                     return false;
110                 }
111             }
112             if ((x_destroyed - x_selected < 0) && (y_destroyed -
    ↪ y_selected > 0)) {
113                 if ((draughts[x_selected][y_selected].check_of_move
    ↪ (x_destroyed, y_destroyed))
114                         && (draughts[x_destroyed][y_destroyed] !=
    ↪ null)
115                         && (draughts[x_destroyed][y_destroyed].
    ↪ get_color() != draughts[x_selected][y_selected].get_color())
116                         && check_free(x_destroyed - 1, y_destroyed
    ↪ + 1)
117                         && (x_destroyed - 1 >= 0)
118                         && (x_destroyed - 1 <= 7)
119                         && (y_destroyed + 1 >= 0)
120                         && (y_destroyed + 1 <= 7)) {
121                     return true;
122                 } else {
123                     return false;
124                 }
125             }
126             if ((x_destroyed - x_selected < 0) && (y_destroyed -
    ↪ y_selected < 0)) {
127                 if ((draughts[x_selected][y_selected].check_of_move
    ↪ (x_destroyed, y_destroyed))
128                         && (draughts[x_destroyed][y_destroyed] !=
    ↪ null)
129                         && (draughts[x_destroyed][y_destroyed].
    ↪ get_color() != draughts[x_selected][y_selected].get_color())
130                         && check_free(x_destroyed - 1, y_destroyed
    ↪ - 1)
131                         && (x_destroyed - 1 >= 0)
132                         && (x_destroyed - 1 <= 7)
133                         && (y_destroyed - 1 >= 0)
134                         && (y_destroyed - 1 <= 7)) {
135                     return true;
136                 } else {
137                     return false;
138                 }
139             }
140             return false;
141         }
```

```
142     }
143
144     public  Draughts get_draught(int x, int y){return draughts[x][y
        ↪ ];}
145
146     public void set_draught(Draughts draught){
147         draughts[draught.get_x()][draught.get_y()] = draught;
148     }
149
150     public void  set_null(int x, int y){
151         draughts[x][y] = null;
152     }
153
154     public void set_color(boolean color){this.color = color;}
155
156     public boolean get_color(){return color;}
157 }
```

```java
package com.example.aleksey.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);
```

```
57                      field.set_null(x_destroyed, y_destroyed);
58                      if ((x_destroyed - x_selected < 0) && (
    ↪ y_destroyed - y_selected < 0)){
59                          draught.set_x(x_destroyed - 1);
60                          draught.set_y(y_destroyed - 1);
61                          field.set_draught(draught);
62                      }
63                      if ((x_destroyed - x_selected < 0) && (
    ↪ y_destroyed - y_selected > 0)){
64                          draught.set_x(x_destroyed - 1);
65                          draught.set_y(y_destroyed + 1);
66                          field.set_draught(draught);
67                      }
68                      if ((x_destroyed - x_selected > 0) && (
    ↪ y_destroyed - y_selected < 0)){
69                          draught.set_x(x_destroyed + 1);
70                          draught.set_y(y_destroyed - 1);
71                          field.set_draught(draught);
72                      }
73                      if ((x_destroyed - x_selected > 0) && (
    ↪ y_destroyed - y_selected > 0)){
74                          draught.set_x(x_destroyed + 1);
75                          draught.set_y(y_destroyed + 1);
76                          field.set_draught(draught);
77                      }
78                  }
79              }
80          }
81          if (!check_destruction_around(field)) {
82              field.set_color(!field.get_color());
83          }
84      }
85
86      public static void save_file(Field field) throws IOException {
87          File file = new File("savefile.txt");
88          if(!file.exists()){
89              file.createNewFile();
90          }
91          FileWriter wrt = new FileWriter(file);
92          String lineSeparator = System.getProperty("line.separator")
    ↪ ;
93          for (int i = 0; i <= 7; ++i){
94              for (int j = 0; j <= 7; ++j){
95                  Draughts draught = field.get_draught(i,j);
96                  if (draught == null){
97                      wrt.append("nl␣");
98                  }else {
99                      if ((draught.get_color()) && (!draught.get_type
    ↪ ())) {
100                         wrt.append("tf␣");
101                     }
102                     if ((draught.get_color()) && (draught.get_type
    ↪ ())) {
103                         wrt.append("tt␣");
104                     }
105                     if ((!draught.get_color()) && (!draught.
    ↪ get_type())) {
106                         wrt.append("ff␣");
107                     }
108                     if ((!draught.get_color()) && (draught.get_type
    ↪ ())) {
109                         wrt.write("ft␣");
```

```
110                     }
111                 }
112             }
113             wrt.append(lineSeparator);
114         }
115         if (field.get_color()){
116             wrt.append("tr");
117         }else {
118             wrt.append("fl");
119         }
120         wrt.close();
121     }
122 
123     public static Field load_file() throws FileNotFoundException {
124         File file = new File("savefile.txt");
125         Scanner s = new Scanner(file);
126         String string;
127         Field field = new Field();
128         for (int i = 0; i <= 7; ++i){
129             for (int j = 0; j <= 7; ++j){
130                 string = s.next();
131                 if (string.equals("nl")){
132                     field.set_null(i,j);
133                 }
134                 if (string.equals("tt")){
135                     field.set_draught(new Draughts(i,j,true,true));
136                 }
137                 if (string.equals("tf")){
138                     field.set_draught(new Draughts(i,j,true,false))
    ↪ ;
139                 }
140                 if (string.equals("ff")){
141                     field.set_draught(new Draughts(i,j,false,false)
    ↪ );
142                 }
143                 if (string.equals("ft")){
144                     field.set_draught(new Draughts(i,j,false,true))
    ↪ ;
145                 }
146             }
147         }
148         string = s.next();
149         if (string.equals("tr")){
150             field.set_color(true);
151         }else {
152             field.set_color(false);
153         }
154         return field;
155     }
156 
157     public static int[] get_statistics(Field field){
158         int[] statistics = {0,0,0,0,0,0};
159         for (int i = 0; i <= 7; ++i){
160             for (int j = 0; j <= 7; ++j){
161                 Draughts draught = field.get_draught(i,j);
162                 if (draught != null) {
163                     if (draught.get_color()) {
164                         ++statistics[0];
165                         if (draught.get_type()) {
166                             ++statistics[1];
167                         }
168                     }else {
```

```
169                         ++statistics[3];
170                         if (draught.get_type()) {
171                             ++statistics[4];
172                         }
173                     }
174                 }
175             }
176         }
177         statistics[2] = 12 - statistics[0];
178         statistics[5] = 12 - statistics[3];
179         return statistics;
180     }
181 
182     private static boolean check_destruction_around(Field field){
183         for (int i = 0; i <= 7; ++i) {
184             for (int j = 0; j <= 7; ++j) {
185                 if ((field.get_draught(i,j) != null) && (field.
    ↪ get_draught(i,j).get_color() == field.get_color())) {
186                     if ((i + 1 <= 7) && (i + 1 >= 0) && (j + 1 <=
    ↪ 7) && (j + 1 >= 0) &&
187                             (field.check_destruction(i, j, i + 1, j
    ↪  + 1))) {
188                         return true;
189                     }
190                     if ((i - 1 <= 7) && (i - 1 >= 0) && (j + 1 <=
    ↪ 7) && (j + 1 >= 0) &&
191                             (field.check_destruction(i, j, i - 1, j
    ↪  + 1))) {
192                         return true;
193                     }
194                     if ((i + 1 <= 7) && (i + 1 >= 0) && (j - 1 <=
    ↪ 7) && (j - 1 >= 0) &&
195                             (field.check_destruction(i, j, i + 1, j
    ↪  - 1))) {
196                         return true;
197                     }
198                     if ((i - 1 <= 7) && (i - 1 >= 0) && (j - 1 <=
    ↪ 7) && (j - 1 >= 0) &&
199                             (field.check_destruction(i, j, i - 1, j
    ↪  - 1))) {
200                         return true;
201                     }
202                 }
203             }
204         }
205         return false;
206     }
207 }
```

## 6.2 Графическое приложение

```
1  package com.example.aleksey.draughts;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5
6  public class About extends Activity{
7      @Override
8      protected void onCreate(Bundle savedInstanceState) {
9          super.onCreate(savedInstanceState);
10         setContentView(R.layout.about);
11     }
12 }
```

```
1  package com.example.aleksey.draughts;
2
3  import android.app.Activity;
4  import android.content.Intent;
5  import android.os.Bundle;
6  import android.view.View;
7
8  public class Draught extends Activity implements View.
       ↪ OnClickListener {
9
10     @Override
11     public void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.main);
14         View newButton = findViewById(R.id.new_button);
15         newButton.setOnClickListener(this);
16         View aboutButton = findViewById(R.id.about_button);
17         aboutButton.setOnClickListener(this);
18         View exitButton = findViewById(R.id.exit_button);
19         exitButton.setOnClickListener(this);
20     }
21
22     @Override
23     protected void onResume() {
24         super.onResume();
25         Music.play(this, R.raw.main);
26     }
27
28     @Override
29     protected void onPause() {
30         super.onPause();
31         Music.stop(this);
32     }
33
34     public void onClick(View v) {
35         switch (v.getId()) {
36             case R.id.new_button:
37                 startGame();
38                 break;
39             case R.id.about_button:
40                 Intent i = new Intent(this, About.class);
41                 startActivity(i);
42                 break;
43             case R.id.exit_button:
44                 finish();
45                 break;
46
47         }
48     }
49
50     private void startGame(){
51         Intent intent = new Intent(Draught.this, Game.class);
52         startActivity(intent);
53     }
54 }
```

```
1  package com.example.aleksey.draughts;
2
3  import android.content.Context;
4  import android.graphics.Canvas;
5  import android.graphics.Color;
6  import android.graphics.Paint;
7  import android.graphics.Path;
8  import android.view.MotionEvent;
9  import android.view.View;
10
11 public class DraughtView extends View {
12     Game game;
13     private static final int cell_size = 130;
14     Field field;
15     private boolean click = false;
16     private int x_first_click = -10;
17     private int y_first_click = -10;
18     private int x_second_click;
19     private int y_second_click;
20     private int fontSize = 50;
21
22     public DraughtView(Context context, Field field, Game game) {
23         super(context);
24         this.field = field;
25         this.game = game;
26         setFocusable(true);
27         setFocusableInTouchMode(true);
28     }
29
30     @Override
31     protected void onDraw(Canvas canvas) {
32         Paint paint_backgroung = new Paint();
33         paint_backgroung.setColor(getResources().getColor(R.color.
   ↪ background_field));
34         canvas.drawRect(0, 0, getWidth(), getHeight(),
   ↪ paint_backgroung);
35         Paint paint_white_cell = new Paint();
36         Paint paint_black_cell = new Paint();
37         paint_white_cell.setColor(getResources().getColor(R.color.
   ↪ white_cell));
38         paint_black_cell.setColor(getResources().getColor(R.color.
   ↪ black_cell));
39         for (int i = 0; i <= 7; i = i + 2){
40             canvas.drawRect(i * cell_size, 0, i * cell_size +
   ↪ cell_size, cell_size, paint_white_cell);
41             canvas.drawRect((i + 1) * cell_size, 0, (i + 1) *
   ↪ cell_size + cell_size, cell_size, paint_black_cell);
42             canvas.drawRect((i + 1) * cell_size, cell_size, (i + 1)
   ↪  * cell_size + cell_size, 2 * cell_size, paint_white_cell);
43             canvas.drawRect(i * cell_size, cell_size, i * cell_size
   ↪  + cell_size, 2 * cell_size, paint_black_cell);
44             canvas.drawRect(i * cell_size, 2 * cell_size, i *
   ↪ cell_size + cell_size, 3 * cell_size, paint_white_cell);
45             canvas.drawRect((i + 1) * cell_size, 2 * cell_size, (i
   ↪ + 1) * cell_size + cell_size, 3 * cell_size,
   ↪ paint_black_cell);
46             canvas.drawRect((i + 1) * cell_size, 3 * cell_size, (i
   ↪ + 1) * cell_size + cell_size, 4 * cell_size,
   ↪ paint_white_cell);
47             canvas.drawRect(i * cell_size, 3 * cell_size, i *
   ↪ cell_size + cell_size, 4 * cell_size, paint_black_cell);
48             canvas.drawRect(i * cell_size, 4 * cell_size, i *
```

```
    ↪ cell_size + cell_size, 5 * cell_size, paint_white_cell);
49              canvas.drawRect((i + 1) * cell_size, 4 * cell_size, (i
    ↪ + 1) * cell_size + cell_size, 5 * cell_size,
    ↪ paint_black_cell);
50              canvas.drawRect((i + 1) * cell_size, 5 * cell_size, (i
    ↪ + 1) * cell_size + cell_size, 6 * cell_size,
    ↪ paint_white_cell);
51              canvas.drawRect(i * cell_size, 5 * cell_size, i *
    ↪ cell_size + cell_size, 6 * cell_size, paint_black_cell);
52              canvas.drawRect(i * cell_size, 6 * cell_size, i *
    ↪ cell_size + cell_size, 7 * cell_size, paint_white_cell);
53              canvas.drawRect((i + 1) * cell_size, 6 * cell_size, (i
    ↪ + 1) * cell_size + cell_size, 7 * cell_size,
    ↪ paint_black_cell);
54              canvas.drawRect((i + 1) * cell_size, 7 * cell_size, (i
    ↪ + 1) * cell_size + cell_size, 8 * cell_size,
    ↪ paint_white_cell);
55              canvas.drawRect(i * cell_size, 7 * cell_size, i *
    ↪ cell_size + cell_size, 8 * cell_size, paint_black_cell);
56          }
57          Paint paint_white_draught = new Paint();
58          Paint paint_black_draught = new Paint();
59          Paint paint_super_draught = new Paint();
60          paint_white_draught.setColor(getResources().getColor(R.
    ↪ color.white_draught));
61          paint_black_draught.setColor(getResources().getColor(R.
    ↪ color.black_draught));
62          paint_super_draught.setColor(getResources().getColor(R.
    ↪ color.super_draught));
63          Path polygon = new Path();
64          for (int i = 0; i <= 7; ++i){
65              for (int j = 0; j <= 7; ++j){
66                  Draughts draught = field.get_draught(i,j);
67                  if ((!field.check_free(i,j)) && (draught.get_color
    ↪ ())) {
68                      canvas.drawCircle(j * cell_size + cell_size /
    ↪ 2, i * cell_size + cell_size / 2, cell_size / 2,
    ↪ paint_white_draught);
69                  }
70                  if ((!field.check_free(i,j)) && (!draught.get_color
    ↪ ())) {
71                      canvas.drawCircle(j * cell_size + cell_size /
    ↪ 2, i * cell_size + cell_size / 2, cell_size / 2,
    ↪ paint_black_draught);
72                  }
73                  if ((!field.check_free(i,j)) && (draught.get_type()
    ↪ )) {
74                      polygon.moveTo(j * cell_size + cell_size / 2 -
    ↪ 45, i * cell_size + cell_size / 2 + 40);
75                      polygon.lineTo(j * cell_size + cell_size / 2 -
    ↪ 45, i * cell_size + cell_size / 2 - 40);
76                      polygon.lineTo(j * cell_size + cell_size / 2 -
    ↪ 23, i * cell_size + cell_size / 2);
77                      polygon.lineTo(j * cell_size + cell_size / 2, i
    ↪  * cell_size + cell_size / 2 - 40);
78                      polygon.lineTo(j * cell_size + cell_size / 2 +
    ↪ 23, i * cell_size + cell_size / 2);
79                      polygon.lineTo(j * cell_size + cell_size / 2 +
    ↪ 45, i * cell_size + cell_size / 2 - 40);
80                      polygon.lineTo(j * cell_size + cell_size / 2 +
    ↪ 45, i * cell_size + cell_size / 2 + 40);
81                      canvas.drawPath(polygon, paint_super_draught);
```

```
82                  }
83              }
84          }
85          Paint paint_select_draught = new Paint();
86          paint_select_draught.setColor(getResources().getColor(R.
    ↪ color.select_draught));
87          canvas.drawCircle(x_first_click * cell_size + cell_size /
    ↪ 2, y_first_click * cell_size + cell_size / 2, cell_size / 2,
    ↪  paint_select_draught);
88          Paint font = new Paint();
89          font.setColor(Color.BLACK);
90          font.setTextSize(fontSize);
91          font.setStyle(Paint.Style.FILL);
92          if (field.get_color()){
93              canvas.drawText("Ход белого игрока", 0, 1090, font);
94          }else {
95              canvas.drawText("Ход чёрного игрока", 0, 1090, font);
96          }
97          int[] statistics = Api.get_statistics(field);
98          canvas.drawText("Ост. белых шашек: " + statistics[0], 0,
    ↪ 1150, font);
99          canvas.drawText("Белых дамок: " + statistics[1], 0, 1210,
    ↪ font);
100         canvas.drawText("Белых шашек уничт.: " + statistics[2], 0,
    ↪ 1270, font);
101         canvas.drawText("Ост. чёрных шашек: " + statistics[3], 0,
    ↪ 1330, font);
102         canvas.drawText("Чёрных дамок: " + statistics[4], 0, 1390,
    ↪ font);
103         canvas.drawText("Чёрных шашек уничт.: " + statistics[5], 0,
    ↪ 1450, font);
104     }
105
106     @Override
107     public boolean onTouchEvent(MotionEvent event) {
108         if ((event.getAction() != MotionEvent.ACTION_DOWN) || ((
    ↪ event.getY() > 8 * cell_size) || (event.getX() > 8 *
    ↪ cell_size)))
109             return super.onTouchEvent(event);
110         click = !click;
111         if (click) {
112             x_first_click = (int) event.getX() / 130;
113             y_first_click = (int) event.getY() / 130;
114             if (!field.check_free(y_first_click,x_first_click)) {
115                 if (field.get_color() == field.get_draught(
    ↪ y_first_click,x_first_click).get_color()) {
116                     postInvalidate();
117                 }else {
118                     x_first_click = -10;
119                     y_first_click = -10;
120                     click = !click;
121                     game.wrong_color();
122                 }
123             }else{
124                 x_first_click = -10;
125                 y_first_click = -10;
126                 click = !click;
127                 game.empty_cell();
128             }
129         }else {
130             x_second_click = (int) event.getX() / 130;
131             y_second_click = (int) event.getY() / 130;
```

```
132             try {
133                 Api.move_draught(field, y_first_click,x_first_click
    ↪ ,y_second_click,x_second_click);
134                 postInvalidate();
135             }catch (IllegalArgumentException ex){
136                 try {
137                     Api.destroy_draught(field,y_first_click,
    ↪ x_first_click,y_second_click,x_second_click);
138                     postInvalidate();
139                 }catch (IllegalArgumentException e){
140                     x_first_click = -10;
141                     y_first_click = -10;
142                     postInvalidate();
143                     game.wrong_move();
144                 }
145             }
146             x_first_click = -10;
147             y_first_click = -10;
148         }
149         return true;
150     }
151 }
```

```
1  package com.example.aleksey.draughts;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.widget.LinearLayout;
6  import android.widget.Toast;
7
8  public class Game extends Activity {
9      Field field;
10     DraughtView draughtView;
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         field = new Field();
16         draughtView = new DraughtView(this, field, this);
17         setContentView(draughtView);
18         draughtView.requestFocus();
19     }
20
21     @Override
22     protected void onResume() {
23         super.onResume();
24         Music.play(this, R.raw.game);
25     }
26
27     @Override
28     protected void onPause() {
29         super.onPause();
30         Music.stop(this);
31     }
32
33     public void wrong_move(){
34         Toast error = Toast.makeText(getApplicationContext(), R.
       ↪ string.wrong_move_text, Toast.LENGTH_LONG);
35         error.show();
36     }
37
38     public void empty_cell (){
39         Toast error = Toast.makeText(getApplicationContext(), R.
       ↪ string.empty_cell_text, Toast.LENGTH_LONG);
40         error.show();
41     }
42
43     public void wrong_color(){
44         Toast error = Toast.makeText(getApplicationContext(), R.
       ↪ string.wrong_color_text, Toast.LENGTH_LONG);
45         error.show();
46     }
47 }
```

```
1  package com.example.aleksey.draughts;
2
3  import android.media.MediaPlayer;
4  import android.content.Context;
5
6  public class Music {
7      private static MediaPlayer mp = null;
8
9      public static void play(Context context, int resource) {
10         stop(context);
11         mp = MediaPlayer.create(context, resource);
12         mp.setLooping(true);
13         mp.start();
14     }
15
16     public static void stop(Context context) {
17         if (mp != null) {
18             mp.stop();
19             mp.release();
20             mp = null;
21         }
22     }
23 }
```