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» 3](#_Toc492711282)

[Задание 3](#_Toc492711283)

[Правила работы программы 3](#_Toc492711284)

[Концепция 3](#_Toc492711285)

[Минимально работоспособный продукт 3](#_Toc492711286)

[Диаграмма прецедентов использования 3](#_Toc492711287)

[Проектирование приложения, реализующего игру «Шашки» 5](#_Toc492711288)

[Компоненты 5](#_Toc492711289)

[Выводы 5](#_Toc492711290)

[Реализация игры «Шашки» 6](#_Toc492711291)

[Версия программ 6](#_Toc492711292)

[Библиотека 6](#_Toc492711293)

[Андроид приложение 6](#_Toc492711294)

[Тестирование 9](#_Toc492711295)

[Вывод 10](#_Toc492711296)

[Листинги 11](#_Toc492711297)

[Библиотека 11](#_Toc492711298)

[Графическое приложение 21](#_Toc492711299)

# Игра «Шашки»

## Задание

У нас имеется прямоугольное поле размером 8х8, 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 8х8. 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)

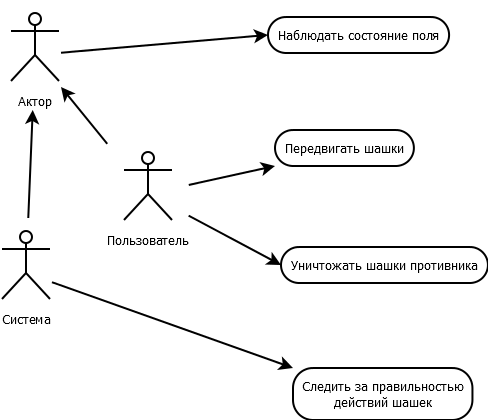


Рис.1 Диаграмма прецедентов использования

# 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.

1. **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.

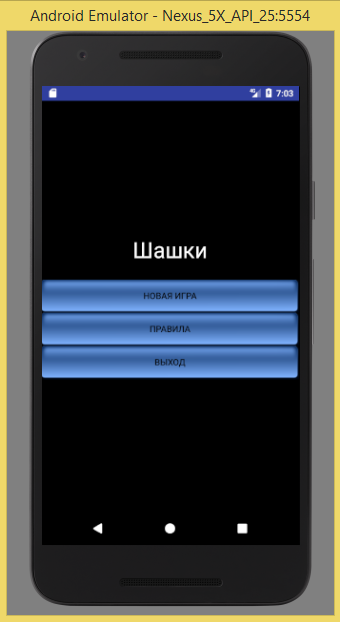


Рис. 2 Главное меню

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**;  } } |

|  |
| --- |
| **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***);  } } |

|  |
| --- |
| **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**;  } } |