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
2016

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. Также важной функцией программы является возможность сохранить текущее состояние игры в файл и загрузка игры из файла.

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 и сохранение и загрузка игры в файл.

1.5 Диаграмма прецедентов использования

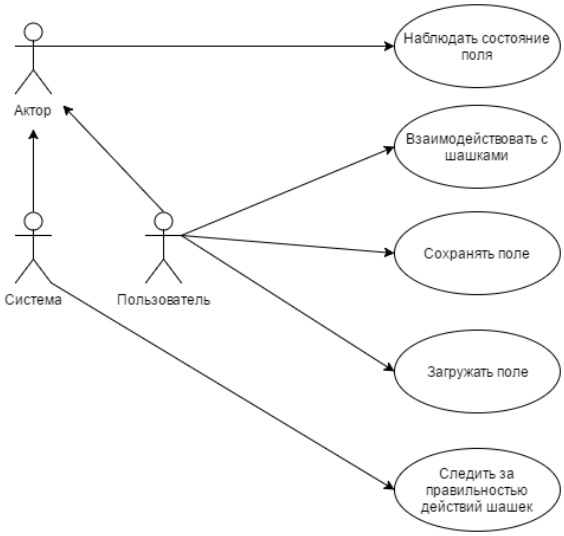

Рис. 1: Диаграмма прецедентов использования

2 Проектрование приложения, реализующего игру Шашки

Программа разделена на 4 подпроекта: app - консольное приложение, core - библиотека, реализующая игру Шашки, gui - графическое приложение, test - тесты для программы.

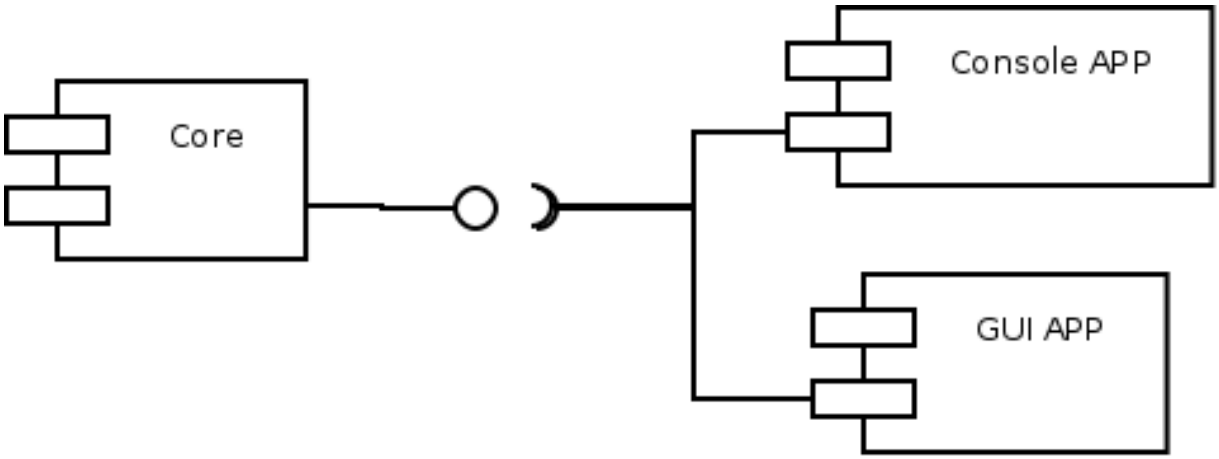

Рис. 2: Диаграмма прецедентов использования

2.1 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.
В API выделены следующие методы:

- move_draught - метод, передвигающий шашку.
- destroy_draught - метод, уничтожающий шашку противника.
- save_file - метод, сохраняющий поле в файл.
- load_file - метод, загружающий поле из файла.
- get_statistics - метод, анализирующий поле и выдающий статистику.
- check_destruction_around - метод, проверяющий может ли шашка кого-нибудь уничтожить вокруг себя.

3 Реализация игры Шашки

3.1 Версии программ

Операционная система: Windows 8, среда разработки: Intellij IDEA 2016.2.4, компилятор: Gradle 3.0, Java 1.8.0.120.

3.2 Консольное приложение

Консольное приложение позволяет работать с моделью через консоль. Основные классы, выделенные в консольном приложении:

- Класс Console_ui. Сначала выводит главное меню, где можно начать новую игру или загрузить. Также есть метод выводящий второе меню, где можно передвинуть шашку, уничтожить шашку противника, вывести текущее состояние поля в консоль и сохранить текущее состояние поля в консоль. Также в классе есть вспомогательный метод, задачей которого является вывод поля в консоль.

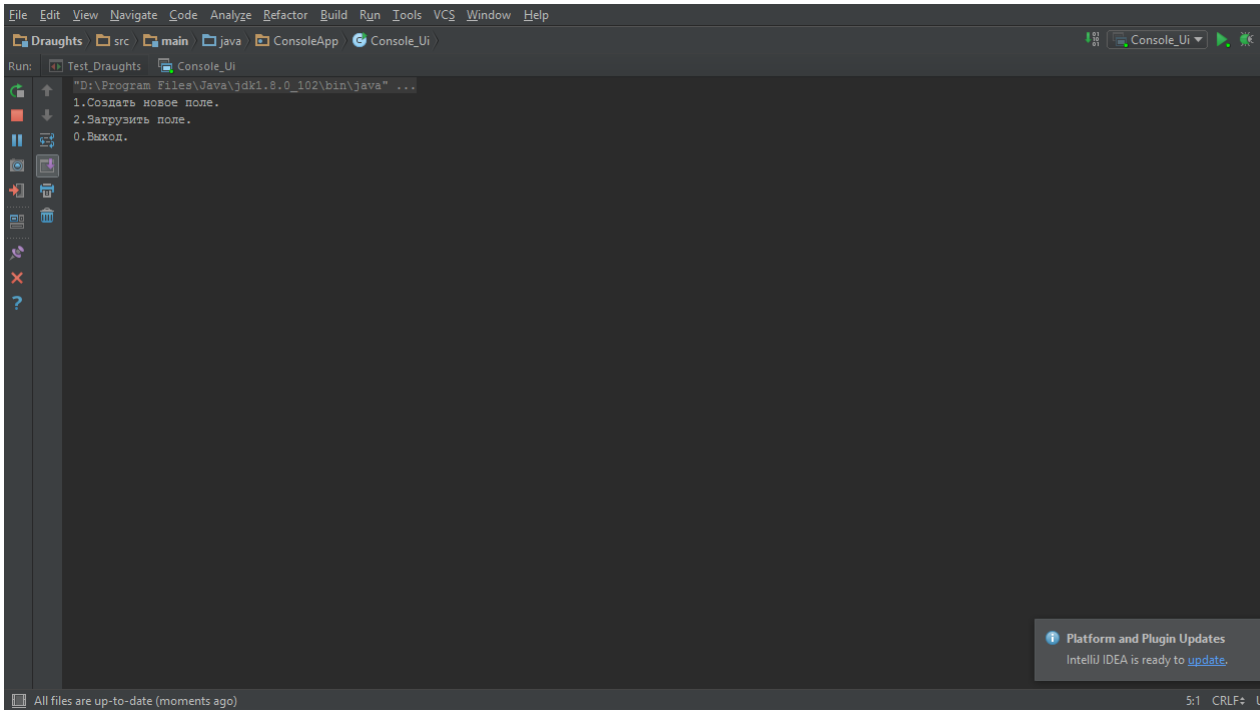

Рис. 3: Главное меню консольного приложения

На рис 3 представлено главное меню приложения. Есть возможность создать новое поле с шашками, загрузить поле из файла и выйти из программы.

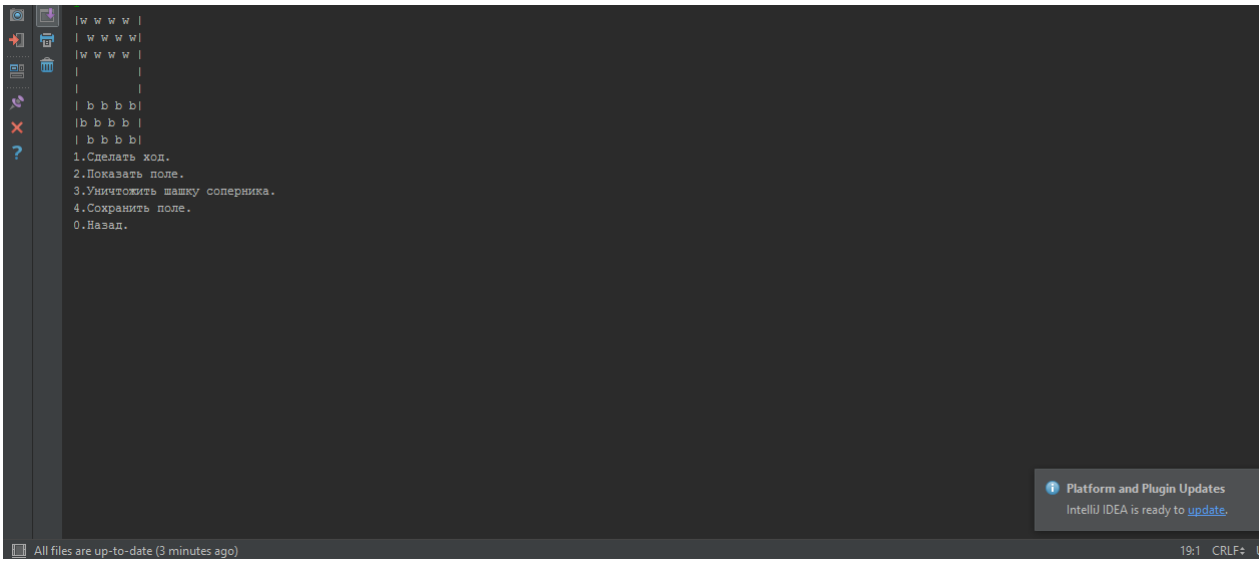

Рис. 4: Поле и второе меню в консольном приложении

На рис 4 показано поле и внизу меню, в котором можно передвинуть шашку, отобразить поле в консоли, уничтожить шашку противника и вернуться назад в главное меню.

3.3 Библиотека

Основные классы, выделенные в библиотеке:

- Класс Draught. 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

3.4 Графическое приложение

Графическое приложение позволяет играть через окна графического приложения.

Основные классы, выделенные в графическом приложении.

- Класс Frame. Главное окно приложения. Присутствуют кнопки «Новая игра», «Загрузить поле», «Выход».
- Класс Board. Окно, где присутствуют кнопки «Сохранить поле», «Назад», а также в этом окне происходит отрисовка поля .
- Класс BorderPanel. Класс отвечающий за отрисовку поля. Все действия над полем происходят именно в этом классе.

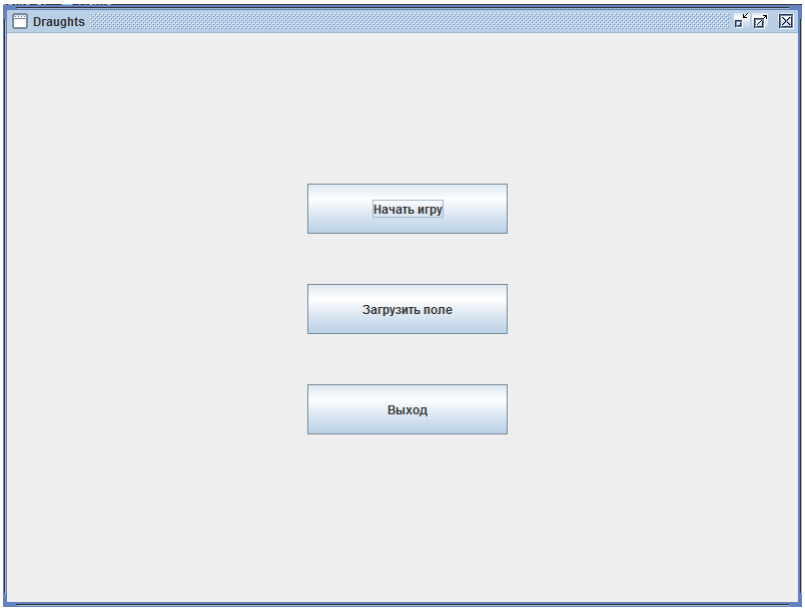

Рис. 5: Главное меню графического приложения

На рис 5 представлено главное окно приложения. В нём пользователю можно начать играть, загрузить поле из файла и выйти из игры.

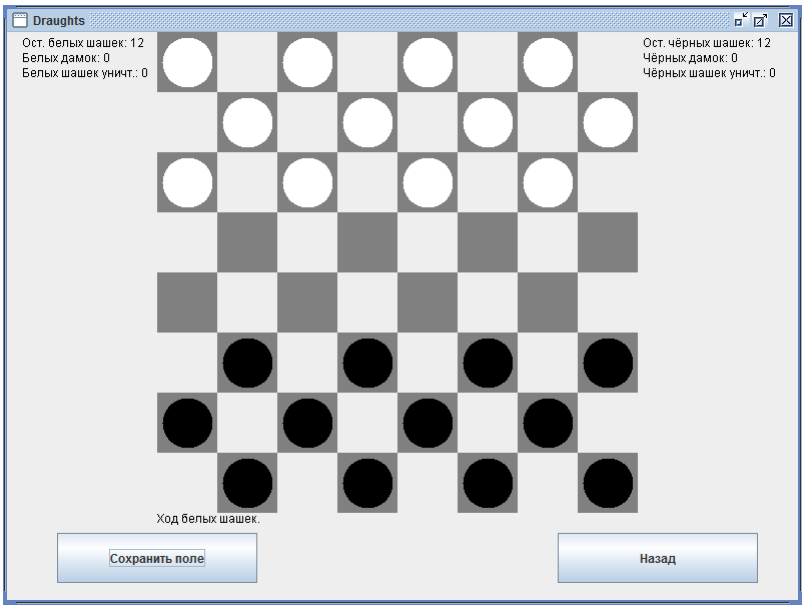

Рис. 6: Представление поля в графическом приложении

На рис 6 – окно с полем, внизу есть кнопки для сохранения поля в файл и возвращения в главное меню. В центре расположено поле с шашками, справа и слева текущие данные об игре для каждой из сторон, между полем и кнопками есть информация о том, чей ход в данный момент.

4 Процесс обеспечения качества и тестирование

4.1 Тестирование

Приложение содержит автоматические тесты. Протестированы некоторые основные функции. Проверяется проверяется свободна ли клетка, может ли шашка сделать данный ход и может ли шашка уничтожить данную шашку противника.

5 Вывод

По окончании семестра автор проекта научился писать программы на языке Java, 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 Консольное приложение

```
1  package ConsoleApp;
2
3  import Core.Api;
4  import Core.Draught;
5  import Core.Field;
6
7  import java.io.FileNotFoundException;
8  import java.io.IOException;
9  import java.util.Scanner;
10
11 public class Console_Ui {
12     public static void main(String[] args) {
13         Scanner s = new Scanner(System.in);
14         print_menu(s);
15     }
16
17     private static void print_menu(Scanner s) {
18         int choice = -1;
19         Field field = new Field();
20         while(choice != 0) {
21             System.out.println("Создать 1. новое поле.");
22             System.out.println("Загрузить 2. поле.");
23             System.out.println("Выход 0..");
24             choice = s.nextInt();
25             switch (choice)
26             {
27                 case 1:
28                     print_field(field);
29                     secondary_menu(field , s);
30                     break;
31                 case 2:
32                     try {
33                         field = Api.load_file();
34                     } catch (FileNotFoundException e) {
35                         System.out.println("Неудалось загрузить поле
    ↪ из файла.");
36                     }
37                     print_field(field);
38                     secondary_menu(field , s);
39                     break;
40                 case 0:
41                     break;
42                 default:
43                     System.out.println("Некорректная команда.");
44                     break;
45             }
46         }
47     }
48
49     private static void secondary_menu(Field field , Scanner s) {
50         int choice = -1;
51         while(choice != 0) {
52             System.out.println("Сделать 1. ход.");
53             System.out.println("Показать 2. поле.");
54             System.out.println("Уничтожить 3. шашку соперника.");
55             System.out.println("Сохранить 4. поле.");
56             System.out.println("Назад 0..");
57             choice = s.nextInt();
```

```
58              switch (choice)
59              {
60                  case 1:
61                      System.out.println("Введите координаты шашки.");
62                      int x_draught, y_draught;
63                      x_draught = s.nextInt();
64                      y_draught = s.nextInt();
65                      System.out.println("Введите координаты места.");
66                      int x_place, y_place;
67                      x_place = s.nextInt();
68                      y_place = s.nextInt();
69                      try {
70                          Api.move_draught(field, x_draught,
    ↪ y_draught, x_place, y_place);
71                      } catch (IllegalArgumentException e){
72                          System.out.println("Неверные координаты шага.
    ↪ ");
73                      }
74                      secondary_menu(field, s);
75                      break;
76                  case 2:
77                      print_field(field);
78                      secondary_menu(field, s);
79                      break;
80                  case 3:
81                      System.out.println("Введите координаты шашки.");
82                      int x_selected, y_selected;
83                      x_selected = s.nextInt();
84                      y_selected = s.nextInt();
85                      System.out.println("Введите координаты
    ↪ уничтожаемой шашки.");
86                      int x_destroyed, y_destroyed;
87                      x_destroyed = s.nextInt();
88                      y_destroyed = s.nextInt();
89                      try {
90                          Api.destroy_draught(field, x_selected,
    ↪ y_selected, x_destroyed, y_destroyed);
91                      } catch (IllegalArgumentException e){
92                          System.out.println("Неверные координаты
    ↪ уничтожаемой шашки.");
93                      }
94                      secondary_menu(field, s);
95                      break;
96                  case 4:
97                      try {
98                          Api.save_file(field);
99                      } catch (IOException e) {
100                         System.out.println("Неудалось сохранить поле
    ↪ в файл.");
101                     }
102                     secondary_menu(field,s);
103                     break;
104                 case 0:
105                     break;
106                 default:
107                     System.out.println("Некорректная команда.");
108                     break;
109             }
110             break;
111         }
112     }
113
```

```
114     private static void print_field(Field field) {
115         for (int i = 0; i <= 7; i++){
116             System.out.print("|");
117             for (int j = 0; j <= 7; j++){
118                 Draught draught = field.get_draught(i,j);
119                 if (draught == null) {
120                     System.out.print("␣");
121                 }else {
122                     if ((draught.get_color() == false) && (draught.
    ↪ get_type() == false)) {
123                         System.out.print("b");
124                     }
125                     if ((draught.get_color() == false) && (draught.
    ↪ get_type() == true)) {
126                         System.out.print("B");
127                     }
128                     if ((draught.get_color() == true) && (draught.
    ↪ get_type() == false)) {
129                         System.out.print("w");
130                     }
131                     if ((draught.get_color() == true) && (draught.
    ↪ get_type() == true)) {
132                         System.out.print("W");
133                     }
134                 }
135             }
136             System.out.print("|");
137             System.out.println();
138         }
139     }
140 }
```

6.2 Графическое приложение

```
1  package Graphic_Ui;
2
3  import java.awt.Dimension;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import java.io.FileNotFoundException;
7  import javax.swing.JButton;
8  import javax.swing.JFrame;
9  import javax.swing.JPanel;
10 import Core.Api;
11 import Core.Field;
12
13 public class Frame extends JFrame{
14
15     public static void main(String[] args) {
16         javax.swing.SwingUtilities.invokeLater(new Runnable() {
17             public void run() {
18                 JFrame.setDefaultLookAndFeelDecorated(true);
19                 Frame frame = new Frame();
20                 frame.setSize(800, 600);
21                 frame.setLocationRelativeTo(null);
22                 frame.setVisible(true);
23             }
24         });
25     }
26
27     public Frame() {
28         super("Draughts");
29         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
30         JPanel panel = new JPanel();
31         panel.setLayout(null);
32         JButton new_game_button = new JButton("Начать игру");
33         new_game_button.setSize(200,50);
34         new_game_button.setLocation(300,150);
35         panel.add(new_game_button);
36         JButton load_button = new JButton("Загрузить поле");
37         load_button.setSize(200,50);
38         load_button.setLocation(300,250);
39         panel.add(load_button);
40         JButton exit_button = new JButton("Выход");
41         exit_button.setSize(200,50);
42         exit_button.setLocation(300,350);
43         panel.add(exit_button);
44         ActionListener actionListenerNewGame = new NewGameListener
    ↪ ();
45         new_game_button.addActionListener(actionListenerNewGame);
46         ActionListener actionListenerLoad = new LoadListener();
47         load_button.addActionListener(actionListenerLoad);
48         ActionListener actionListenerExit = new ExitListener();
49         exit_button.addActionListener(actionListenerExit);
50         getContentPane().add(panel);
51         setPreferredSize(new Dimension(320, 100));
52     }
53
54     public class ExitListener implements ActionListener {
55         public void actionPerformed(ActionEvent e) {
56             System.exit(0);
57         }
58     }
59
```

```
60      public class LoadListener implements ActionListener {
61          public void actionPerformed(ActionEvent e) {
62              try {
63                  Field field = Api.load_file();
64                  Board board = new Board(field);
65                  board.setSize(800, 600);
66                  board.setLocationRelativeTo(null);
67                  board.setVisible(true);
68                  dispose();
69              } catch (FileNotFoundException e1) {
70                  System.out.println("Неудалось загрузить поле из файла."
    ↪ );
71              }
72          }
73      }
74
75      public class NewGameListener implements ActionListener {
76          public void actionPerformed(ActionEvent e) {
77              Field field = new Field();
78              Board board = new Board(field);
79              board.setSize(800, 600);
80              board.setLocationRelativeTo(null);
81              board.setVisible(true);
82              dispose();
83          }
84      }
85  }
```

```
1  package Graphic_Ui;
2
3  import javax.swing.*;
4  import Core.Api;
5  import Core.Field;
6  import java.awt.*;
7  import java.awt.event.ActionEvent;
8  import java.awt.event.ActionListener;
9  import java.io.IOException;
10
11 public class Board extends JFrame{
12     private Field field;
13
14     public Board(Field field){
15         super("Draughts");
16         this.field = field;
17         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18         BorderPanel panel = new BorderPanel(field ,this);
19         panel.setLayout(null);
20         JButton save_button = new JButton("Сохранить поле");
21         save_button.setSize(200,50);
22         save_button.setLocation(50,500);
23         panel.add(save_button);
24         JButton back_button = new JButton("Назад");
25         back_button.setSize(200,50);
26         back_button.setLocation(550,500);
27         panel.add(back_button);
28         ActionListener actionListenerBack = new BackListener();
29         back_button.addActionListener(actionListenerBack);
30         ActionListener actionListenerSave = new SaveListener();
31         save_button.addActionListener(actionListenerSave);
32         getContentPane().add(panel);
33         setPreferredSize(new Dimension(320, 100));
34     }
35
36     public class BackListener implements ActionListener {
37         public void actionPerformed(ActionEvent e) {
38             Frame frame = new Frame();
39             frame.setSize(800, 600);
40             frame.setLocationRelativeTo(null);
41             frame.setVisible(true);
42             dispose();
43         }
44     }
45
46     public class SaveListener implements ActionListener {
47         public void actionPerformed(ActionEvent e) {
48             try {
49                 Api.save_file(field);
50             } catch (IOException e1) {
51                 e1.printStackTrace();
52             }
53         }
54     }
55 }
```

```java
package Graphic_Ui;

import Core.Api;
import Core.Draught;
import Core.Field;
import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;

public class BorderPanel extends JPanel{
    private Field field;
    private Board board;
    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;

    public BorderPanel(Field field, Board board) {
        this.field = field;
        this.board = board;
        setOpaque(true);
        this.addMouseListener(new CustomListener());
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D)g;
        int i = 0,x = 150;
        g2d.setColor(Color.GRAY);
        while (i <= 7){
            g2d.fillRect(i*60 + x,0,60,60);
            i = i + 2;
        }
        i = 1;
        while (i <= 7){
            g2d.fillRect(i*60 + x,60,60,60);
            i = i + 2;
        }
        i = 0;
        while (i <= 7){
            g2d.fillRect(i*60 + x,120,60,60);
            i = i + 2;
        }
        i = 1;
        while (i <= 7){
            g2d.fillRect(i*60 + x,180,60,60);
            i = i + 2;
        }
        i = 0;
        while (i <= 7){
            g2d.fillRect(i*60 + x,240,60,60);
            i = i + 2;
        }
        i = 1;
        while (i <= 7) {
            g2d.fillRect(i*60 + x,300,60,60);
            i = i + 2;
        }
        i = 0;
```

```
63          while (i <= 7){
64              g2d.fillRect(i*60 + x,360,60,60);
65              i = i + 2;
66          }
67          i = 1;
68          while (i <= 7) {
69              g2d.fillRect(i*60 + x,420,60,60);
70              i = i + 2;
71          }
72          for(i = 0; i <= 7; ++i){
73              for (int j = 0; j <= 7; ++j){
74                  Draught draught = field.get_draught(i,j);
75                  if (draught != null) {
76                      if (draught.get_color() == true) {
77                          g2d.setColor(Color.WHITE);
78                          g2d.fillOval(j * 60 + x + 5, i * 60 + 5,
    ↪ 50, 50);
79                      }
80                      if (draught.get_color() == false) {
81                          g2d.setColor(Color.BLACK);
82                          g2d.fillOval(j * 60 + x + 5, i * 60 + 5,
    ↪ 50, 50);
83                      }
84                      if (draught.get_type() == true) {
85                          g2d.setColor(Color.YELLOW);
86                          int[] xPoints = {j * 60 + x + 10, j * 60 +
    ↪ x + 10, j * 60 + x + 20, j * 60 + x + 30, j * 60 + x + 40, j
    ↪  * 60 + x + 50, j * 60 + x + 50};
87                          int[] yPoints = {i * 60 + 45, i * 60 + 15,
    ↪ i * 60 + 30, i * 60 + 15, i * 60 + 30, i * 60 + 15, i * 60 +
    ↪  45};
88                          g2d.fillPolygon(xPoints,yPoints,7);
89                      }
90                  }
91              }
92          }
93          g2d.setColor(Color.RED);
94          g2d.fillOval(x_first_click * 60 + x + 5,y_first_click * 60
    ↪ + 5,50,50);
95          boolean color = field.get_color();
96          g2d.setColor(Color.BLACK);
97          if (color == true){
98              g2d.drawString("Ход белых шашек.",150,490);
99          }else {
100             g2d.drawString("Ход чёрных шашек.",150,490);
101         }
102         int[] statistics = Api.get_statistics(field);
103         g2d.drawString("Ост. белых шашек: " + statistics[0],15,15);
104         g2d.drawString("Белых дамок: " + statistics[1],15,30);
105         g2d.drawString("Белых шашек уничт.: " + statistics[2],15,45)
    ↪ ;
106         g2d.drawString("Ост. чёрных шашек: " + statistics[3],635,15)
    ↪ ;
107         g2d.drawString("Чёрных дамок: " + statistics[4],635,30);
108         g2d.drawString("Чёрных шашек уничт.: " + statistics
    ↪ [5],635,45);
109     }
110
111     public class CustomListener implements MouseListener {
112
113         public void mouseClicked(MouseEvent e) {
114             click = !click;
```

```
115             if (click == true){
116                 x_first_click = (e.getX() - 150) / 60;
117                 y_first_click = e.getY() / 60;
118                 if (field.get_draught(y_first_click,x_first_click).
    ↪ get_color() != field.get_color()){
119                     x_first_click = -10;
120                     y_first_click = -10;
121                     click = !click;
122                     JOptionPane.showMessageDialog(board, "Вы␣
    ↪ выбрали␣шашку␣не␣того␣цвета.", "Ошибка", JOptionPane.
    ↪ ERROR_MESSAGE);
123                 }else {
124                     repaint();
125                 }
126             }else {
127                 x_second_click = (e.getX() - 150) / 60;
128                 y_second_click = e.getY() / 60;
129                 try {
130                     Api.move_draught(field, y_first_click,
    ↪ x_first_click,y_second_click,x_second_click);
131                     repaint();
132                 }catch (IllegalArgumentException ex){
133                     try {
134                         Api.destroy_draught(field,y_first_click,
    ↪ x_first_click,y_second_click,x_second_click);
135                         repaint();
136                     }catch (IllegalArgumentException exc){
137                         JOptionPane.showMessageDialog(board, "
    ↪ Данный␣ход␣невозможен.", "Ошибка", JOptionPane.ERROR_MESSAGE);
138                     }
139                 }
140                 x_first_click = -10;
141                 y_first_click = -10;
142             }
143         }
144
145         public void mouseEntered(MouseEvent e) {
146
147         }
148
149         public void mouseExited(MouseEvent e) {
150
151         }
152
153         public void mousePressed(MouseEvent e) {
154
155         }
156
157         public void mouseReleased(MouseEvent e) {
158
159         }
160     }
161 }
```

6.3 Библиотека

```
1  package Core;
2
3  import java.io.File;
4  import java.io.FileNotFoundException;
5  import java.io.FileWriter;
6  import java.io.IOException;
7  import java.util.Scanner;
8
9  public class Api {
10
11     public static void move_draught(Field field, int x_draught, int
   ↪  y_draught, int x_place, int y_place) {
12         Draught draught = field.get_draught(x_draught, y_draught);
13         if (check_destruction_around(field) == true){
14             throw new IllegalArgumentException();
15         }
16         if (field.get_color() == draught.get_color()) {
17             if (draught.check_of_move(x_place, y_place) == false) {
18                 throw new IllegalArgumentException();
19             } else {
20                 if (field.check_free(x_place, y_place) == false) {
21                     throw new IllegalArgumentException();
22                 } else {
23                     field.set_null(x_draught, y_draught);
24                     draught.set_x(x_place);
25                     draught.set_y(y_place);
26                     field.set_draught(draught);
27                     if (draught.check_of_type() == true) {
28                         draught.set_type(true);
29                     }
30                 }
31             }
32         }else {
33             throw new IllegalArgumentException();
34         }
35         field.set_color(!field.get_color());
36     }
37
38     public static void destroy_draught(Field field, int x_selected,
   ↪  int y_selected, int x_destroyed, int y_destroyed) {
39         Draught draught = field.get_draught(x_selected, y_selected)
   ↪ ;
40         if (field.get_color() == draught.get_color()) {
41             if (field.check_destruction(x_selected, y_selected,
   ↪ x_destroyed, y_destroyed) == false) {
42                 throw new IllegalArgumentException();
43             } else {
44                 field.set_null(x_selected, y_selected);
45                 field.set_null(x_destroyed, y_destroyed);
46                 draught.set_x(x_selected + 2 * (x_destroyed -
   ↪ x_selected));
47                 draught.set_y(y_selected + 2 * (y_destroyed -
   ↪ y_selected));
48                 field.set_draught(draught);
49                 draught.check_of_type();
50                 if (draught.check_of_type() == true) {
51                     draught.set_type(true);
52                 }
53             }
54         }
```

```
55          if (check_destruction_around(field) == false) {
56              field.set_color(!field.get_color());
57          }
58      }
59
60      public static void save_file(Field field) throws IOException {
61          File file = new File("savefile.txt");
62          if(!file.exists()){
63              file.createNewFile();
64          }
65          FileWriter wrt = new FileWriter(file);
66          String lineSeparator = System.getProperty("line.separator")
    ↪ ;
67          for (int i = 0; i <= 7; ++i){
68              for (int j = 0; j <= 7; ++j){
69                  Draught draught = field.get_draught(i,j);
70                  if (draught == null){
71                      wrt.append("nl ");
72                  }else {
73                      if ((draught.get_color() == true) && (draught.
    ↪ get_type() == false)) {
74                          wrt.append("tf ");
75                      }
76                      if ((draught.get_color() == true) && (draught.
    ↪ get_type() == true)) {
77                          wrt.append("tt ");
78                      }
79                      if ((draught.get_color() == false) && (draught.
    ↪ get_type() == false)) {
80                          wrt.append("ff ");
81                      }
82                      if ((draught.get_color() == false) && (draught.
    ↪ get_type() == true)) {
83                          wrt.write("ft ");
84                      }
85                  }
86              }
87              wrt.append(lineSeparator);
88          }
89          if (field.get_color() == true){
90              wrt.append("tr");
91          }else {
92              wrt.append("fl");
93          }
94          wrt.close();
95      }
96
97      public static Field load_file() throws FileNotFoundException {
98          File file = new File("savefile.txt");
99          Scanner s = new Scanner(file);
100         String string;
101         Field field = new Field();
102         for (int i = 0; i <= 7; ++i){
103             for (int j = 0; j <= 7; ++j){
104                 string = s.next();
105                 if (string.equals("nl")){
106                     field.set_null(i,j);
107                 }
108                 if (string.equals("tt")){
109                     field.set_draught(new Draught(i,j,true,true));
110                 }
111                 if (string.equals("tf")){
```

```
112                     field.set_draught(new Draught(i,j,true,false));
113                 }
114                 if (string.equals("ff")){
115                     field.set_draught(new Draught(i,j,false,false))
    ↪ ;
116                 }
117                 if (string.equals("ft")){
118                     field.set_draught(new Draught(i,j,false,true));
119                 }
120             }
121         }
122         string = s.next();
123         if (string.equals("tr")){
124             field.set_color(true);
125         }else {
126             field.set_color(false);
127         }
128         return field;
129     }
130 
131     public static int[] get_statistics(Field field){
132         int[] statistics = {0,0,0,0,0,0};
133         for (int i = 0; i <= 7; ++i){
134             for (int j = 0; j <= 7; ++j){
135                 Draught draught = field.get_draught(i,j);
136                 if (draught != null) {
137                     if (draught.get_color() == true) {
138                         ++statistics[0];
139                         if (draught.get_type() == true) {
140                             ++statistics[1];
141                         }
142                     }else {
143                         ++statistics[3];
144                         if (draught.get_type() == true) {
145                             ++statistics[4];
146                         }
147                     }
148                 }
149             }
150         }
151         statistics[2] = 12 - statistics[0];
152         statistics[5] = 12 - statistics[3];
153         return statistics;
154     }
155 
156     private static boolean check_destruction_around(Field field){
157         for (int i = 0; i <= 7; ++i) {
158             for (int j = 0; j <= 7; ++j) {
159                 if ((field.get_draught(i,j) != null) && (field.
    ↪ get_draught(i,j).get_color() == field.get_color())) {
160                     if ((i + 1 <= 7) && (i + 1 >= 0) && (j + 1 <=
    ↪ 7) && (j + 1 >= 0) &&
161                             (field.check_destruction(i, j, i + 1, j
    ↪  + 1) == true)) {
162                         return true;
163                     }
164                     if ((i - 1 <= 7) && (i - 1 >= 0) && (j + 1 <=
    ↪ 7) && (j + 1 >= 0) &&
165                             (field.check_destruction(i, j, i - 1, j
    ↪  + 1) == true)) {
166                         return true;
167                     }
```

```
168                     if ((i + 1 <= 7) && (i + 1 >= 0) && (j - 1 <=
    ↪ 7) && (j - 1 >= 0) &&
169                             (field.check_destruction(i, j, i + 1, j
    ↪  - 1) == true)) {
170                         return true;
171                     }
172                     if ((i - 1 <= 7) && (i - 1 >= 0) && (j - 1 <=
    ↪ 7) && (j - 1 >= 0) &&
173                             (field.check_destruction(i, j, i - 1, j
    ↪  - 1) == true)) {
174                         return true;
175                     }
176                 }
177             }
178         }
179         return false;
180     }
181 }
```

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

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

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

```
141     }
142
143     public  Draught get_draught(int x, int y){return draughts[x][y
        ↪ ];}
144
145     public void set_draught(Draught draught){
146         draughts[draught.get_x()][draught.get_y()] = draught;
147     }
148
149     public void  set_null(int x, int y){
150         draughts[x][y] = null;
151     }
152
153     public void set_color(boolean color){this.color = color;}
154
155     public boolean get_color(){return color;}
156 }
```

6.4 Тесты

```
1  import Core.Draught;
2  import Core.Field;
3  import org.junit.Test;
4  import static org.junit.Assert.*;
5
6  public class Test_Draughts {
7      @Test
8      public void test_free_true(){
9          Field field = new Field();
10         assertTrue(field.check_free(5,0));
11     }
12     @Test
13     public void test_free_false(){
14         Field field = new Field();
15         assertFalse(field.check_free(5,1));
16     }
17     @Test
18     public void test_check_of_move_true(){
19         Field field = new Field();
20         Draught draught = field.get_draught(5,1);
21         assertTrue(draught.check_of_move(4,2));
22     }
23     @Test
24     public void test_check_of_move_false(){
25         Field field = new Field();
26         Draught draught = field.get_draught(5,1);
27         assertFalse(draught.check_of_move(4,3));
28     }
29     @Test
30     public void test_check_destruction_true(){
31         Field field = new Field();
32         Draught draught = new Draught(4,2,true,false);
33         field.set_draught(draught);
34         assertTrue(field.check_destruction(5,1,4,2));
35     }
36     @Test
37     public void test_check_destruction_false(){
38         Field field = new Field();
39         Draught draught = new Draught(4,2,true,true);
40         field.set_draught(draught);
41         assertFalse(field.check_destruction(4,2,5,1));
42     }
43 }
```