МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное образовательное учреждение высшего образования «Санкт-петербургский государственный морской технический университет»

ФАКУЛЬТЕТ ЦИФРОВЫХ ПРОМЫШЛЕННЫХ ТЕХНОЛОГИЙ

Кафедра Киберфизических систем

КУРСОВОЙ ПРОЕКТ

По дисциплине «Программирование»

Выполнила: Иноземцева Маргарита Васильевна

Проверил: Поделенюк Павел Петрович

Санкт-Петербург

2025

Оглавление

[1. Цели и формулировка задачи #](#__RefHeading___1)

2. Разработка классов и UML диаграмма……………………….………#

[3. Результаты работы #](#__RefHeading___2)

[**3.1. Реализация программы с использованием функционального программирования языка Python** #](#__RefHeading___3)

[3.1.1. Ход работы #](#__RefHeading___4)

[3.1.2. Демонстрация работы программы #](#__RefHeading___5)

[3.1.3. Листинг кода #](#__RefHeading___6)

**3.2. Реализация программы с использованием ООП языка**

**Python**……………………………………………………………………….#

3.2.1. Ход работы…………………………………………………...…….#

3.2.2. Демонстрация работы программы………………………………..#

3.2.3. Листинг кода……………………………………………………….#

[ЗАКЛЮЧЕНИЕ #](#__RefHeading___7)

[Список использованных источников #](#__RefHeading___8)

# Цели и формулировка задачи

Цели:

1. Анализ ходов фигуры на шахматной доске.
2. Работа с основами функционального программирования языка Python.
3. Работы с основами ООП языка Python.
4. Разработка классов и UML диаграмма.
5. Работы с пакетами Python.
6. Создание GUI приложения с использованием PySide6.

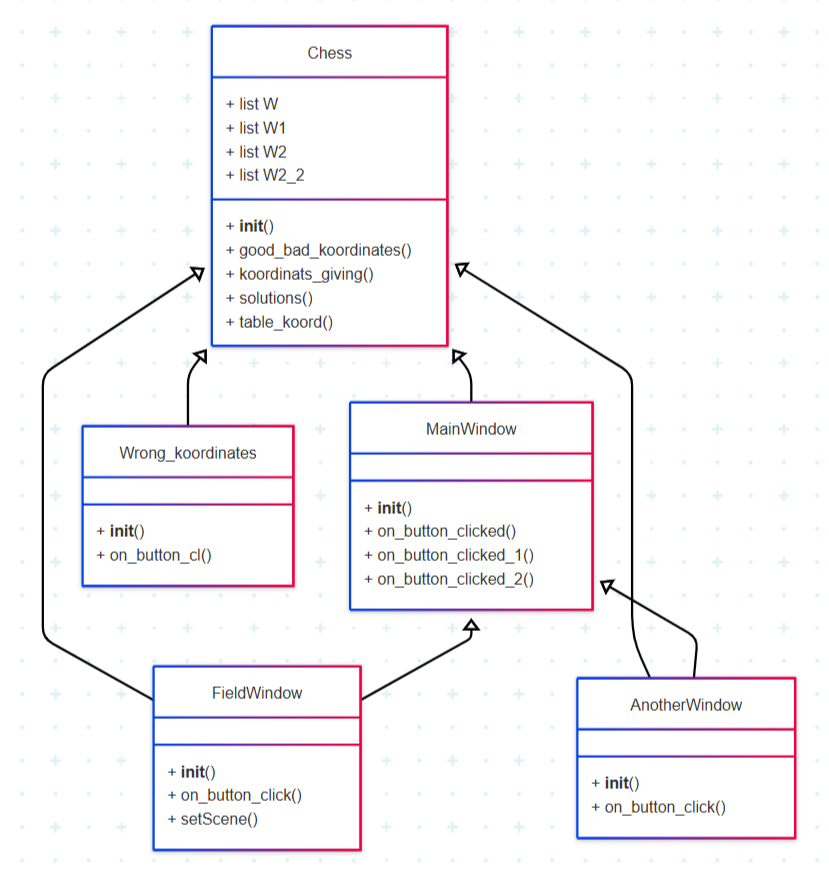
Задача:

Создать приложение на основе кода шахмат прошлого семестра, реализовать его с помощью PySide6.

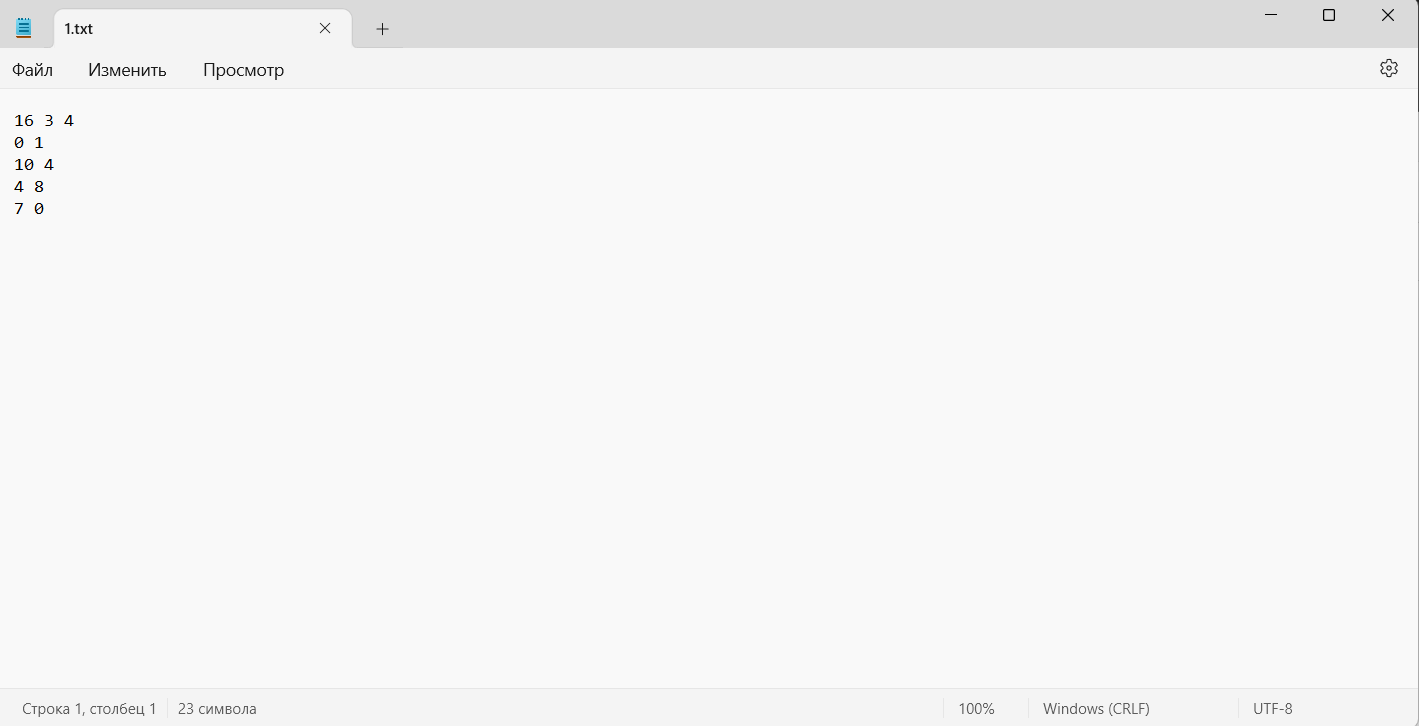
# Разработка классов и UML диаграмма

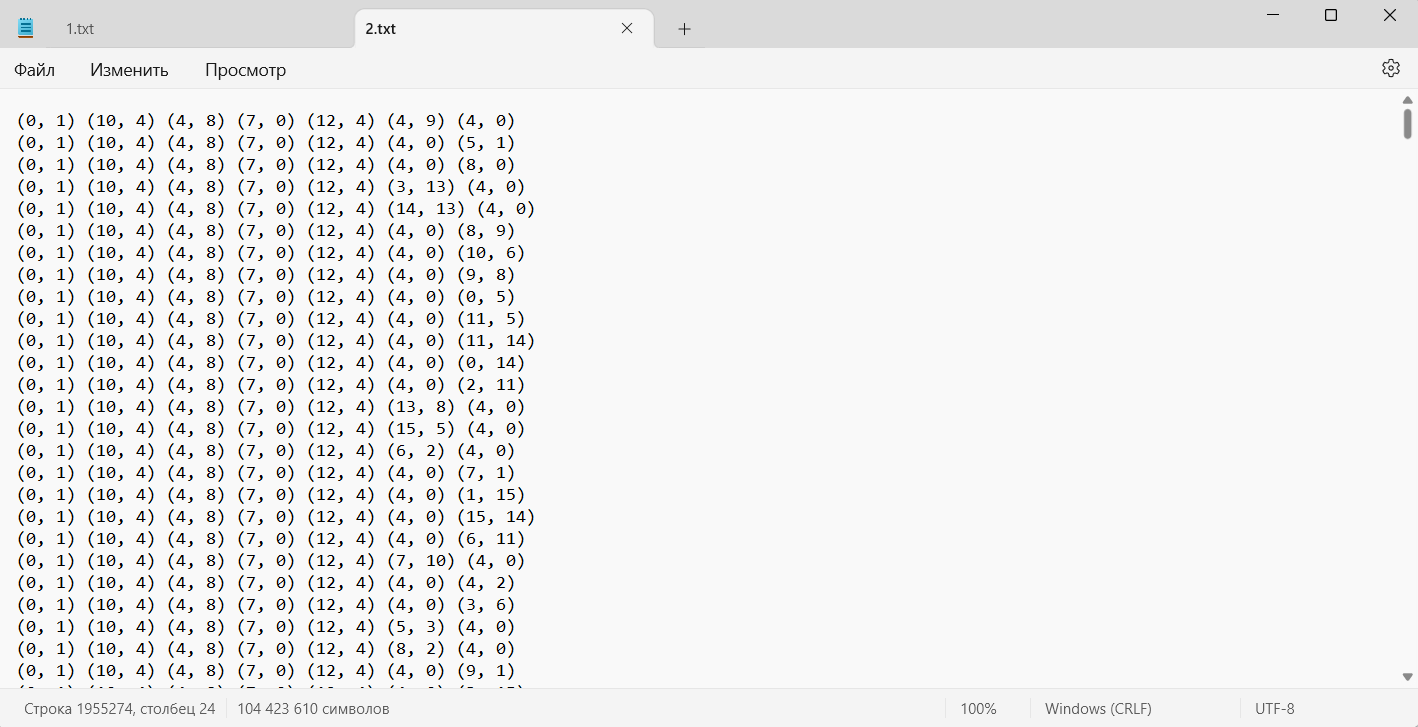
# Классы: Chess, Wrong\_koordinates, FileWindow, AnotherWindow, MainWindow.

UML диаграмма:



1. **Результаты работы**
   1. **Реализация программы с использованием функционального программирования языка Python**
      1. **Ход работы**
2. Создать классовые переменные: W, W1, W2, W2\_2 в виде списков, для запоминания различных координат клеток доски.
3. Создать вводную функцию \_\_init\_\_ для создания новых переменных, используемых в этом классе.
4. Создать функцию good\_bad\_koordinats для записи данных, полученных с главного окна, в файл.
5. Создать функцию koordinats\_giving для записи введенных пользователем координат фигуры в файл.
6. Создать функцию solutions, которая будет искать все возможные решения расстановки фигур на шахматной доске и при этом будет записывать их в файл.
7. Создать класс table\_koord для записи в списки, содержащие определенные значения клеток шахматной доски.
   * 1. **Демонстрация работы программы**





* + 1. **Листинг кода**

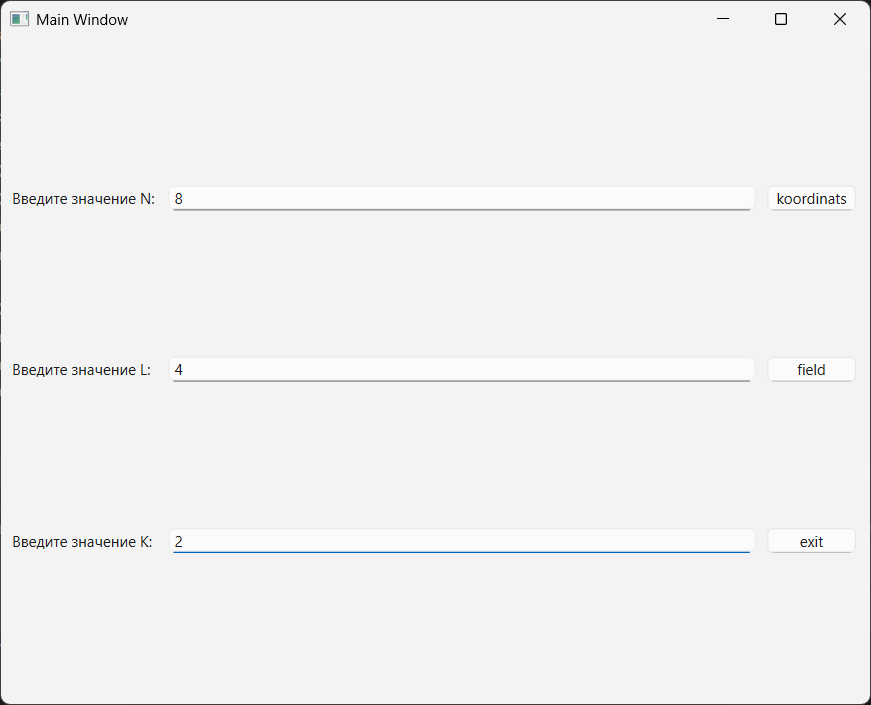
class Chess(object):  
 *''' Класс для работы с шахматами'''* W = []  
 W1 = []  
 W2 = []  
 W2\_2 = []  
 def \_\_init\_\_(self):  
 *''' Вводная функция'''* self.line\_edit = QLineEdit()  
 self.line\_edit\_1 = QLineEdit()  
 self.line\_edit\_2 = QLineEdit()  
 self.lineEdit\_list = []  
 self.one\_varient = []  
 self.open\_window = []  
 self.all\_field\_boxes = []  
  
 def good\_bad\_koordinats(self):  
 *''' Функция для записи данных в файл с главного окна'''* text = self.line\_edit.text()  
 text\_1 = self.line\_edit\_1.text()  
 text\_2 = self.line\_edit\_2.text()  
 with open('output.txt', 'w') as file:  
 file.write(f'{text} {text\_1} {text\_2}\n')  
 with open('output.txt', 'r') as file:  
 f = file.readlines()  
 if f[0].count(' ') == 2:  
 with open('input.txt', 'w') as file:  
 file.write(f'{text} {text\_1} {text\_2}\n')  
 if f[0].count(' ') < 2:  
 pass  
  
 def koordinats\_giving(self):  
 *''' Функция для записи введенных координат в файл input'''* with open('output.txt', 'r') as file:  
 f = file.readline()  
 N, L, K = map(int, f.strip().split())  
 with open('input.txt', 'w') as file:  
 file.write(f'{N} {L} {K}\n')  
 listing = []  
 \_list = [lineEdit.text() for lineEdit in self.lineEdit\_list]  
 if (len(\_list) == K) and ('' not in \_list):  
 for j in \_list:  
 k = 0  
 for i in range(len(j)):  
 if j[i] == ' ':  
 k += i  
 listing.append((str(int(j[0:k]) - 1) + ' ' + str(int(j[k + 1:]))))  
 listing.append((str(int(j[0:k]) + 1) + ' ' + str(int(j[k + 1:]))))  
 listing.append((str(int(j[0:k]) + 1) + ' ' + str(int(j[k + 1:]) + 2)))  
 listing.append((str(int(j[0:k]) - 1) + ' ' + str(int(j[k + 1:]) + 2)))  
 listing.append((str(int(j[0:k]) - 1) + ' ' + str(int(j[k + 1:]) - 2)))  
 listing.append((str(int(j[0:k]) + 1) + ' ' + str(int(j[k + 1:]) - 2)))  
  
 with open('input.txt', 'w') as f:  
 f.write(f'{N} {L} {K}\n')  
 for k in \_list:  
 if k in listing:  
 error = Wrong\_koordinates()  
 error.show()  
 self.open\_window.append(error)  
 \_list.remove(k)  
 if len(\_list) == K:  
 f.write(f'{k}\n')  
  
 def solutions(self):  
 *''' Функция для записи решеений в файл output'''* figure\_stay = []  
 figure\_stay\_go = []  
 f\_f = []  
 with open('input.txt', 'r') as file:  
 f = file.readlines()  
 N, L, K = map(int, f[0].strip().split())  
 for i in f[1:]:  
 k = 0  
 for j in range(len(i)):  
 if i[j] == ' ':  
 k += j  
 f\_f.append((int(i[0:k]), int(i[k + 1:])))  
 figure\_stay.append((int(i[0:k]), int(i[k + 1:])))  
 if int(i[k + 1:]) - 1 >= 0:  
 f\_f.append((int(i[0:k]), int(i[k + 1:]) - 1))  
 figure\_stay\_go.append((int(i[0:k]), int(i[k + 1:]) - 1))  
 f\_f.append((int(i[0:k]), int(i[k + 1:]) + 1))  
 figure\_stay\_go.append((int(i[0:k]), int(i[k + 1:]) + 1))  
 if int(i[0:k]) - 2 >= 0:  
 f\_f.append((int(i[0:k]) - 2, int(i[k + 1:]) + 1))  
 figure\_stay\_go.append((int(i[0:k]) - 2, int(i[k + 1:]) + 1))  
 if int(i[0:k]) - 2 >= 0 and int(i[k + 1:]) - 1 >= 0:  
 f\_f.append((int(i[0:k]) - 2, int(i[k + 1:]) - 1))  
 figure\_stay\_go.append((int(i[0:k]) - 2, int(i[k + 1:]) - 1))  
 if int(i[k + 1:]) - 1 >= 0:  
 f\_f.append((int(i[0:k]) + 2, int(i[k + 1:]) - 1))  
 figure\_stay\_go.append((int(i[0:k]) + 2, int(i[k + 1:]) - 1))  
 f\_f.append((int(i[0:k]) + 2, int(i[k + 1:]) + 1))  
 figure\_stay\_go.append((int(i[0:k]) + 2, int(i[k + 1:]) + 1))  
  
 self.all\_field\_boxes = [(y, x) for y in range(N) for x in range(N)]  
  
 free\_field\_boxes = [(i[1], i[0]) for i in (set(self.all\_field\_boxes) - set(f\_f))]  
  
 with open('output.txt', 'w') as file\_1:  
 all\_varients = []  
 all\_combinations = itertools.combinations(free\_field\_boxes, L)  
 for i in all\_combinations:  
 all\_varients.append(i)  
 if len(all\_varients) > 0:  
 for i in all\_varients:  
 while len(i) > 0:  
 z = []  
 for j in range(len(i)):  
 if (int(i[j][0]) - 1, int(i[j][1])) in i: z.append((int(i[j][0]) - 1, int(i[j][1])))  
 if (int(i[j][0]) + 1, int(i[j][1])) in i: z.append((int(i[j][0]) + 1, int(i[j][1])))  
 if (int(i[j][0]) + 1, int(i[j][1]) - 2) in i: z.append((int(i[j][0]) + 1, int(i[j][1]) - 2))  
 if (int(i[j][0]) - 1, int(i[j][1]) - 2) in i: z.append((int(i[j][0]) - 1, int(i[j][1]) - 2))  
 if (int(i[j][0]) - 1, int(i[j][1]) + 2) in i: z.append((int(i[j][0]) - 1, int(i[j][1]) + 2))  
 if (int(i[j][0]) + 1, int(i[j][1]) + 2) in i: z.append((int(i[j][0]) + 1, int(i[j][1]) + 2))  
 self.one\_varient.append((set(i) - set(z)))  
 if len(list(set(i) - set(z))) == L:  
 for p in figure\_stay:  
 file\_1.write(str(p[::-1]) + ' ')  
 for p1 in list(set(i) - set(z)):  
 file\_1.write(str(p1) + ' ')  
 file\_1.write('\n')  
 break  
 else:  
 file\_1.write("No solutions!")  
 def table\_koord(self):  
 *''' Функция для записи данных в списки для шахматной доски'''* with open('input.txt', 'r') as file\_3:  
 with open('output.txt', 'r') as file\_4:  
 f1 = file\_4.readlines()  
 f = file\_3.readlines()  
  
 if ('No solutions!' not in f1) and (f[0][0] != ' '):  
 N, L, K = map(int, f[0].strip().split())  
 K\_K = [(-1, 0), (1, 0), (1, -2), (-1, -2), (-1, 2), (1, 2)]  
 f12 = f1[0].replace(') (', ')\*(').split('\*')  
 l = [ast.literal\_eval(i) for i in f12]  
 for i in set(l[0:K]):  
 for j in set(K\_K):  
 Chess.W.append(i)  
 Chess.W1.append((int(i[0]) + int(j[0]), int(i[1]) + int(j[1])))  
 for i in set(l[K:]):  
 for j in set(K\_K):  
 Chess.W2.append(i)  
 Chess.W1.append((int(i[0]) + int(j[0]), int(i[1]) + int(j[1])))  
 for i in set(self.all\_field\_boxes):  
 if (i not in Chess.W) and (i not in Chess.W2) and (i not in Chess.W1):  
 Chess.W2\_2.append(i)

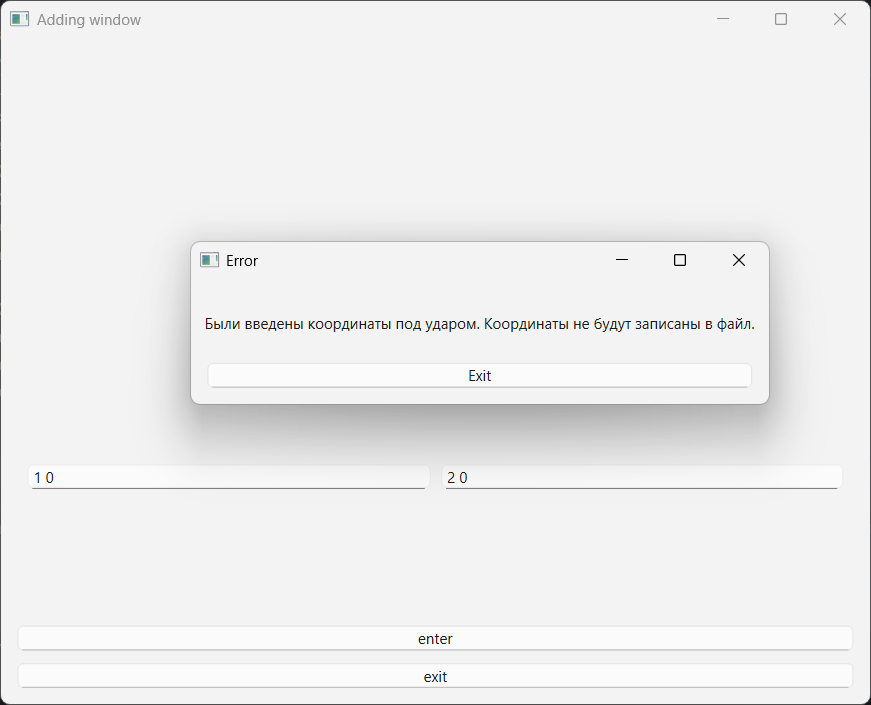
**3.2. Реализация программы с использованием ООП языка**

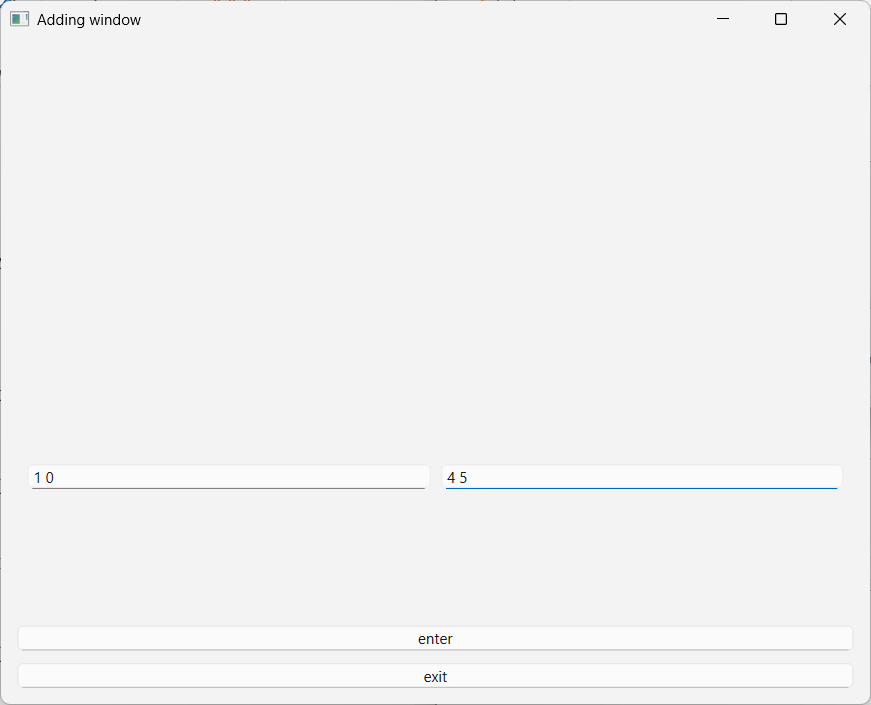
**Python**

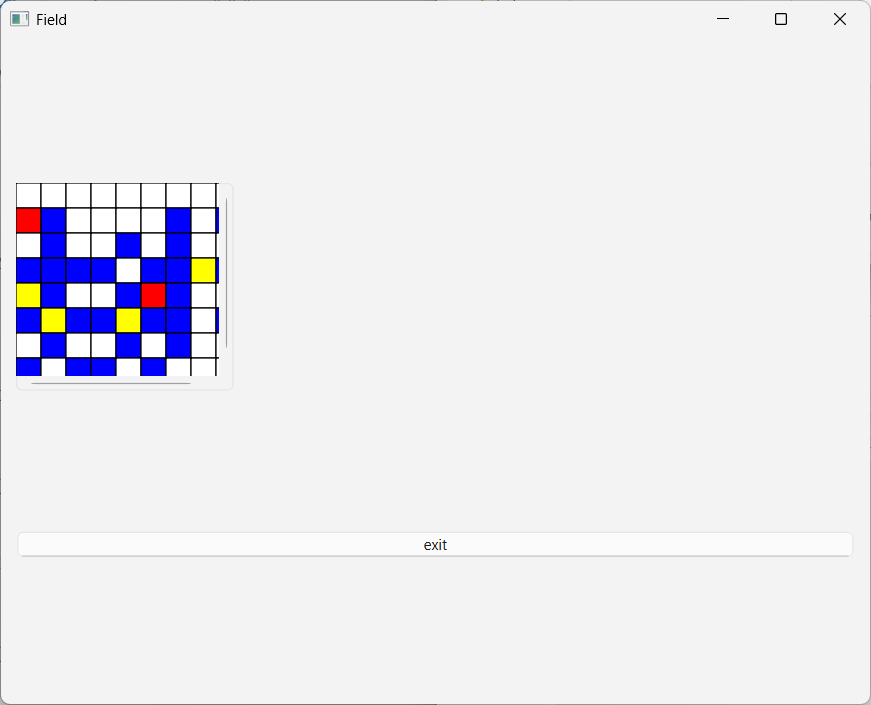
* + 1. **Ход Работы**

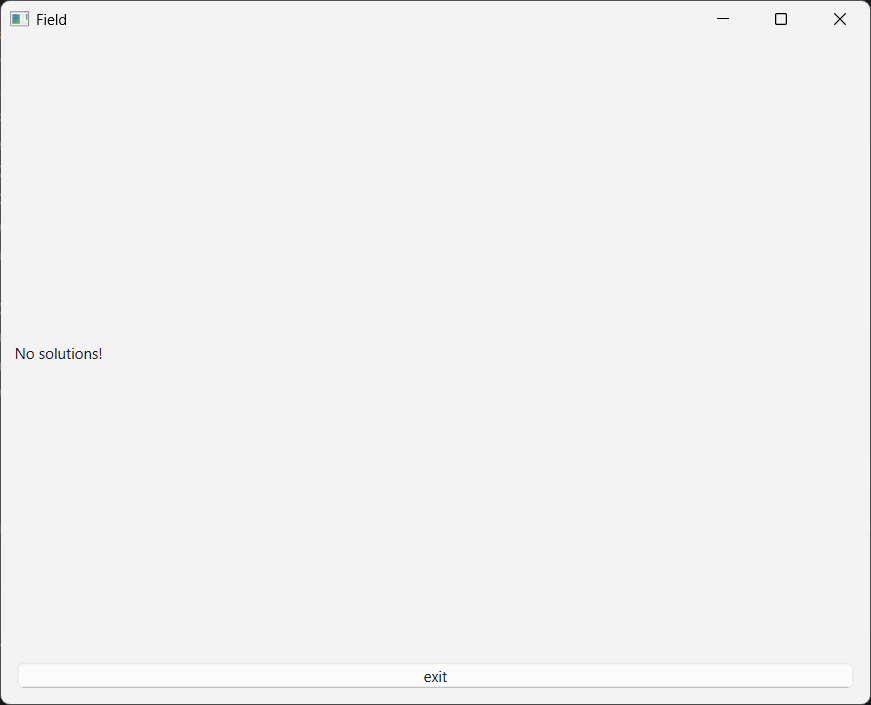
1. Создать класс MainWindow для создания главного окна приложения.
2. В классе MainWindow создать вводную функцию \_\_init\_\_ для декорации окна.
3. В классе MainWindow создать функцию on\_button\_clicked для того, чтобы при нажатии на кнопку заданные в первом окне данные считывались и открывали правильно сконструированное окно ввода координат.
4. В классе MainWindow создать функцию on\_button\_clicked\_1 для того, чтобы при нажатии на эту кнопку записывались найденные решения расстановки фигур в файл “output.txt” и выводилась доска одного из этих решений.
5. В классе MainWindow создать функцию on\_button\_clickes\_2 для того, чтобы при нажатии на кнопку окно автоматически закрывалось.
6. Создать класс AnotherWindow для создания окна ввода координат.
7. В классе AnotherWindow создать вводную функцию \_\_init\_\_ для декорации окна с кнопками, одна из которых записывает введенные координаты в файл.
8. В классе AnotherWindow создать функцию on\_button\_click для того, чтобы при нажатии на кнопку окно автоматически закрывалось.
9. Создать класс Wrong\_koordinates для создания окна при вводе координат под ударом.
10. В классе Wrong\_koordinates создать вводную функцию \_\_init\_\_ для декорирования окна.
11. В классе Wrong\_koordinates создать функцию on\_button\_cl для того, чтобы при нажатии на кнопку окно автоматически закрывалось.
12. Создать класс FieldWindow для создания окна с выводом доски или сообщения о том, что нет решений.
13. В классе FieldWindow создать функцию \_\_init\_\_ для декорации окна, а так же если есть решения, то оформления шахматной доски с помощью цветных полей.
14. В классе FieldWindow создать функцию on\_button\_click для того, чтобы при нажатии на кнопку окно автоматически закрывалось.
15. В классе FieldWindow создать функцию setScene для того, чтобы работала команда setScene.
16. Прописать условия по заданию.
17. Привести программу в действие.
    * 1. **Демонстрация работы программы**











* + 1. **Листинг кода**

import itertools  
import sys  
import ast  
from PySide6.QtCore import QRegularExpression, Qt  
from PySide6.QtGui import QRegularExpressionValidator, QColor, QPainter, QBrush  
from PySide6.QtWidgets import QLineEdit, QApplication, QGraphicsScene, QGraphicsView, QGraphicsRectItem, QWidget, QLabel, QVBoxLayout, QPushButton, QMainWindow, QGridLayout, QTableWidget, QTableWidgetItem  
  
class Chess(object):  
 *''' Класс для работы с шахматами'''* W = []  
 W1 = []  
 W2 = []  
 W2\_2 = []  
 def \_\_init\_\_(self):  
 *''' Вводная функция'''* self.line\_edit = QLineEdit()  
 self.line\_edit\_1 = QLineEdit()  
 self.line\_edit\_2 = QLineEdit()  
 self.lineEdit\_list = []  
 self.one\_varient = []  
 self.open\_window = []  
 self.all\_field\_boxes = []  
  
 def good\_bad\_koordinats(self):  
 *''' Функция для записи данных в файл с главного окна'''* text = self.line\_edit.text()  
 text\_1 = self.line\_edit\_1.text()  
 text\_2 = self.line\_edit\_2.text()  
 with open('output.txt', 'w') as file:  
 file.write(f'{text} {text\_1} {text\_2}\n')  
 with open('output.txt', 'r') as file:  
 f = file.readlines()  
 if f[0].count(' ') == 2:  
 with open('input.txt', 'w') as file:  
 file.write(f'{text} {text\_1} {text\_2}\n')  
 if f[0].count(' ') < 2:  
 pass  
  
 def koordinats\_giving(self):  
 *''' Функция для записи введенных координат в файл input'''* with open('output.txt', 'r') as file:  
 f = file.readline()  
 N, L, K = map(int, f.strip().split())  
 with open('input.txt', 'w') as file:  
 file.write(f'{N} {L} {K}\n')  
 listing = []  
 \_list = [lineEdit.text() for lineEdit in self.lineEdit\_list]  
 if (len(\_list) == K) and ('' not in \_list):  
 for j in \_list:  
 k = 0  
 for i in range(len(j)):  
 if j[i] == ' ':  
 k += i  
 listing.append((str(int(j[0:k]) - 1) + ' ' + str(int(j[k + 1:]))))  
 listing.append((str(int(j[0:k]) + 1) + ' ' + str(int(j[k + 1:]))))  
 listing.append((str(int(j[0:k]) + 1) + ' ' + str(int(j[k + 1:]) + 2)))  
 listing.append((str(int(j[0:k]) - 1) + ' ' + str(int(j[k + 1:]) + 2)))  
 listing.append((str(int(j[0:k]) - 1) + ' ' + str(int(j[k + 1:]) - 2)))  
 listing.append((str(int(j[0:k]) + 1) + ' ' + str(int(j[k + 1:]) - 2)))  
  
 with open('input.txt', 'w') as f:  
 f.write(f'{N} {L} {K}\n')  
 for k in \_list:  
 if k in listing:  
 error = Wrong\_koordinates()  
 error.show()  
 self.open\_window.append(error)  
 \_list.remove(k)  
 if len(\_list) == K:  
 f.write(f'{k}\n')  
  
 def solutions(self):  
 *''' Функция для записи решеений в файл output'''* figure\_stay = []  
 figure\_stay\_go = []  
 f\_f = []  
 with open('input.txt', 'r') as file:  
 f = file.readlines()  
 N, L, K = map(int, f[0].strip().split())  
 for i in f[1:]:  
 k = 0  
 for j in range(len(i)):  
 if i[j] == ' ':  
 k += j  
 f\_f.append((int(i[0:k]), int(i[k + 1:])))  
 figure\_stay.append((int(i[0:k]), int(i[k + 1:])))  
 if int(i[k + 1:]) - 1 >= 0:  
 f\_f.append((int(i[0:k]), int(i[k + 1:]) - 1))  
 figure\_stay\_go.append((int(i[0:k]), int(i[k + 1:]) - 1))  
 f\_f.append((int(i[0:k]), int(i[k + 1:]) + 1))  
 figure\_stay\_go.append((int(i[0:k]), int(i[k + 1:]) + 1))  
 if int(i[0:k]) - 2 >= 0:  
 f\_f.append((int(i[0:k]) - 2, int(i[k + 1:]) + 1))  
 figure\_stay\_go.append((int(i[0:k]) - 2, int(i[k + 1:]) + 1))  
 if int(i[0:k]) - 2 >= 0 and int(i[k + 1:]) - 1 >= 0:  
 f\_f.append((int(i[0:k]) - 2, int(i[k + 1:]) - 1))  
 figure\_stay\_go.append((int(i[0:k]) - 2, int(i[k + 1:]) - 1))  
 if int(i[k + 1:]) - 1 >= 0:  
 f\_f.append((int(i[0:k]) + 2, int(i[k + 1:]) - 1))  
 figure\_stay\_go.append((int(i[0:k]) + 2, int(i[k + 1:]) - 1))  
 f\_f.append((int(i[0:k]) + 2, int(i[k + 1:]) + 1))  
 figure\_stay\_go.append((int(i[0:k]) + 2, int(i[k + 1:]) + 1))  
  
 self.all\_field\_boxes = [(y, x) for y in range(N) for x in range(N)]  
  
 free\_field\_boxes = [(i[1], i[0]) for i in (set(self.all\_field\_boxes) - set(f\_f))]  
  
 with open('output.txt', 'w') as file\_1:  
 all\_varients = []  
 all\_combinations = itertools.combinations(free\_field\_boxes, L)  
 for i in all\_combinations:  
 all\_varients.append(i)  
 if len(all\_varients) > 0:  
 for i in all\_varients:  
 while len(i) > 0:  
 z = []  
 for j in range(len(i)):  
 if (int(i[j][0]) - 1, int(i[j][1])) in i: z.append((int(i[j][0]) - 1, int(i[j][1])))  
 if (int(i[j][0]) + 1, int(i[j][1])) in i: z.append((int(i[j][0]) + 1, int(i[j][1])))  
 if (int(i[j][0]) + 1, int(i[j][1]) - 2) in i: z.append((int(i[j][0]) + 1, int(i[j][1]) - 2))  
 if (int(i[j][0]) - 1, int(i[j][1]) - 2) in i: z.append((int(i[j][0]) - 1, int(i[j][1]) - 2))  
 if (int(i[j][0]) - 1, int(i[j][1]) + 2) in i: z.append((int(i[j][0]) - 1, int(i[j][1]) + 2))  
 if (int(i[j][0]) + 1, int(i[j][1]) + 2) in i: z.append((int(i[j][0]) + 1, int(i[j][1]) + 2))  
 self.one\_varient.append((set(i) - set(z)))  
 if len(list(set(i) - set(z))) == L:  
 for p in figure\_stay:  
 file\_1.write(str(p[::-1]) + ' ')  
 for p1 in list(set(i) - set(z)):  
 file\_1.write(str(p1) + ' ')  
 file\_1.write('\n')  
 break  
 else:  
 file\_1.write("No solutions!")  
 def table\_koord(self):  
 *''' Функция для записи данных в списки для шахматной доски'''* with open('input.txt', 'r') as file\_3:  
 with open('output.txt', 'r') as file\_4:  
 f1 = file\_4.readlines()  
 f = file\_3.readlines()  
  
 if ('No solutions!' not in f1) and (f[0][0] != ' '):  
 N, L, K = map(int, f[0].strip().split())  
 K\_K = [(-1, 0), (1, 0), (1, -2), (-1, -2), (-1, 2), (1, 2)]  
 f12 = f1[0].replace(') (', ')\*(').split('\*')  
 l = [ast.literal\_eval(i) for i in f12]  
 for i in set(l[0:K]):  
 for j in set(K\_K):  
 Chess.W.append(i)  
 Chess.W1.append((int(i[0]) + int(j[0]), int(i[1]) + int(j[1])))  
 for i in set(l[K:]):  
 for j in set(K\_K):  
 Chess.W2.append(i)  
 Chess.W1.append((int(i[0]) + int(j[0]), int(i[1]) + int(j[1])))  
 for i in set(self.all\_field\_boxes):  
 if (i not in Chess.W) and (i not in Chess.W2) and (i not in Chess.W1):  
 Chess.W2\_2.append(i)  
  
class Wrong\_koordinates(QWidget):  
 *''' Класс для создания окна при вводе неправильных координат фигур'''* def \_\_init\_\_(self):  
 *''' Начальная функция для декорирования окна'''* super().\_\_init\_\_()  
 self.setWindowTitle("Error")  
 self.resize(100, 100)  
 self.layout = QVBoxLayout()  
  
 self.label = QLabel("Были введены координаты под ударом. Координаты не будут записаны в файл.")  
  
 self.button = QPushButton("Exit")  
 self.button.clicked.connect(self.on\_button\_cl)  
  
 self.layout.addWidget(self.label)  
 self.layout.addWidget(self.button)  
 self.setLayout(self.layout)  
  
 def on\_button\_cl(self):  
 *''' Функция для кнопки закрытия'''* Wrong\_koordinates.deleteLater(self)  
  
class FieldWindow(QWidget, Chess):  
 *''' Класс для создания окна с доской'''* def \_\_init\_\_(self):  
 *''' Функция для декорирования окна'''* super().\_\_init\_\_()  
 self.setWindowTitle("Field")  
 self.resize(695, 533)  
  
 self.show()  
  
 self.vBox = QVBoxLayout()  
 self.setLayout(self.vBox)  
 with open('input.txt', 'r') as file\_3:  
 with open('output.txt', 'r') as file\_4:  
 f1 = file\_4.readlines()  
 f = file\_3.readlines()  
 k = ''  
 for i in f[0]:  
 while i != ' ':  
 k += str(i)  
 break  
  
 if 'No solutions!' in f1:  
 self.label = QLabel("No solutions!")  
 self.button = QPushButton("exit")  
 self.button.clicked.connect(self.on\_button\_click)  
 self.vBox.addWidget(self.label)  
 self.vBox.addWidget(self.button)  
 else:  
 self.scene = QGraphicsScene()  
 self.view = QGraphicsView()  
 self.view.setScene(self.scene)  
 self.view.setRenderHint(QPainter.Antialiasing)  
 self.vBox.addWidget(self.view)  
 self.setScene(self.scene)  
  
 self.view.setBackgroundBrush(QBrush(QColor("#FFFFFF")))  
 self.view.setSceneRect(0, 0, int(k) / 5 + 6, int(k) / 5 - 2)  
 self.scene.setSceneRect(self.view.sceneRect())  
  
 for i in range(int(k)):  
 for j in range(int(k)):  
 if (j, i) in Chess.W:  
 rect\_item = QGraphicsRectItem(j\*20, i\*20, 20, 20)  
 rect\_item.setBrush(Qt.red)  
 self.scene.addItem(rect\_item)  
 elif (j, i) in Chess.W2:  
 rect\_item = QGraphicsRectItem(j \* 20, i \* 20, 20, 20)  
 rect\_item.setBrush(Qt.yellow)  
 self.scene.addItem(rect\_item)  
 elif (j, i) in Chess.W1:  
 rect\_item = QGraphicsRectItem(j \* 20, i \* 20, 20, 20)  
 rect\_item.setBrush(Qt.blue)  
 self.scene.addItem(rect\_item)  
 else:  
 rect\_item = QGraphicsRectItem(j \* 20, i \* 20, 20, 20)  
 rect\_item.setBrush(Qt.white)  
 self.scene.addItem(rect\_item)  
  
 self.button = QPushButton("exit")  
 self.button.clicked.connect(self.on\_button\_click)  
 self.view.setFixedSize(self.view.sizeHint())  
 self.vBox.addWidget(self.view)  
 self.vBox.addWidget(self.button)  
  
 def on\_button\_click(self):  
 *''' Функция для кнопки закрытия окна'''* FieldWindow.deleteLater(self)  
  
 def setScene(self, scene):  
 *''' Функция для работы setScene'''* pass  
  
class AnotherWindow(QWidget, Chess):  
 *''' Класс для создания окна с ввода координат'''* def \_\_init\_\_(self):  
 *''' Фунуия для декорирования окна'''* super().\_\_init\_\_()  
 self.setWindowTitle("Adding window")  
 self.resize(695, 533)  
 self.widget = QWidget()  
 self.layout = QGridLayout(self.widget)  
  
 self.open\_window = []  
  
 self.label = QLabel()  
 self.button = QPushButton("enter")  
 self.button\_1 = QPushButton("exit")  
 self.button.clicked.connect(self.koordinats\_giving)  
 self.button\_1.clicked.connect(self.on\_button\_click)  
  
 with open('output.txt', 'r') as file:  
 f = file.readline()  
 N, L, K = map(int, f.strip().split())  
 if K > 1:  
 count\_K = 0  
 for i in range(1, K):  
 for j in range(0, K):  
 while count\_K < K:  
 self.line\_edit = QLineEdit()  
 self.lineEdit\_list.append(self.line\_edit)  
 self.layout.addWidget(self.line\_edit, i, j)  
 count\_K += 1  
 break  
 else:  
 self.line\_edit = QLineEdit()  
 self.lineEdit\_list.append(self.line\_edit)  
 self.layout.addWidget(self.line\_edit, 0, 0)  
  
 self.layout1 = QVBoxLayout()  
 self.layout1.addWidget(self.label)  
 self.layout1.addWidget(self.widget)  
 self.layout1.addWidget(self.button)  
 self.layout1.addWidget(self.button\_1)  
 self.setLayout(self.layout1)  
  
 def on\_button\_click(self):  
 *''' Функция для кнопки закрытия окна'''* AnotherWindow.deleteLater(self)  
  
class MainWindow(QMainWindow, Chess):  
 *''' Класс для создания главного окна'''* def \_\_init\_\_(self):  
 *''' Функция для декорирования главного окна'''* super().\_\_init\_\_()  
 self.setWindowTitle("Main Window")  
 self.resize(695, 533)  
  
 self.open\_window = []  
  
 self.label = QLabel("Введите значение N: ")  
 self.label\_1 = QLabel("Введите значение L: ")  
 self.label\_2 = QLabel("Введите значение K: ")  
  
 self.validator = QRegularExpressionValidator(QRegularExpression("^[0-9]+$"))  
 self.line\_edit = QLineEdit()  
 self.line\_edit.setValidator(self.validator)  
 self.line\_edit.setPlaceholderText("Введите размер поля")  
 self.line\_edit\_1 = QLineEdit()  
 self.line\_edit\_1.setValidator(self.validator)  
 self.line\_edit\_1.setPlaceholderText("Введите количество фигур, которые нужно поставить")  
 self.line\_edit\_2 = QLineEdit()  
 self.line\_edit\_2.setValidator(self.validator)  
 self.line\_edit\_2.setPlaceholderText("Введите количество уже поставленных фигур")  
  
 self.button = QPushButton("koordinats")  
 self.button\_1 = QPushButton("field")  
 self.button\_2 = QPushButton("exit")  
  
 self.button.clicked.connect(self.good\_bad\_koordinats)  
 self.button\_1.clicked.connect(self.solutions)  
 self.button\_1.clicked.connect(self.table\_koord)  
 self.button.clicked.connect(self.on\_button\_clicked)  
 self.button\_1.clicked.connect(self.on\_button\_clicked\_1)  
 self.button\_2.clicked.connect(self.on\_button\_clicked\_2)  
  
 layout = QGridLayout()  
 layout.addWidget(self.label, 0, 0)  
 layout.addWidget(self.line\_edit, 0, 1)  
 layout.addWidget(self.button, 0, 2)  
  
 layout.addWidget(self.label\_1, 1, 0)  
 layout.addWidget(self.line\_edit\_1, 1, 1)  
 layout.addWidget(self.button\_1, 1, 2)  
  
 layout.addWidget(self.label\_2, 2, 0)  
 layout.addWidget(self.line\_edit\_2, 2, 1)  
 layout.addWidget(self.button\_2, 2, 2)  
  
 central\_widget = QWidget()  
 central\_widget.setLayout(layout)  
 self.setCentralWidget(central\_widget)  
  
 def on\_button\_clicked(self):  
 *''' Функция для кнопки для запуска окна ввода координат'''* with open('output.txt', 'r') as file\_6:  
 f6 = file\_6.readlines()  
 if (len(self.line\_edit.text()) > 0) and (len(self.line\_edit\_1.text()) > 0) and (len(self.line\_edit\_2.text()) > 0) and (int(f6[0][-2]) != 0):  
 new\_window = AnotherWindow()  
 new\_window.show()  
 self.open\_window.append(new\_window)  
 else:  
 None  
 def on\_button\_clicked\_1(self):  
 *''' Функция для кнопки вывода доски'''* with open('input.txt', 'r') as file\_6:  
 f6 = file\_6.readlines()  
 if (len(self.line\_edit.text()) < 0) and (len(self.line\_edit\_1.text()) < 0) and (len(self.line\_edit\_2.text()) < 0) and (len(f6) > 1):  
 new\_window = FieldWindow()  
 new\_window.show()  
 self.open\_window.append(new\_window)  
 if (len(self.line\_edit\_1.text()) > 0) and (len(f6) > 1):  
 new\_window = FieldWindow()  
 new\_window.show()  
 self.open\_window.append(new\_window)  
 if f6[0][0] != ' ':  
 if (len(f6) == 1) and (int(f6[0][-2]) == 0):  
 new\_window = FieldWindow()  
 new\_window.show()  
 self.open\_window.append(new\_window)  
 else:  
 None  
  
 def on\_button\_clicked\_2(self):  
 *''' Функция для кнопки закрытия окна'''* MainWindow.deleteLater(self)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
  
 window = MainWindow()  
 window.show()  
  
 sys.exit(app.exec())

# ЗАКЛЮЧЕНИЕ

В ходе работы мы написали программу, которая приводит в работу приложение, основанное на коде с шахматами из прошлого семестра. Кроме того, в ходе работы была изучена библиотека PySide6, которая помогла написать программу GUI приложения для шахмат.

# Список использованных источников

# <https://www.pythonguis.com/tutorials/pyside6-layouts/>

1. <https://fadeevlecturer.github.io/python_lectures/notebooks/qt/basics.html>
2. https://www.pythoncentral.io/pyside6-how-to-build-gui/