ФЕДЕРАЛЬНОЕ АГЕНСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ Императора Александра I»

Кафедра «Информационные и вычислительные системы» Дисциплина «Программирование на языках высокого уровня (Python)»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8

Выполнил студент Шефнер А.

Факультет: АИТ Группа: ИВБ-211

Проверил: Баталов Д.И.

Санкт-Петербург 2023

Оценочный лис	г результатов	ЛР	No	8

Ф.И.О. студента	Шефнер Альберт	
Группа	ИВБ-211	

№ п/ п	Материалы необходимые для оценки знаний, умений и навыков	Показатель оценивания	Критерии Оценивания	Шкала оценивания	Оценка
1		Соответствие	Соответствует	7	
		методике	He	0	
		выполнения	соответствует		
		Срок	Выполнена в	2	
		выполнения	срок		
	Лабораторная		Выполнена с	0	
	работа№		опозданием на 2		
			недели		
		оформление	Соответствует	1	
			требованиям	0	
			He		
			соответствует		
	итого			10	
	количество баллов			10	

Доцент кафедры		
«Информационные и вычислительные		
системы»	Баталов Д.И.	«»
2023 г.		

10. Напишите программу, позволяющую заходить в ее основное окно только авторизированным пользователям, с возможностью регистрации нового пользователя. Хеш связки «логин:пароль» пользователей хранится в отдельном файле.

Код Импорты

```
import sys
import json
import os
import hashlib
from PySide6.QtCore import Qt, Slot
from PySide6.QtGui import QFont
from PySide6.QtWidgets import (
    QHBoxLayout,
    QMessageBox,
    QPushButton,
    QVBoxLayout,
    QLabel,
    QLineEdit,
    QMainWindow,
    QWidget,
    QApplication
```

Константа с относительным путём к файлу с паролями

```
25
26 PASS_FILE = "passwords.json"
27
```

Функции для работы с логином и паролем

```
def pass_hash(login: str, pass_: str) → str:
   return hashlib.sha256((pass_ + login).encode()).hexdigest()
def create_pass_file_if_dont_exists():
   if os.path.exists(PASS_FILE):
        return
   with open(PASS_FILE, 'w') as file:
       file.write("{}")
def add_pass(login: str, pass_: str) → None:
    create_pass_file_if_dont_exists()
   with open(PASS_FILE, 'r') as file:
        passwords = json.load(file)
   passwords[login] = pass_hash(login, pass_)
   with open(PASS_FILE, 'w') as file:
       json.dump(passwords, file)
def check_pass(login: str, pass_: str) → bool:
   hash_ = pass_hash(login, pass_)
   with open(PASS_FILE, 'r') as file:
        passwords = json.load(file)
   return login in passwords and passwords[login] = hash_
def check_login(login: str):
   with open(PASS_FILE, 'r') as file:
        passwords = json.load(file)
   return login in passwords
```

Класс виджета авторизации

```
65 class AuthWidget(QWidget):
66 def __init__(self) → None:
67 super().__init__()
```

```
primary_font = QFont("Times", 32, QFont.Bold)
             secondary_font = QFont("Times", 20)
             self.auth_label = QLabel("Авторизация")
             self.auth_label.setFont(primary_font)
             self.login_label = QLabel("Логин:")
             self.login_label.setFont(secondary_font)
             self.login_edit = QLineEdit()
             self.pass_label = QLabel("Пароль:")
             self.pass_label.setFont(secondary_font)
             self.pass_edit = QLineEdit()
             self.enter_button = QPushButton("Войти")
             self.enter_button.setFont(secondary_font)
             self.reg_button = QPushButton("Зарегистрироваться")
             self.reg_button.setFont(secondary_font)
89
             buttons_layout = QHBoxLayout()
             buttons_layout.addWidget(self.enter_button)
             buttons_layout.addWidget(self.reg_button)
93
             vbox = QVBoxLayout()
             vbox.addWidget(self.auth_label)
             vbox.addStretch(1)
             vbox.addWidget(self.login_label)
             vbox.addWidget(self.login_edit)
             vbox.addStretch(1)
             vbox.addWidget(self.pass_label)
             vbox.addWidget(self.pass_edit)
             vbox.addStretch(1)
             vbox.addLayout(buttons_layout)
             vbox.addSpacing(30)
             vbox.setAlignment(Qt.AlignJustify)
             hbox = QHBoxLayout()
             hbox.addSpacing(30)
             hbox.addLayout(vbox)
             hbox.addSpacing(30)
             self.setLayout(hbox)
```

Класс виджета регистрации

```
113
      class RegWidget(QWidget):
          def \_init\_(self) \rightarrow None:
              super().__init__()
              primary_font = QFont("Times", 32, QFont.Bold)
              secondary_font = QFont("Times", 20)
              self.reg_label = QLabel("Регистрация")
              self.reg_label.setFont(primary_font)
              self.back_button = QPushButton("Назад")
              self.back_button.setFont(secondary_font)
              self.login_label = QLabel("Логин:")
              self.login_label.setFont(secondary_font)
              self.login_edit = QLineEdit()
              self.pass_label = QLabel("Пароль:")
              self.pass_label.setFont(secondary_font)
              self.pass_edit = QLineEdit()
              self.confirm_label = QLabel("Подтвердите пароль:")
              self.confirm_label.setFont(secondary_font)
              self.confirm_edit = QLineEdit()
              self.reg_button = QPushButton("Зарегистрироваться")
              self.reg_button.setFont(secondary_font)
              top_hbox = OHBoxLayout()
              top_hbox.addWidget(self.back_button)
              top_hbox.addWidget(self.reg_label)
```

```
vbox = QVBoxLayout()
vbox.addLayout(top_hbox)
vbox.addStretch(1)
vbox.addWidget(self.login_label)
vbox.addWidget(self.login_edit)
vbox.addStretch(1)
vbox.addWidget(self.pass_label)
vbox.addWidget(self.pass_edit)
vbox.addStretch(1)
vbox.addWidget(self.confirm_label)
vbox.addWidget(self.confirm_edit)
vbox.addStretch(1)
vbox.addWidget(self.reg_button)
vbox.addSpacing(30)
vbox.setAlignment(Qt.AlignJustify)
hbox = QHBoxLayout()
hbox.addSpacing(30)
hbox.addLayout(vbox)
hbox.addSpacing(30)
self.setLayout(hbox)
```

Класс виджета Казахстан

```
class KazakhstanWidget(QWidget):
    def \_init\_(self) \rightarrow None:
        super().__init__()
        primary_font = QFont("Times", 32, QFont.Bold)
        secondary_font = QFont("Times", 20)
        self.Kazakhstan_label = QLabel("Добро пожаловать в Казахстан!")
        self.hymn_label = QLabel("""
Жаралған намыстан қаһарман халықпыз,
Азаттық жолында жалындап жаныппыз.
Тағдырдың тезінен, тозақтың өзінен
Аман-сау қалыппыз, аман-сау қалыппыз.
Кайырмасы:
Еркіндік қыраны шарықта,
Елдікке шақырып тірлікте!
Алыптың қуаты – халықта,
Халықтың қуаты - бірлікте!
```

```
Ардақтап анасын, құрметтеп данасын,
Бауырға басқанбыз баршаның баласын.
Татулық, достықтың киелі бесігі
Мейірбан Ұлы Отан, қазақтың даласы!
Кайырмасы
Талайды өткердік, өткенге салауат,
Тәуліктік сәулетті, келешек ғаламат!
Ар-ождан, ана тіл, өнеге-салтымыз,
Ерлік те, елдік те ұрпаққа аманат!
Кайырмасы[3]
        self.Kazakhstan_label.setFont(primary_font)
        self.hymn_label.setFont(secondary_font)
        self.hymn_label.setAlignment(Qt.AlignHCenter)
        self.Kazakhstan_label.setAlignment(Qt.AlignHCenter)
        vbox = QVBoxLayout()
        vbox.addWidget(self.Kazakhstan_label)
        vbox.addWidget(self.hymn_label)
        self.setLayout(vbox)
```

Класс главного окна

```
@Slot()
def switch_to_reg(self):
   reg_widget = RegWidget()
    reg_widget.reg_button.clicked.connect(self.reg)
    reg_widget.back_button.clicked.connect(self.switch_to_auth)
    self.setCentralWidget(reg_widget)
@Slot()
def auth(self):
   login = self.centralWidget().login_edit.text()
    password = self.centralWidget().pass_edit.text()
    if(check_pass(login, password)):
        self.switch_to_Kazakhstan()
    else:
        QMessageBox().warning(
            "Неправильный логин или пароль! Попробуйте снова...",
            QMessageBox.Ok,
            QMessageBox.Ok
@Slot()
def reg(self):
    login = self.centralWidget().login_edit.text()
    password = self.centralWidget().pass_edit.text()
    confirm = self.centralWidget().confirm_edit.text()
    if login = '':
        self.warning("Логин не должен быть пустым.")
        return
    if password = '':
        self.warning("Пароль не должен быть пустым")
        return
    if password \neq confirm:
        self.warning("Пароли не совпадают!\nПовторите ввод.")
        return
    if check_login(login):
        QMessageBox().warning(
            self, "",
            "Данный логин уже существует.\пПопробуйте другой",
            QMessageBox.Ok,
            QMessageBox.Ok
        return
```

```
add_pass(login, password)

QMessageBox().information(

self, "",

"Вы успешно зарегистрированы!\пИспользуйте ваш новый логин для входа.",

QMessageBox.Ok,

QMessageBox.Ok

)

self.switch_to_auth()

def warning(self, text: str):

QMessageBox().warning(
self, "",
text,
QMessageBox.Ok

)

QMessageBox.Ok

)

QMessageBox.Ok,
QMessageBox.Ok

)

QMessageBox.Ok

)

ger

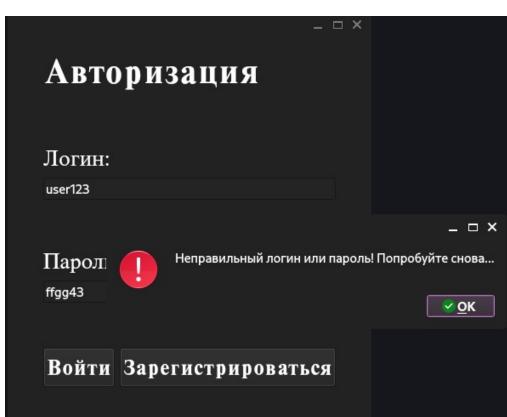
def switch_to_Kazakhstan(self):
self.setCentralWidget(KazakhstanWidget())
```

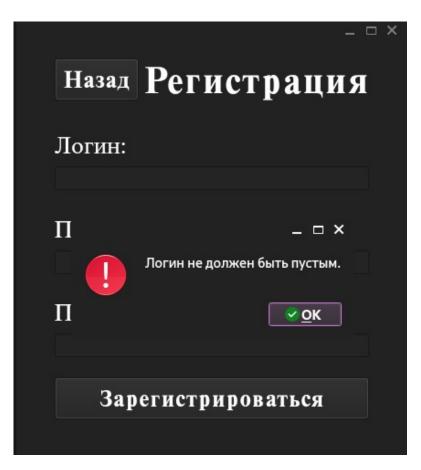
Точка входа программы

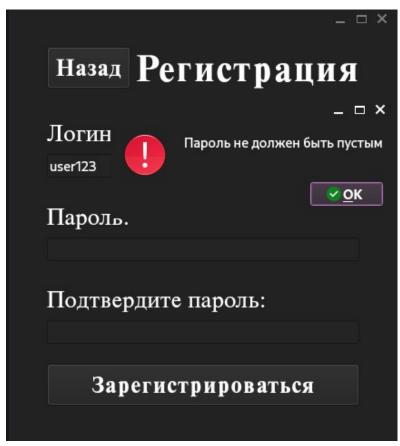
```
if __name__ = "__main__":
    app = QApplication()
    window = MainWindow()
    window.show()
    sys.exit(app.exec())
```

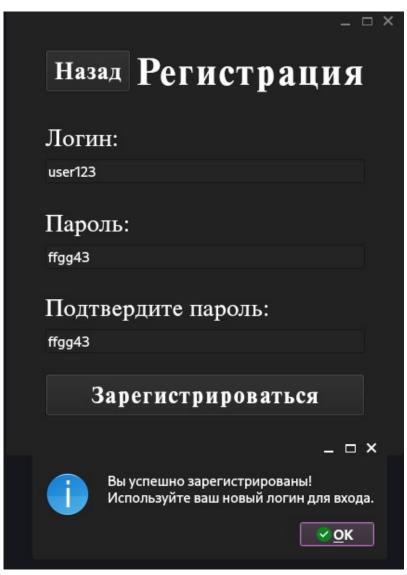
Отладка

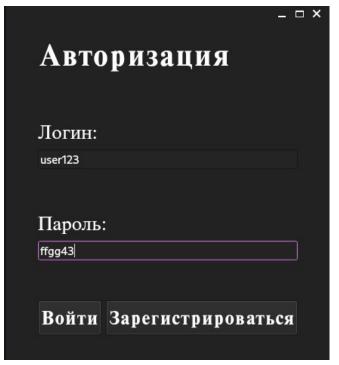












Добро пожаловать в Казахстан!

Жаралған намыстан қаһарман халықпыз, Азаттық жолында жалындап жаныппыз. Тағдырдың тезінен, тозақтың өзінен Аман-сау қалыппыз, аман-сау қалыппыз.

Қайырмасы:

Еркіндік қыраны шарықта, Елдікке шақырып тірлікте! Алыптың қуаты – халықта, Халықтың қуаты – бірлікте!

Ардақтап анасын, құрметтеп данасын, Бауырға басқанбыз баршаның баласын. Татулық, достықтың киелі бесігі Мейірбан Ұлы Отан, қазақтың даласы!

Қайырмасы

Талайды өткердік, өткенге салауат, Тәуліктік сәулетті, келешек ғаламат! Ар-ождан, ана тіл, өнеге-салтымыз, Ерлік те, елдік те ұрпаққа аманат!

Қайырмасы[3]

```
auth.py × {} passwords.json ×

1 {
2          "awd": "b8120cbdf388ff2e8e7f40f0d07733d797a0cfe67440019b548d190b4498e25a",
3          "2": "785f3ec7eb32f30b90cd0fcf3657d388b5ff4297f2f9716ff66e9b69c05ddd09",
4          "234": "114bd151f8fb0c58642d2170da4ae7d7c57977260ac2cc8905306cab6b2acabc",
5          "345": "7c84ddd902607dc8d16498fa0700577ebed03c4f6527c8f885d15eb4c9ec07f1",
6          "x": "a048e640908046be06e00eab37742b5d5ff80964af58cfd22f7cb2de4dfe375f",
7          "lol": "42c3286e243c3009056c2f27f473093de251fa745ef298928a207a659f0109b6",
8          "dolefuariaq": "e50a16f5bd235e8342a24f8a0bb8551971da7cc7f372c3c3595c021865b0a42b",
9          "user123": "2a159fc82f0e28b8b64ffc7fa9a52c457d3f5cb47ae88a7cf9b6baca329e2968"
```

20. Напишите программу-игру «Змейка».

Код

Импорты

```
import enum
import random
import sys
from typing import List
from PySide6.QtWidgets import (
    QWidget,
    QPushButton,
    QMainWindow,
    QApplication,
from PySide6.QtCore import QPoint, QRect, QSize, QTimer, Qt, Slot
from PySide6.QtGui import (
    QKeyEvent,
    QPaintEvent,
    QPen,
    QPainter,
    QPixmap,
```

Перечисление направлений змейки

```
    class Direction(enum.IntEnum):
        UP = 0,
        DOWN = 1,
        LEFT = 2,
        RIGHT = 3
```

Класс точки

Класс змейки (главного героя)

```
class Snake:
    segments: List[Point]
    direction: Direction
    def __init__(self, x: int, y: int, direction: Direction) → None:
        self.direction = direction
        self.segments = [Point(x, y)]
    Oproperty
    def head(self):
       return self.segments[0]
    def move(self):
        self.grow()
        self.segments.pop()
    def grow(self):
        new_head = Point(self.head.x, self.head.y)
        match self.direction:
            case Direction.UP:
               new_head.y -= 1
            case Direction.DOWN:
               new_head.y += 1
            case Direction.LEFT:
               new_head.x -= 1
            case Direction.RIGHT:
                new_head.x += 1
        self.segments.insert(0, new_head)
```

Класс игры, в котором хранится текущее состояние игры и происходит основная логика

```
class Game:
    snake: Snake
    cells_x: int
    cells_y: int
    apple: Point
    running: bool
    def = init_{self}, cells_x: int, cells_y: int) \rightarrow None:
        self.snake_direction = Direction.RIGHT
        self.snake = Snake(5, 5, self.snake_direction)
        for _ in range(3):
            self.snake.grow()
        self.cells_x = cells_x
        self.cells_y = cells_y
        self.apple = Point(-1, -1)
        self.running = True
        self.spawn_apple()
    def update(self) \rightarrow None:
        if not self.running:
            return
        self.snake.direction = self.snake_direction
        head = self.snake.head
        if head = self.apple:
            self.spawn_apple()
            self.snake.grow()
        else:
            self.snake.move()
        self.snake.head.x %= self.cells_x
        self.snake.head.y %= self.cells_y
        if self.snake.head in self.snake.segments[1:]:
            self.running = False
```

```
def change_direction(self, direction: Direction) \rightarrow None:
    if direction = Direction.DOWN and self.snake.direction \neq Direction.UP:
        self.snake_direction = direction
    if direction = Direction.UP and self.snake.direction \neq Direction.DOWN:
        self.snake_direction = direction
    if direction = Direction.LEFT and self.snake.direction \neq Direction.RIGHT:
        self.snake_direction = direction
    if direction = Direction.RIGHT and self.snake.direction \neq Direction.LEFT:
        self.snake_direction = direction
def spawn_apple(self):
    while True:
        self.apple = Point(
            random.randint(0, self.cells_x - 1),
            random.randint(0, self.cells_y - 1)
        if self.apple not in self.snake.segments:
            break
```

Класс-виджет, отвечающий за отрисовку игры

```
class SnakePaint(QWidget):
    def __init__(self, width: int, height: int, segment_width: int) \rightarrow None:
        super().__init__()
        self.cell_width: int = segment_width
        self.cell_size = QSize(self.cell_width, self.cell_width)
        self.setFixedSize(width, height)
        self.pixmap = QPixmap(self.size())
        self.pixmap.fill(Qt.white)
        self.pen = QPen()
        self.pen.setWidth(16)
    def paintEvent(self, _: QPaintEvent) \rightarrow None:
        with QPainter(self) as painter:
            painter.drawPixmap(0, 0, self.pixmap)
    def draw_game(self, game: Game):
        def crd(coord):
            return self.cell_width * coord
        self.pixmap.fill(Qt.white)
        with QPainter(self.pixmap) αs painter:
            apple_rect = QRect(
                QPoint(crd(game.apple.x), crd(game.apple.y)),
                self.cell_size
            painter.fillRect(apple_rect, Qt.red)
```

Класс главного окна, в котором происходит управление игрой и обработка событий

```
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.cells_x = 25
        self.cells_y = 20
        self.cell_width = 50
        width = self.cells_x * self.cell_width
        height = self.cells_y * self.cell_width
        self.setGeometry(0, 0, width, height)
        self.setWindowTitle("Snake.py")
        self.button = QPushButton()
        self.button.setText("xd")
        self.snakepaint = SnakePaint(width, height, 50)
        self.setCentralWidget(self.snakepaint)
        self.timer = QTimer(self)
        self.timer.timeout.connect(self.update)
        self.game = Game(self.cells_x, self.cells_y)
    def run(self):
        self.timer.start(100)
```

Точка входа программы

```
v221 if __name__ = "__main__":
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    window.run()
    sys.exit(app.exec())
```

Отладка

