

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
Факультет: АИТ
Группа: ИВБ-211

Шефнер А.

Проверил:

Баталов Д.И.

Санкт-Петербург

2023

Оценочный лист результатов ЛР № 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	Лабораторная работа№	Соответствие методике выполнения	Соответствует	7	
			Не соответствует	0	
		Срок выполнения	Выполнена в срок	2	
			Выполнена с опозданием на 2 недели	0	
		оформление	Соответствует требованиям	1 0	
			Не соответствует		
	ИТОГО количество баллов			10	

Доцент кафедры

«Информационные и вычислительные
системы»

_____ 2023 г.

Баталов Д.И. «__»

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

Код

Импорты

```
1  import sys
2  import json
3  import os
4  import hashlib
5  from PySide6.QtCore import Qt, Slot
6  from PySide6.QtGui import QFont
7  from PySide6.QtWidgets import (
8      QHBoxLayout,
9      QMessageBox,
10     QPushButton,
11     QVBoxLayout,
12     QLabel,
13     QLineEdit,
14     QMainWindow,
15     QWidget,
16     QApplication
17 )
```

Константа с относительным путём к файлу с паролями

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

Функции для работы с логином и паролем

```
29 def pass_hash(login: str, pass_: str) → str:
30     return hashlib.sha256((pass_ + login).encode()).hexdigest()
31
32 def create_pass_file_if_dont_exists():
33     if os.path.exists(PASS_FILE):
34         return
35
36     with open(PASS_FILE, 'w') as file:
37         file.write("{}")
38
39 def add_pass(login: str, pass_: str) → None:
40     create_pass_file_if_dont_exists()
41
42     with open(PASS_FILE, 'r') as file:
43         passwords = json.load(file)
44
45     passwords[login] = pass_hash(login, pass_)
46
47     with open(PASS_FILE, 'w') as file:
48         json.dump(passwords, file)
49
50 def check_pass(login: str, pass_: str) → bool:
51     hash_ = pass_hash(login, pass_)
52
53     with open(PASS_FILE, 'r') as file:
54         passwords = json.load(file)
55
56     return login in passwords and passwords[login] == hash_
57
58 def check_login(login: str):
59     with open(PASS_FILE, 'r') as file:
60         passwords = json.load(file)
61
62     return login in passwords
```

Класс виджета авторизации

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

```
69     primary_font = QFont("Times", 32, QFont.Bold)
70     secondary_font = QFont("Times", 20)
71
72     self.auth_label = QLabel("Авторизация")
73     self.auth_label.setFont(primary_font)
74
75     self.login_label = QLabel("Логин:")
76     self.login_label.setFont(secondary_font)
77     self.login_edit = QLineEdit()
78
79     self.pass_label = QLabel("Пароль:")
80     self.pass_label.setFont(secondary_font)
81     self.pass_edit = QLineEdit()
82
83     self.enter_button = QPushButton("Войти")
84     self.enter_button.setFont(secondary_font)
85
86     self.reg_button = QPushButton("Зарегистрироваться")
87     self.reg_button.setFont(secondary_font)
88
89     buttons_layout = QHBoxLayout()
90     buttons_layout.addWidget(self.enter_button)
91     buttons_layout.addWidget(self.reg_button)
92
93     vbox = QVBoxLayout()
94     vbox.addWidget(self.auth_label)
95     vbox.addStretch(1)
96     vbox.addWidget(self.login_label)
97     vbox.addWidget(self.login_edit)
98     vbox.addStretch(1)
99     vbox.addWidget(self.pass_label)
100    vbox.addWidget(self.pass_edit)
101    vbox.addStretch(1)
102    vbox.addLayout(buttons_layout)
103    vbox.addSpacing(30)
104    vbox.setAlignment(Qt.AlignJustify)
105
106    hbox = QHBoxLayout()
107    hbox.addSpacing(30)
108    hbox.addLayout(vbox)
109    hbox.addSpacing(30)
110    self.setLayout(hbox)
```


Класс виджета регистрации

```
113 class RegWidget(QWidget):
114     def __init__(self) → None:
115         super().__init__()
116
117         primary_font = QFont("Times", 32, QFont.Bold)
118         secondary_font = QFont("Times", 20)
119
120         self.reg_label = QLabel("Регистрация")
121         self.reg_label.setFont(primary_font)
122
123         self.back_button = QPushButton("Назад")
124         self.back_button.setFont(secondary_font)
125
126         self.login_label = QLabel("Логин:")
127         self.login_label.setFont(secondary_font)
128         self.login_edit = QLineEdit()
129
130         self.pass_label = QLabel("Пароль:")
131         self.pass_label.setFont(secondary_font)
132         self.pass_edit = QLineEdit()
133
134         self.confirm_label = QLabel("Подтвердите пароль:")
135         self.confirm_label.setFont(secondary_font)
136         self.confirm_edit = QLineEdit()
137
138
139         self.reg_button = QPushButton("Зарегистрироваться")
140         self.reg_button.setFont(secondary_font)
141
142         top_hbox = QHBoxLayout()
143         top_hbox.addWidget(self.back_button)
144         top_hbox.addWidget(self.reg_label)
```

```

146 | vbox = QVBoxLayout()
147 | vbox.addLayout(top_hbox)
148 | vbox.addStretch(1)
149 | vbox.addWidget(self.login_label)
150 | vbox.addWidget(self.login_edit)
151 | vbox.addStretch(1)
152 | vbox.addWidget(self.pass_label)
153 | vbox.addWidget(self.pass_edit)
154 | vbox.addStretch(1)
155 | vbox.addWidget(self.confirm_label)
156 | vbox.addWidget(self.confirm_edit)
157 | vbox.addStretch(1)
158 | vbox.addWidget(self.reg_button)
159 | vbox.addSpacing(30)
160 | vbox.setAlignment(Qt.AlignJustify)
161 |
162 | hbox = QHBoxLayout()
163 | hbox.addSpacing(30)
164 | hbox.addLayout(vbox)
165 | hbox.addSpacing(30)
166 | self.setLayout(hbox)

```

Класс виджета Казахстан

```

169 class KazakhstanWidget(QWidget):
170     def __init__(self) → None:
171         super().__init__()
172
173         primary_font = QFont("Times", 32, QFont.Bold)
174         secondary_font = QFont("Times", 20)
175
176         self.Kazakhstan_label = QLabel("Добро пожаловать в Казахстан!")
177         self.hymn_label = QLabel("""
178     Жаралған намыстан қаһарман халықпыз,
179     Азаттық жолында жалындап жаныппыз.
180     Тағдырдың тезінен, тозақтың өзінен
181     Аман-сау қалыппыз, аман-сау қалыппыз.
182
183     Қайырмасы:
184     Еркіндік қыраны шарықта,
185     Елдікке шақырып тірлікте!
186     Алыптың қуаты – халықта,
187     Халықтың қуаты – бірлікте!

```



```

189 Ардақтап анасын, құрметтеп данасын,
190 Бауырға басқанбыз баршаның баласын.
191 Татулық, достықтың киелі бесігі
192 Мейірбан Ұлы Отан, қазақтың даласы!
193
194 Қайырмасы
195
196 Талайды өткердік, өткенге салауат,
197 Тәуліктік сәулетті, келешек ғаламат!
198 Ар-ождан, ана тіл, өнеге-салтымыз,
199 Ерлік те, елдік те ұрпаққа аманат!
200
201 Қайырмасы[3]
202     """
203
204     self.Kazakhstan_label.setFont(primary_font)
205     self.hymn_label.setFont(secondary_font)
206
207     self.hymn_label.setAlignment(Qt.AlignHCenter)
208     self.Kazakhstan_label.setAlignment(Qt.AlignHCenter)
209
210     vbox = QVBoxLayout()
211     vbox.addWidget(self.Kazakhstan_label)
212     vbox.addWidget(self.hymn_label)
213     self.setLayout(vbox)

```

Класс главного окна

```

~ 216 class MainWindow(QMainWindow):
~ 217     def __init__(self) → None:
218         super().__init__()
219
220         self.setGeometry(0, 0, 300, 450)
221         self.switch_to_auth()
222
223     @Slot()
~ 224     def switch_to_auth(self):
225         auth_widget = AuthWidget()
226         auth_widget.reg_button.clicked.connect(self.switch_to_reg)
227         auth_widget.enter_button.clicked.connect(self.auth)
228
229         self.setCentralWidget(auth_widget)

```



```
231 @Slot()
232 def switch_to_reg(self):
233     reg_widget = RegWidget()
234     reg_widget.reg_button.clicked.connect(self.reg)
235     reg_widget.back_button.clicked.connect(self.switch_to_auth)
236     self.setCentralWidget(reg_widget)
237
238 @Slot()
239 def auth(self):
240     login = self.centralWidget().login_edit.text()
241     password = self.centralWidget().pass_edit.text()
242     if(check_pass(login, password)):
243         self.switch_to_Kazakhstan()
244     else:
245         QMessageBox().warning(
246             self, "",
247             "Неправильный логин или пароль! Попробуйте снова...",
248             QMessageBox.Ok,
249             QMessageBox.Ok
250         )
251
252 @Slot()
253 def reg(self):
254
255     login = self.centralWidget().login_edit.text()
256     password = self.centralWidget().pass_edit.text()
257     confirm = self.centralWidget().confirm_edit.text()
258
259     if login == '':
260         self.warning("Логин не должен быть пустым.")
261         return
262
263     if password == '':
264         self.warning("Пароль не должен быть пустым")
265         return
266
267
268     if password != confirm:
269         self.warning("Пароли не совпадают!\nПовторите ввод.")
270         return
271
272     if check_login(login):
273         QMessageBox().warning(
274             self, "",
275             "Данный логин уже существует.\nПопробуйте другой",
276             QMessageBox.Ok,
277             QMessageBox.Ok
278         )
279         return
```

```
281         add_pass(login, password)
282     QMessageBox().information(
283         self, "",
284         "Вы успешно зарегистрированы!\nИспользуйте ваш новый логин для входа.",
285         QMessageBox.Ok,
286         QMessageBox.Ok
287     )
288     self.switch_to_auth()
289
290     def warning(self, text: str):
291     QMessageBox().warning(
292         self, "",
293         text,
294         QMessageBox.Ok,
295         QMessageBox.Ok
296     )
297
298     @Slot()
299     def switch_to_Kazakhstan(self):
300         self.setCentralWidget(KazakhstanWidget())
```

Точка входа программы

```
303     if __name__ == "__main__":
304         app = QApplication()
305         window = MainWindow()
306         window.show()
307         sys.exit(app.exec())
```

Отладка

Авторизация

Логин:

Пароль:

Войти

Зарегистрироваться

Авторизация

Логин:

user123

Пароль:

ffgg43

!

Неправильный логин или пароль! Попробуйте снова...

Войти

Зарегистрироваться

✓

OK

Назад

Регистрация

Логин:

П

!

Логин не должен быть пустым.

П

✓

OK

Зарегистрироваться

Назад

Регистрация

Логин

user123

!

Пароль не должен быть пустым

✓

OK

Пароль.

Подтвердите пароль:

Зарегистрироваться

Назад

Регистрация

Логин:

user123


Пароль:

ffgg43

Подтвердите пароль:

ffgg43

Зарегистрироваться



Вы успешно зарегистрированы!

Используйте ваш новый логин для входа.

✓

OK

Авторизация

Логин:

user123

Пароль:

ffgg43

Войти

Зарегистрироваться

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 x {} passwords.json x
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"
10 }
```

20. Напишите программу-игру «Змейка».

Код

Импорты

```
5 import enum
6 import random
7 import sys
8 from typing import List
9 from PySide6.QtWidgets import (
10     QWidget,
11     QPushButton,
12     QMainWindow,
13     QApplication,
14 )
15 from PySide6.QtCore import QPoint, QRect, QSize, QTimer, Qt, Slot
16 from PySide6.QtGui import (
17     QKeyEvent,
18     QPaintEvent,
19     QPen,
20     QPainter,
21     QPixmap,
22 )
```

Перечисление направлений змейки

```
25 class Direction(enum.IntEnum):
26     UP = 0,
27     DOWN = 1,
28     LEFT = 2,
29     RIGHT = 3
```

Класс точки

```
32 class Point:
33     x: int
34     y: int
35
36     def __init__(self, x: int, y: int) → None:
37         self.x = x
38         self.y = y
39
40     def __eq__(self, other: "Point") → bool:
41         return self.x == other.x and self.y == other.y
42
```

Класс змейки (главного героя)

```
44 class Snake:
45     segments: List[Point]
46     direction: Direction
47
48     def __init__(self, x: int, y: int, direction: Direction) → None:
49         self.direction = direction
50         self.segments = [Point(x, y)]
51
52     @property
53     def head(self):
54         return self.segments[0]
55
56     def move(self):
57         self.grow()
58         self.segments.pop()
59
60     def grow(self):
61         new_head = Point(self.head.x, self.head.y)
62         match self.direction:
63             case Direction.UP:
64                 new_head.y -= 1
65             case Direction.DOWN:
66                 new_head.y += 1
67             case Direction.LEFT:
68                 new_head.x -= 1
69             case Direction.RIGHT:
70                 new_head.x += 1
71         self.segments.insert(0, new_head)
```

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

```
74 class Game:
75     snake: Snake
76     cells_x: int
77     cells_y: int
78     apple: Point
79     running: bool
80
81     def __init__(self, cells_x: int, cells_y: int) → None:
82         self.snake_direction = Direction.RIGHT
83
84         self.snake = Snake(5, 5, self.snake_direction)
85         for _ in range(3):
86             self.snake.grow()
87
88         self.cells_x = cells_x
89         self.cells_y = cells_y
90         self.apple = Point(-1, -1)
91         self.running = True
92         self.spawn_apple()
93
94     def update(self) → None:
95         if not self.running:
96             return
97         self.snake.direction = self.snake_direction
98         head = self.snake.head
99         if head == self.apple:
100             self.spawn_apple()
101             self.snake.grow()
102         else:
103             self.snake.move()
104
105         self.snake.head.x %= self.cells_x
106         self.snake.head.y %= self.cells_y
107
108         if self.snake.head in self.snake.segments[1:]:
109             self.running = False
```



```

111 def change_direction(self, direction: Direction) → None:
112     if direction == Direction.DOWN and self.snake.direction != Direction.UP:
113         self.snake.direction = direction
114     if direction == Direction.UP and self.snake.direction != Direction.DOWN:
115         self.snake.direction = direction
116     if direction == Direction.LEFT and self.snake.direction != Direction.RIGHT:
117         self.snake.direction = direction
118     if direction == Direction.RIGHT and self.snake.direction != Direction.LEFT:
119         self.snake.direction = direction
120
121 def spawn_apple(self):
122     while True:
123         self.apple = Point(
124             random.randint(0, self.cells_x - 1),
125             random.randint(0, self.cells_y - 1)
126         )
127         if self.apple not in self.snake.segments:
128             break

```

Класс-виджет, отвечающий за отрисовку игры

```

131 class SnakePaint(QWidget):
132     def __init__(self, width: int, height: int, segment_width: int) → None:
133         super().__init__()
134         self.cell_width: int = segment_width
135         self.cell_size = QSize(self.cell_width, self.cell_width)
136         self.setFixedSize(width, height)
137
138         self.pixmap = QPixmap(self.size())
139         self.pixmap.fill(Qt.white)
140
141         self.pen = QPen()
142         self.pen.setWidth(16)
143
144     def paintEvent(self, _: QPaintEvent) → None:
145         with QPainter(self) as painter:
146             painter.drawPixmap(0, 0, self.pixmap)
147
148     def draw_game(self, game: Game):
149         def crd(coord):
150             return self.cell_width * coord
151
152         self.pixmap.fill(Qt.white)
153
154         with QPainter(self.pixmap) as painter:
155             apple_rect = QRect(
156                 QPoint(crd(game.apple.x), crd(game.apple.y)),
157                 self.cell_size
158             )
159             painter.fillRect(apple_rect, Qt.red)
160

```

```

161         for segment in game.snake.segments[1:]:
162             rect = QRect(
163                 QPoint(crd(segment.x), crd(segment.y)),
164                 self.cell_size
165             )
166             painter.fillRect(rect, Qt.darkGreen)
167
168         head_rect = QRect(
169             QPoint(crd(game.snake.head.x), crd(game.snake.head.y)),
170             self.cell_size
171         )
172         painter.fillRect(head_rect, Qt.black)
173
174         self.repaint()
175

```

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

```

177 class MainWindow(QMainWindow):
178     def __init__(self):
179         super().__init__()
180         self.cells_x = 25
181         self.cells_y = 20
182         self.cell_width = 50
183
184         width = self.cells_x * self.cell_width
185         height = self.cells_y * self.cell_width
186
187         self.setGeometry(0, 0, width, height)
188         self.setWindowTitle("Snake.py")
189
190         self.button = QPushButton()
191         self.button.setText("xd")
192
193         self.snakepaint = SnakePaint(width, height, 50)
194         self.setCentralWidget(self.snakepaint)
195
196         self.timer = QTimer(self)
197         self.timer.timeout.connect(self.update)
198
199         self.game = Game(self.cells_x, self.cells_y)
200
201     def run(self):
202         self.timer.start(100)
203

```

```

204     @Slot()
205     def update(self):
206         self.game.update()
207         self.snakepaint.draw_game(self.game)
208
209     def keyPressEvent(self, event: QKeyEvent) → None:
210         match event.key():
211             case Qt.Key_Up:
212                 self.game.change_direction(Direction.UP)
213             case Qt.Key_Down:
214                 self.game.change_direction(Direction.DOWN)
215             case Qt.Key_Left:
216                 self.game.change_direction(Direction.LEFT)
217             case Qt.Key_Right:
218                 self.game.change_direction(Direction.RIGHT)

```

Точка входа программы

```

220
221     if __name__ == "__main__":
222         app = QApplication(sys.argv)
223         window = MainWindow()
224         window.show()
225         window.run()
226         sys.exit(app.exec())
227

```


Отладка

