МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Отчет

Лабораторная работа №4

по дисциплине «Технология разработки программного обеспечения»

**«Разработка оконного приложения»**

|  |  |  |  |
| --- | --- | --- | --- |
| Студенты: | Арнольд Э.В.  Антонов С.С | Преподаватель: | Гаврилов А. В. |
| Группа: | АММ2-21 | Дата проверки: |  |
| Вариант: | 5 | Балл: |  |
| Дата сдачи: |  |  |  |

Новосибирск 2021

**Постановка задачи**

Разработать оконное приложение, работающее со структурой данных.

Функции:

* добавление в конец, получение, вставка и удаление по логическому номеру (индексу). Логический номер – порядковый номер хранимого элемента в соответствии с последовательностью обхода структуры данных (перебора элементов);
* для исходно упорядоченной (например, «двоичное дерево») вставка с сохранением порядка, получение и удаление по по логическому номеру (индексу);
* итератор. В Scala – формальный параметр = функция;
* сортировка

**Ход работы**

В основе данного оконного приложения лежит реализованная в третьей лабораторной работе структура данных (односвязный список) на языке программирования Scala. Также для создания оконного приложения использовали код на языке Java.

Для более удобного отображения данных был выбран элемент формы JTable (таблица). В класс PersonList были добавлены функции для работы с элементом JTable:

1. override def getRowCount: Int = getSize
2. override def getColumnCount: Int = columnCount
4. override def getValueAt(rowIndex: Int, columnIndex: Int): Object = {
5. val rows : Array[String] = getData(rowIndex)
6. rows(columnIndex)
7. }
9. override def getColumnName(columnIndex: Int): String = {
10. columnIndex match {
11. case 0 =>
12. return "id"
13. case 1 =>
14. return "Имя"
15. case 2 =>
16. return "Возраст"
17. }
18. ""
19. }

Был создан класс GUI. Созданный интерфейс показан на рисунке:

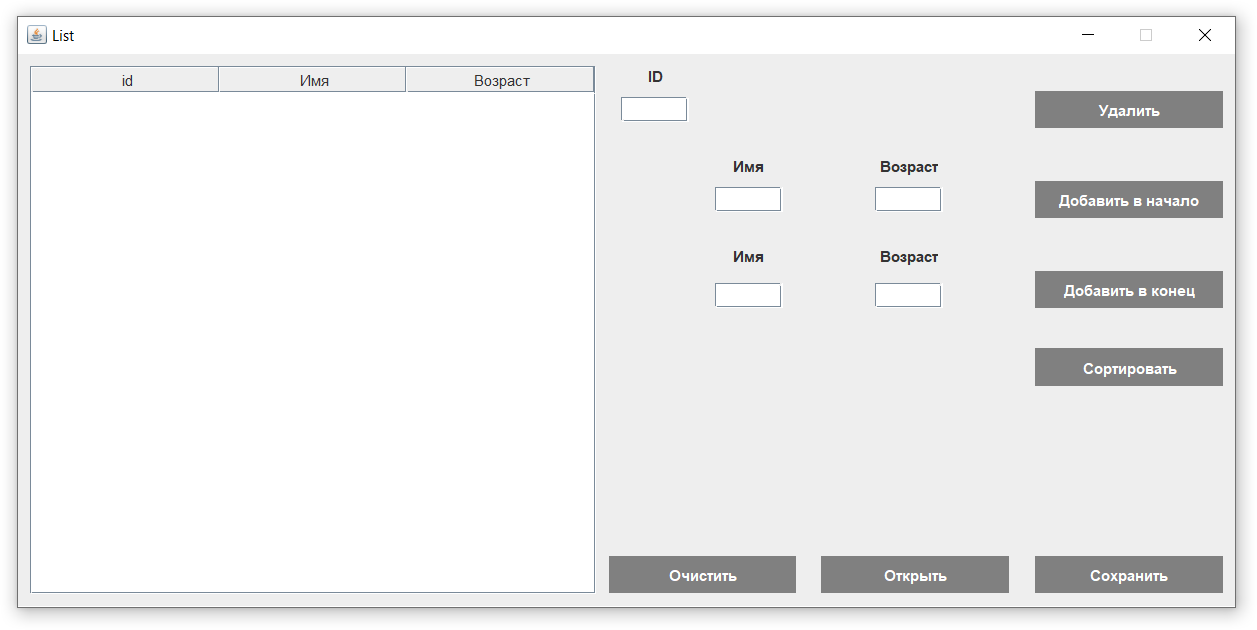


Рисунок 1 – Интерфейс оконного приложения

В левой части интерфейса располагается таблица для отображения структуры данных. В правой части располагаются текстовые поля и кнопки для взаимодействия со структурой данных.

При загрузке приложения открывается пустая форма. Можно создать новую структуру, добавляя элементы кнопками добавить в начало и конец (рисунок 2).

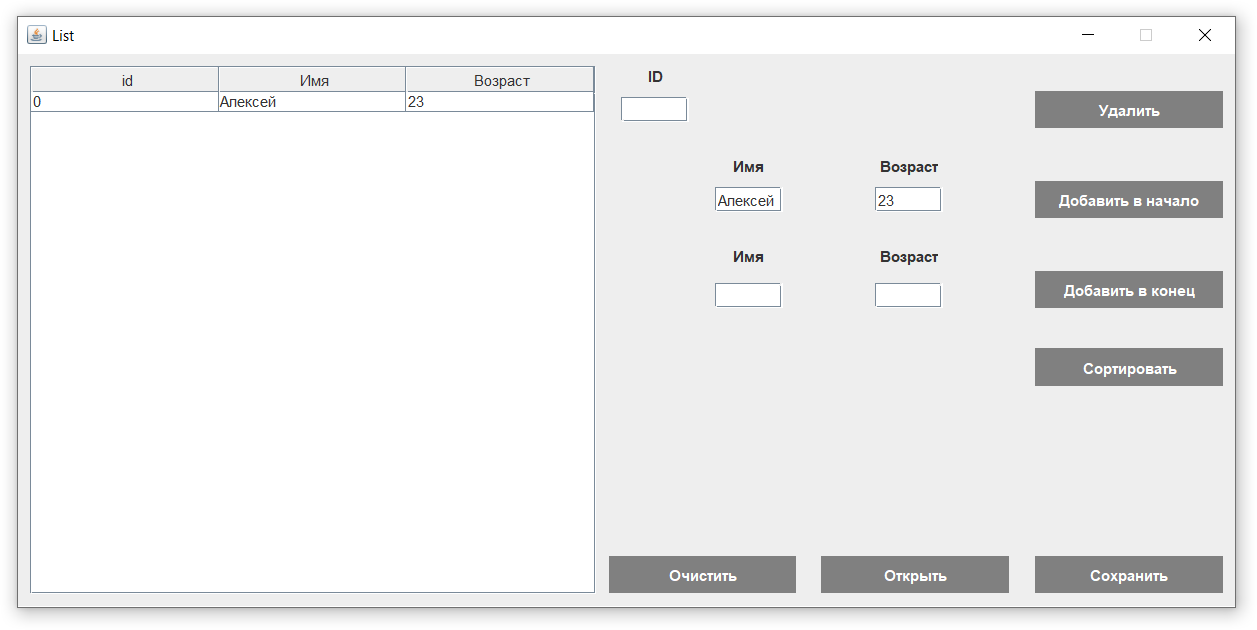


Рисунок 2 – Новая структура

Добавление элемента в структуру:

* Добавление в начало. Ввести данные и нажать кнопку «Добавить в начало».
* Добавление в конец. Ввести данные и нажать кнопку «Добавить в конец».

Результат после добавления:

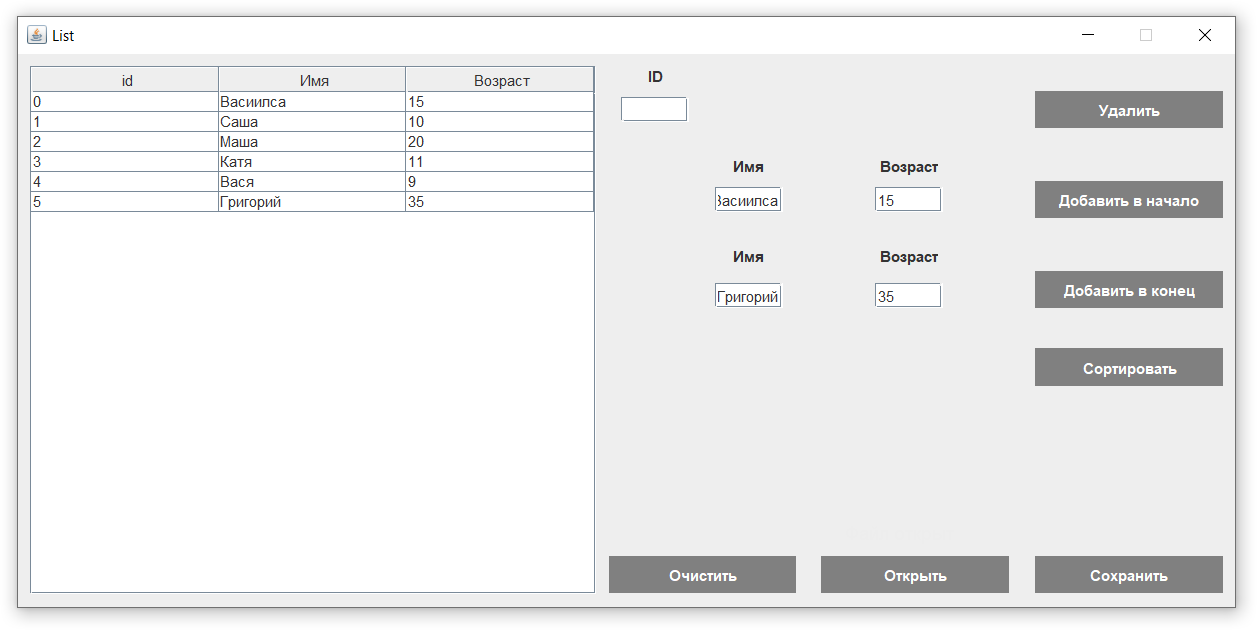


Рисунок 3 – Добавление элементов

Для добавления созданы два ActionListener, которые вызывают методы для добавления элемента, приведем код одного из них:

1. class b\_addLastActionListener extends ActionListener {
2. override def actionPerformed(e: ActionEvent): Unit = {
3. if (!(textField\_Name2.getText == "") && !(textField\_age2.getText == "")) list.addLast(textField\_Name2.getText, textField\_age2.getText.toInt)
4. list.fireTableDataChanged()
5. }
6. }

В данном случае вызывается метод класса PersonList, addLast (добавить в конец) куда передаются параметры из текстовых полей (Имя и Возраст):

1. *//добавление узла в конец списка*
2. def addLast(Name: String, age: Int): Unit = {
3. val tmp = new Person(Name, age)
4. var cur = head
5. if (cur == null) {
6. tmp.next = head
7. head = tmp
8. return
9. }
10. while ( {
11. cur.next != null
12. }) cur = cur.next
13. cur.next = tmp
14. tmp.next = null
15. }
17. *//добавление узла в начало списка*
18. def add(Name: String, age: Int): Unit = {
19. val tmp = new Person(Name, age)
20. tmp.next = head
21. head = tmp

}

Для удаления элемента из структуры нужно ввести в поле ID индекс элемента из таблицы или выбрать нужную строку в таблице и нажать кнопку удалить (рисунок 4а и 4б).

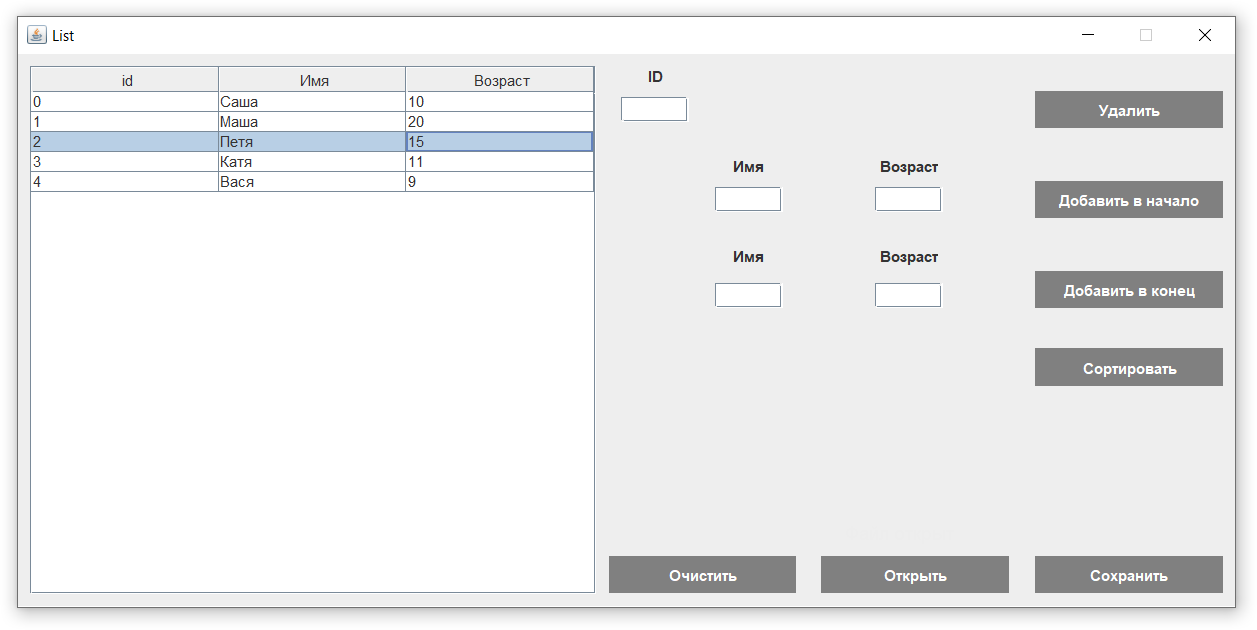


Рисунок 4а – До удаления

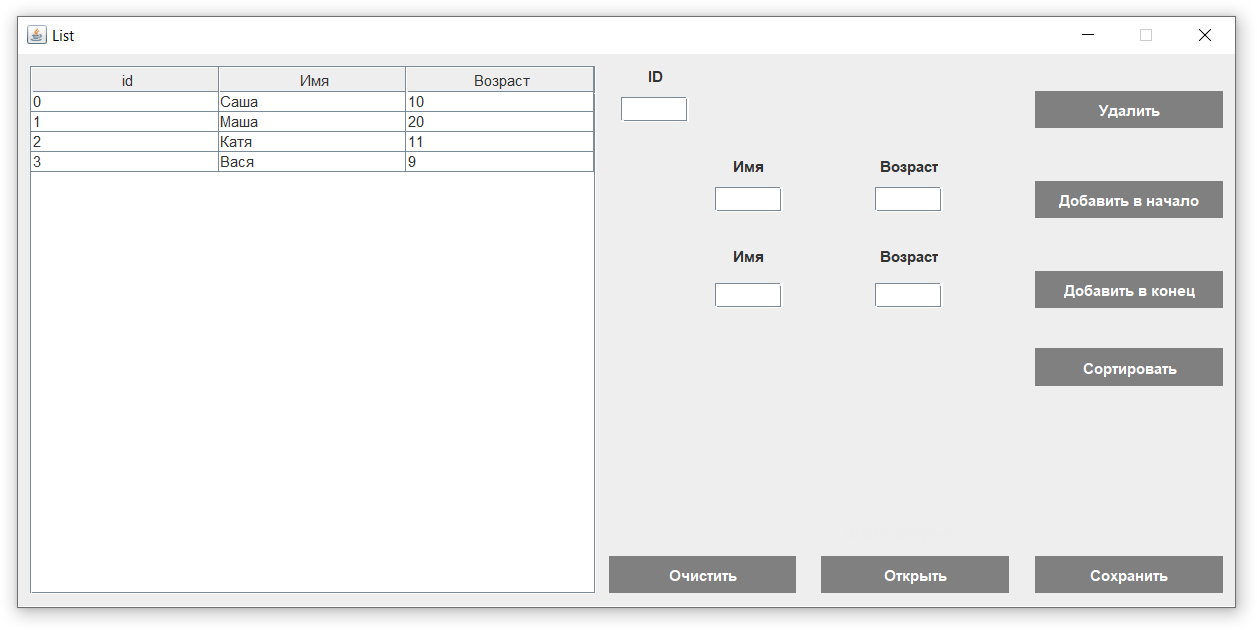


Рисунок 4б – После удаления

Удаление происходит с помощью deleteActionListener кнопки удаления:

1. class b\_deleteActionListener extends ActionListener {
2. override def actionPerformed(e: ActionEvent): Unit = {
3. if (!(textField\_delete.getText == "")) list.remove\_At(textField\_delete.getText.toInt)
4. else if (table.getSelectionModel.getLeadSelectionIndex >= 0) list.remove\_At(table.getSelectionModel.getLeadSelectionIndex)
5. list.fireTableDataChanged()
6. }
7. }

Который вызывает метод из класса списка remove\_At и передает в него номер строки для удаления:

1. def remove\_At(key: Int): Unit = {
2. var cur = head
3. var prev = head
4. if (head != null) {
5. var i = 0
6. while ( {
7. i < getSize
8. }) {
9. if (i == key) if (cur eq head) {
10. head = head.next
11. return
12. }
13. else {
14. prev.next = cur.next
15. return
16. }
17. prev = cur
18. cur = cur.next
19. {
20. i += 1;
21. i - 1
22. }
23. }
24. }
25. }

При нажатии кнопки «Сортировка» выполнится сортировка пузырьковым методом по полю «age». Результат показан на рисунке ниже:

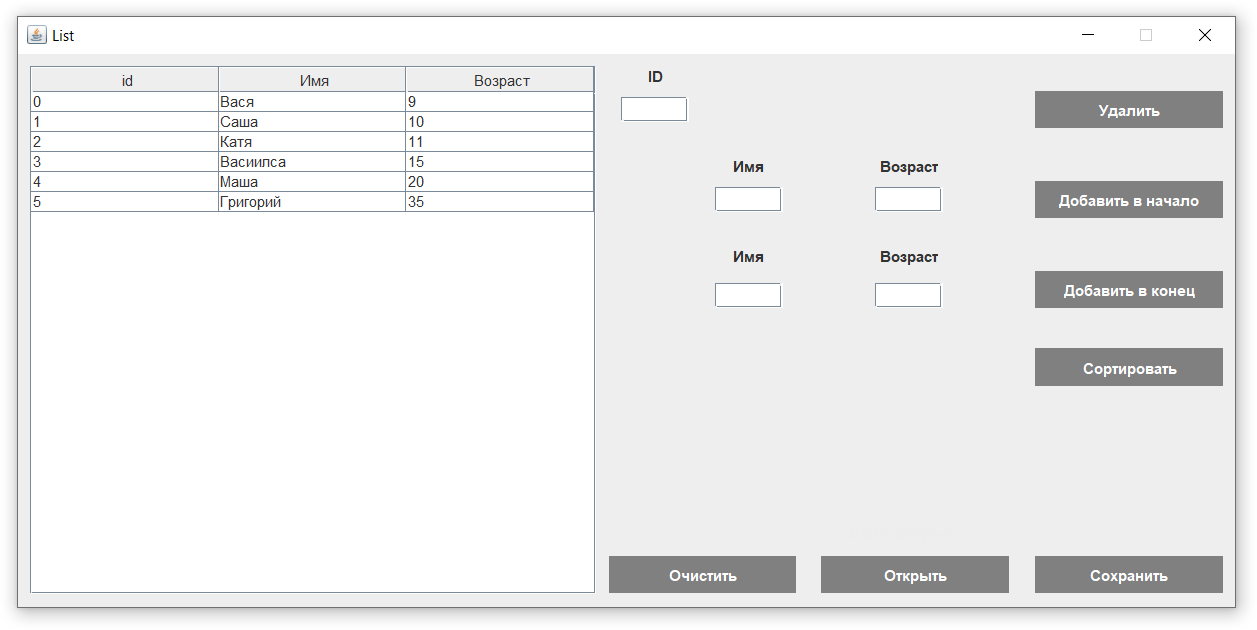


Рисунок 5 – Сортировка элементов

ActionListener для кнопки «Сортировка»:

1. class b\_sortActionListener extends ActionListener {
2. override def actionPerformed(e: ActionEvent): Unit = {
3. list.sort(new Ordering[Int]() {
4. override def compare(o1: Int, o2: Int): Int = {
5. val a = o1.asInstanceOf[Integer]
6. val b = o2.asInstanceOf[Integer]
7. if (a < b) 1
8. else if (a == b) 0
9. else -1
10. }
11. })
12. list.fireTableDataChanged()
13. }
14. }

Вызывает метод sort из класса PersonList:

1. *//Сортировка узлов списка по возрастанию по полю Age*
2. def sort(comparator: Ordering[Int]): Unit = {
3. var k: Int = 0
4. var prev: Person = new Person()
5. prev = null
6. var cur: Person = new Person()
7. cur = null
8. var save: Person = new Person()
9. save = null
10. var save2: Person = new Person()
11. save2 = null
12. if (head != null) {
13. var i = 0
14. for (i <- 0 to getSize) {
15. cur = head
16. prev = head
17. breakable { while (cur.next != null) {
18. if (comparator.compare(cur.age, cur.next.age) == -1) {
19. save = cur
20. save2 = cur.next.next
21. if (cur == head) {
22. head = cur.next
23. cur.next.next = save
24. cur.next.next.next = save2
25. }
26. else if (cur.next.next == null) {
27. prev.next = cur.next
28. cur.next.next = save
29. cur.next.next.next = null
30. break
31. }
32. else {
33. prev.next = cur.next
34. cur.next.next = save
35. cur.next.next.next = save2
36. }
37. }
38. prev = cur
39. cur = cur.next
40. }
41. }
42. }
43. }
44. else System.out.println("Список пуст")
45. }

Листинг кода полностью:

1. import javax.swing.\_
2. import java.awt.\_
3. import java.io.\_
4. import java.util.\_
5. import java.awt.event.ActionEvent
6. import java.awt.event.ActionListener
8. import javax.swing.table.AbstractTableModel
9. import scala.util.control.Breaks.\_
11. object Main {
12. def main(args: Array[String]): Unit = {
13. val gui = new GUI()
14. gui.run()
15. }
16. }
18. class GUI{
19. private val width\_b = 150
20. private val height\_b = 30
21. private var save = ""
22. private val path = "list.txt"
24. def run(): Unit = {
25. var frame = new JFrame("List")
26. frame.setResizable(false)
27. val font = new Font("Arial", Font.PLAIN, 15)
28. *//Укажем размеры окна*
29. frame.setSize(1000, 700)
30. frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE)
31. frame.setLayout(new GridBagLayout)
32. var lable\_id = new JLabel("ID")
33. var lable\_status\_open = new JLabel("")
34. var lable\_status\_save = new JLabel("")
35. lable\_status\_open.setForeground(new Color(0, 120, 34))
36. lable\_status\_open.setFont(font)
37. lable\_status\_save.setFont(font)
38. var lable\_Name = new JLabel("Имя")
39. var lable\_Age = new JLabel("Возраст")
40. var lable\_Name2 = new JLabel("Имя")
41. var lable\_Age2 = new JLabel("Возраст")
42. var list = new PersonList
43. var table = new JTable(list)
45. table.setFillsViewportHeight(true)
46. var textField\_delete = new JTextField(5)
47. var textField\_Name = new JTextField(5)
48. var textField\_age = new JTextField(5)
49. var textField\_id = new JTextField(5)
50. var textField\_Name2 = new JTextField(5)
51. var textField\_age2 = new JTextField(5)
52. var b\_delete = new JButton("Удалить")
53. b\_delete.setPreferredSize(new Dimension(width\_b, height\_b))
54. var b\_add = new JButton("Добавить в начало")
55. b\_add.setPreferredSize(new Dimension(width\_b, height\_b))
56. var b\_add\_in = new JButton("Добавить в конец")
57. b\_add\_in.setPreferredSize(new Dimension(width\_b, height\_b))
58. var b\_sort = new JButton("Сортировать")
59. b\_sort.setPreferredSize(new Dimension(width\_b, height\_b))
60. var b\_save = new JButton("Сохранить")
61. b\_save.setPreferredSize(new Dimension(width\_b, height\_b))
62. var b\_open = new JButton("Открыть")
63. b\_open.setPreferredSize(new Dimension(width\_b, height\_b))
64. b\_delete.setForeground(Color.WHITE)
65. b\_delete.setBackground(Color.GRAY)
66. b\_delete.setBorderPainted(false)
67. b\_delete.setFocusPainted(false)
68. b\_add.setForeground(Color.WHITE)
69. b\_add.setBackground(Color.GRAY)
70. b\_add.setBorderPainted(false)
71. b\_add.setFocusPainted(false)
72. b\_add\_in.setForeground(Color.WHITE)
73. b\_add\_in.setBackground(Color.GRAY)
74. b\_add\_in.setBorderPainted(false)
75. b\_add\_in.setFocusPainted(false)
76. b\_sort.setForeground(Color.WHITE)
77. b\_sort.setBackground(Color.GRAY)
78. b\_sort.setBorderPainted(false)
79. b\_sort.setFocusPainted(false)
80. b\_save.setForeground(Color.WHITE)
81. b\_save.setBackground(Color.GRAY)
82. b\_save.setBorderPainted(false)
83. b\_save.setFocusPainted(false)
84. b\_open.setForeground(Color.WHITE)
85. b\_open.setBackground(Color.GRAY)
86. b\_open.setBorderPainted(false)
87. b\_open.setFocusPainted(false)
88. var b\_clear = new JButton("Очистить")
89. b\_clear.setPreferredSize(new Dimension(width\_b, height\_b))
90. b\_clear.setForeground(Color.WHITE)
91. b\_clear.setBackground(Color.GRAY)
92. b\_clear.setBorderPainted(false)
93. b\_clear.setFocusPainted(false)
94. var scrollPane = new JScrollPane(table)
95. val c = new GridBagConstraints
96. val c2 = new GridBagConstraints
97. frameadd()
98. *//слушатели кликов по кнопкам*
99. b\_delete.addActionListener(new b\_deleteActionListener)
100. b\_sort.addActionListener(new b\_sortActionListener)
101. b\_add.addActionListener(new b\_addActionListener)
102. b\_add\_in.addActionListener(new b\_addLastActionListener)
103. b\_save.addActionListener(new b\_saveActionListener)
104. b\_open.addActionListener(new b\_openActionListener)
105. b\_clear.addActionListener(new b\_clearActionListener)
106. *//Окно программы по центру*
107. frame.setLocationRelativeTo(null)
108. frame.pack()
109. frame.setVisible(true)

112. class b\_addLastActionListener extends ActionListener {
113. override def actionPerformed(e: ActionEvent): Unit = {
114. if (!(textField\_Name2.getText == "") && !(textField\_age2.getText == "")) list.addLast(textField\_Name2.getText, textField\_age2.getText.toInt)
115. list.fireTableDataChanged()
116. }
117. }
119. class b\_addActionListener extends ActionListener {
120. override def actionPerformed(e: ActionEvent): Unit = {
121. if (!(textField\_Name.getText == "") && !(textField\_age.getText == "")) list.add(textField\_Name.getText, textField\_age.getText.toInt)
122. list.fireTableDataChanged()
123. }
124. }
126. class b\_sortActionListener extends ActionListener {
127. override def actionPerformed(e: ActionEvent): Unit = {
128. list.sort(new Ordering[Int]() {
129. override def compare(o1: Int, o2: Int): Int = {
130. val a = o1.asInstanceOf[Integer]
131. val b = o2.asInstanceOf[Integer]
132. if (a < b) 1
133. else if (a == b) 0
134. else -1
135. }
136. })
137. list.fireTableDataChanged()
138. }
139. }
141. class b\_clearActionListener extends ActionListener {
142. override def actionPerformed(e: ActionEvent): Unit = {
143. while ( {
144. list.getRowCount > 0
145. }) list.remove\_At(0)
146. list.fireTableDataChanged()
147. }
148. }
150. class b\_deleteActionListener extends ActionListener {
151. override def actionPerformed(e: ActionEvent): Unit = {
152. if (!(textField\_delete.getText == "")) list.remove\_At(textField\_delete.getText.toInt)
153. else if (table.getSelectionModel.getLeadSelectionIndex >= 0) list.remove\_At(table.getSelectionModel.getLeadSelectionIndex)
154. list.fireTableDataChanged()
155. }
156. }
158. class b\_saveActionListener extends ActionListener {
159. override def actionPerformed(e: ActionEvent): Unit = {
160. try {
161. var writer = new FileWriter("List.txt", false)
162. *// запись всей строки*
163. list.todo(new IForEach() {
164. override def toDo(list: Person): Unit = {
165. save = list.Name + ":" + list.age
166. try {
167. writer.write(save)
168. writer.append('**\n**')
169. val thread2 = new MyThreadSave
170. thread2.start()
171. } catch {
172. case ioException: IOException =>
173. ioException.printStackTrace()
174. }
175. }
176. })
177. writer.flush()
178. } catch {
179. case ex: IOException =>
180. System.out.println(ex.getMessage)
181. lable\_status\_save.setText("Ошибка сохранения")
182. }
183. }
184. }
186. class b\_openActionListener extends ActionListener {
187. override def actionPerformed(e: ActionEvent): Unit = {
188. while ( {
189. list.getRowCount > 0
190. }) list.remove\_At(0)
191. var file = new File(path)
192. try {
193. val scanner = new Scanner(file)
194. var line = ""
195. var words: Array[String] = null
196. while ( {
197. scanner.hasNext
198. }) {
199. line = scanner.nextLine
200. words = line.split(":")
201. list.addLast(words(0), words(1).toInt)
202. }
203. val thread = new MyThreadOpen
204. thread.start()
205. list.fireTableDataChanged()
206. } catch {
207. case ex: FileNotFoundException =>
208. ex.printStackTrace()
209. }
210. }
211. }
213. class MyThreadOpen extends Thread {
214. override def run(): Unit = {
215. lable\_status\_open.setForeground(new Color(0, 120, 34, 255))
216. lable\_status\_open.setText("Файл открыт")
217. try {
218. Thread.sleep(1000)
219. var alpha = 255
220. while ( {
221. alpha > 0
222. }) {
223. Thread.sleep(1)
224. lable\_status\_open.setForeground(new Color(0, 120, 34, alpha))
226. {
227. alpha -= 1;
228. alpha + 1
229. }
230. }
231. } catch {
232. case e: InterruptedException =>
233. e.printStackTrace()
234. }
235. }
236. }
238. class MyThreadSave extends Thread {
239. override def run(): Unit = {
240. lable\_status\_save.setForeground(new Color(0, 120, 34, 255))
241. lable\_status\_save.setText("Файл сохранен")
242. try {
243. Thread.sleep(1000)
244. var alpha = 255
245. while ( {
246. alpha > 0
247. }) {
248. Thread.sleep(1)
249. lable\_status\_save.setForeground(new Color(0, 120, 34, alpha))
251. {
252. alpha -= 1;
253. alpha + 1
254. }
255. }
256. } catch {
257. case e: InterruptedException =>
258. e.printStackTrace()
259. }
260. }
261. }
263. def frameadd(): Unit = {
264. frame.add(scrollPane, new GridBagConstraints(0, 0, 1, 8, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(10, 10, 10, 10), 0, 0))
265. frame.add(textField\_delete, new GridBagConstraints(1, 1, 1, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(2, 10, 20, 10), 0, 0))
266. frame.add(b\_delete, new GridBagConstraints(4, 1, 3, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(2, 10, 20, 10), 0, 0))
267. frame.add(textField\_Name, new GridBagConstraints(2, 3, 1, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(2, 10, 20, 10), 0, 0))
268. frame.add(textField\_age, new GridBagConstraints(3, 3, 1, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(2, 10, 20, 10), 0, 0))
269. frame.add(b\_add, new GridBagConstraints(4, 3, 3, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(2, 10, 20, 10), 0, 0))
270. frame.add(textField\_Name2, new GridBagConstraints(2, 5, 1, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(2, 10, 20, 10), 0, 0))
271. frame.add(textField\_age2, new GridBagConstraints(3, 5, 1, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(2, 10, 20, 10), 0, 0))
272. frame.add(b\_add\_in, new GridBagConstraints(4, 5, 3, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(2, 10, 30, 10), 0, 0))
273. frame.add(b\_sort, new GridBagConstraints(4, 6, 3, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(2, 10, 10, 10), 0, 0))
274. frame.add(lable\_Name, new GridBagConstraints(2, 2, 1, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(2, 10, 2, 10), 0, 0))
275. frame.add(lable\_Age, new GridBagConstraints(3, 2, 1, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(2, 10, 2, 10), 0, 0))
276. frame.add(lable\_id, new GridBagConstraints(1, 0, 1, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(10, 10, 2, 10), 0, 0))
277. frame.add(lable\_Name2, new GridBagConstraints(2, 4, 1, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(2, 10, 2, 10), 0, 0))
278. frame.add(lable\_Age2, new GridBagConstraints(3, 4, 1, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(2, 10, 2, 10), 0, 0))
279. frame.add(b\_save, new GridBagConstraints(4, 7, 2, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(125, 10, 10, 10), 0, 0))
280. frame.add(b\_open, new GridBagConstraints(2, 7, 2, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(125, 95, 10, 10), 0, 0))
281. frame.add(lable\_status\_open, new GridBagConstraints(2, 7, 3, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(70, 60, 20, 0), 0, 0))
282. frame.add(lable\_status\_save, new GridBagConstraints(3, 7, 3, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(70, 170, 20, 0), 0, 0))
283. frame.add(b\_clear, new GridBagConstraints(1, 7, 2, 1, 0, 0, GridBagConstraints.CENTER, GridBagConstraints.CENTER, new Insets(125, 0, 10, 0), 0, 0))
284. }
285. }
286. }

289. class PersonList extends AbstractTableModel{
290. private var head: Person = null
291. private val columnCount = 3
293. *//проверка на пустой список*
294. private def isEmpty = head == null
296. *//удаление по индексу*
297. def remove\_At(key: Int): Unit = {
298. var cur = head
299. var prev = head
300. if (head != null) {
301. var i = 0
302. while ( {
303. i < getSize
304. }) {
305. if (i == key) if (cur eq head) {
306. head = head.next
307. return
308. }
309. else {
310. prev.next = cur.next
311. return
312. }
313. prev = cur
314. cur = cur.next
316. {
317. i += 1;
318. i - 1
319. }
320. }
321. }
322. }
324. def getSize: Int = {
326. var cur = new Person
327. cur = head
329. var i = 0
330. while (cur != null) {
331. i += 1
332. cur = cur.next
333. }
334. i
335. }
337. def eachStep(forEach: IForEach): Unit = {
338. var tmp = head
339. System.out.println("Вывод списка на печать todo:")
340. while ( {
341. tmp != null
342. }) {
343. forEach.toDo(tmp)
344. tmp = tmp.next
345. }
346. }
348. def printList(): Unit = {
349. var tmp = new Person
350. tmp = head
352. println("Вывод списка на печать:")
353. while ( {
354. tmp != null
355. }) {
356. println(tmp.Name + ":" + tmp.age)
357. tmp = tmp.next
358. }
359. }
361. *//добавление узла в начало*
362. def add(Name: String, age: Int): Unit = {
363. val tmp = new Person(Name, age)
364. tmp.next = head
365. head = tmp
366. }
368. *//добавление узла в конец*
369. def addLast(Name: String, age: Int): Unit = {
370. val tmp = new Person(Name, age)
371. var cur = head
372. if (cur == null) {
373. tmp.next = head
374. head = tmp
375. return
376. }
377. while ( {
378. cur.next != null
379. }) cur = cur.next
380. cur.next = tmp
381. tmp.next = null
382. }
384. *//Извлечение данных узла по индексу*
385. def getData(key: Int): Array[String] = {
386. var cur = head
387. var rows :Array[String] = new Array[String](3)
388. if (head != null) {
389. var i = 0
390. while ( {
391. i < getSize
392. }) {
393. if (i == key) {
394. rows(0) = "" + i
395. rows(1) = cur.Name
396. rows(2) = "" + cur.age
397. }
398. println(rows)
399. cur = cur.next
401. {
402. i += 1; i - 1
403. }
404. }
405. }
406. return rows
407. }
409. *//Сортировка узлов списка по возрастанию*
410. def sort(comparator: Ordering[Int]): Unit = {
411. var k: Int = 0
412. var prev: Person = new Person()
413. prev = null
414. var cur: Person = new Person()
415. cur = null
416. var save: Person = new Person()
417. save = null
418. var save2: Person = new Person()
419. save2 = null
420. if (head != null) {
421. var i = 0
422. for (i <- 0 to getSize) {
423. cur = head
424. prev = head
425. breakable { while (cur.next != null) {
426. if (comparator.compare(cur.age, cur.next.age) == -1) {
427. save = cur
428. save2 = cur.next.next
429. if (cur == head) {
430. head = cur.next
431. cur.next.next = save
432. cur.next.next.next = save2
433. }
434. else if (cur.next.next == null) {
435. prev.next = cur.next
436. cur.next.next = save
437. cur.next.next.next = null
438. break
439. }
440. else {
441. prev.next = cur.next
442. cur.next.next = save
443. cur.next.next.next = save2
444. }
445. }
446. prev = cur
447. cur = cur.next
448. }
449. }
450. }
451. }
452. else System.out.println("Список пуст")
453. }
455. override def getRowCount: Int = getSize
457. override def getColumnCount: Int = columnCount
459. override def getValueAt(rowIndex: Int, columnIndex: Int): Object = {
460. val rows : Array[String] = getData(rowIndex)
461. println(rows)
462. rows(columnIndex)
463. }
465. override def getColumnName(columnIndex: Int): String = {
466. columnIndex match {
467. case 0 =>
468. return "id"
469. case 1 =>
470. return "Имя"
471. case 2 =>
472. return "Возраст"
473. }
474. ""
475. }
476. }

**Заключение**

В ходе выполнения данной лабораторной работы на языке Scala было реализовано оконное приложения для работы со структурой данных односвязный список. Для структуры были реализованы методы добавления, вставки, удаления, получения значения по индексу, а также метод сортировки списка.