МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по учебной практике

Тема: Визуализация сортировок

Студент гр. 0381 Ибатов Н.Э.

Студент гр. 0381 Печёркин А.С.

Студент гр. 0381 Котов Д.А.

Руководитель Ефремов М.А.

Санкт-Петербург
2022

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студент Котов Д.А группы 0381

Студент Ибатов Н.Э группы 0381

Студент Печеркин А.С группы 0381

Тема практики: Различные алгоритмы сортировок

Задание на практику:

Командная итеративная разработка визуализатора алгоритма(ов) на Java с графическим интерфейсом.

Алгоритмы: Быстрая сортировка (quicksort), Сортировка пузырьком (bubble sort), Битонная сортировка (bitonic sort).

Сроки прохождения практики: 29.06.2022 – 12.07.2022

Дата сдачи отчета: 12.07.2022

Дата защиты отчета: 12.07.2022

Студент		Котов Д.А.
Студент	_____	Ибатов Н.Э.
Студент	_____	Печеркин А.С.
Руководитель	_____	Ефремов М.А.
	_____	

АННОТАЦИЯ

Цель учебной практики заключается в изучении языка программирования Java и получении навыков работы в команде.

Основная задача - разработка приложения с графическим интерфейсом, которое позволяет визуализировать различные алгоритмы сортировки.

SUMMARY

The purpose of the training practice is to learn the Java programming language and gain teamwork skills.

The main task is to develop an application with a graphical interface that allows you to visualize various sorting algorithms.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью учебной практики является разработка приложения для визуализации работы алгоритмов сортировок: quicksort, bubble sort и bitonic sort. Приложение разрабатывается на языке Java с графическим интерфейсом с использованием Swing. Пользователю предоставляется возможность ввести заданный массив или сгенерировать его случайным образом. Информация о работе алгоритмов выводится на экран.

Задание выполняется командой, где за каждым участником поставлены определенный задачи. Готовая программа собирается в jar-архив.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные требования к программе

1.1.1. Требования к вводу исходных данных

Должна быть возможность ввода данных из файла и генерации случайного массива заданного размера, а также ввод пользователем вручную.

1.1.2. Требования к визуализации

Необходимо реализовать визуализацию сортировки массива с возможностью выполнения в автоматическом режиме и в пошаговом режиме. В пошаговом режиме должна быть возможность перехода к следующему и предыдущему шагу, перейти к первому и последнему шагу алгоритма, запустить и приостановить автоматический режим.

1.1.3. Сценарии использования

1) Считать массив из файла

   Пользователю предлагается выбрать файл, из которого будет считан массив с данными для сортировки.

2) Сгенерировать массив

   Пользователю предлагается сгенерировать массив заданного размера, который будет перемешан случайным образом.

3) Ввести массив вручную

   Пользователю предлагается ввести значения массива вручную в текстовое поле.

4) Выбрать алгоритм сортировки

   Пользователю предлагается выбор алгоритма сортировки, который будет визуализирован.

5) Визуализировать алгоритм в ручном режиме

Пользователю предоставляется возможность просмотреть работу алгоритма в пошаговом режиме, при этом ему доступны переход к первому шагу, переход к последнему шагу, переход к следующему и предыдущему шагам работы алгоритма. Также пользователь может запустить и остановить автоматический режим.

6) Визуализировать алгоритм в автоматическом режиме.

Пользователю предлагается запустить сортировку в автоматическом режиме, то есть шаги алгоритма будут выполняться последовательно с некоторой задержкой.

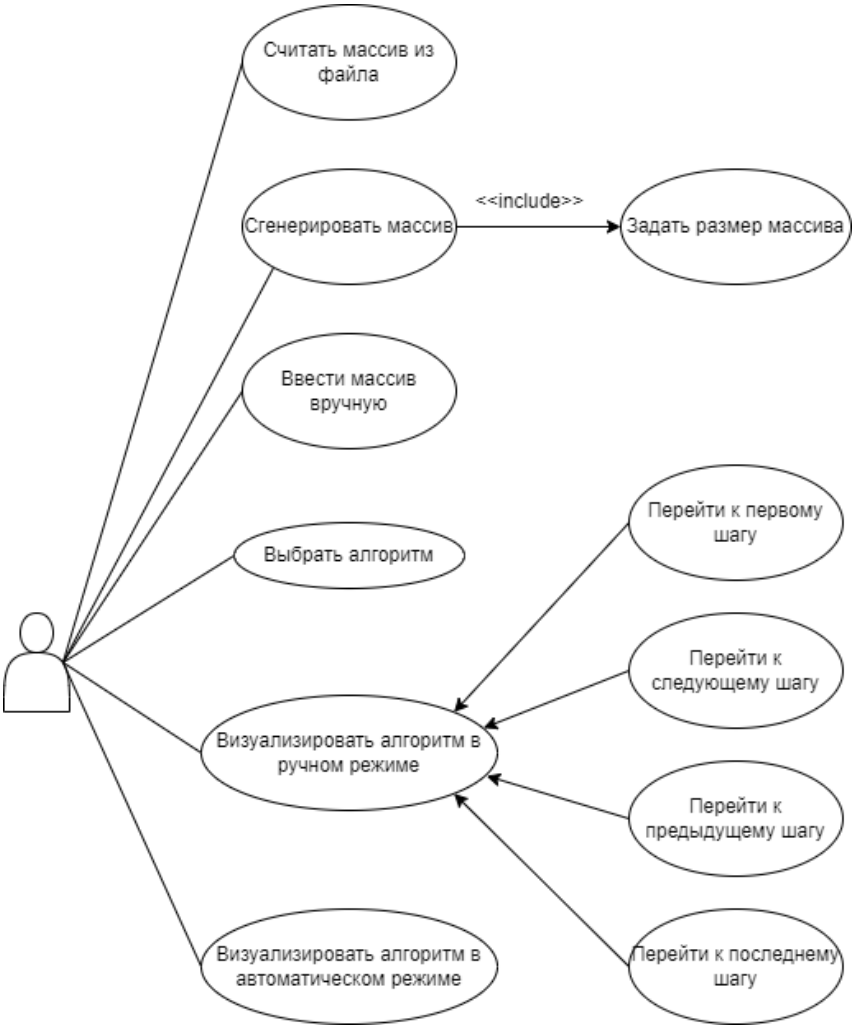

Рис. 1. Диаграмма сценариев использования

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

04.07 - Реализован интерфейс приложения на заглушках. Реализована структура данных и требующийся алгоритм. Реализованы тесты для структуры данных и алгоритму.

06.07 - Реализована генерация данных: из файла, случайная, при помощи графического интерфейса. Реализована кнопка "Показать результат" с выполнением и отображением итогового результата работы алгоритма.

08.07 - Элементы графического интерфейса, отвечающие за пошаговое выполнение (кнопки, лог) - реализованы и корректно работают. Реализованы структуры данных, отвечающие за пошаговое выполнение алгоритма. Реализованы тесты структур данных, отвечающих за пошаговое выполнение алгоритма.

10.07 - Программа работает корректно, собирается в исполняемый jar-архив при помощи maven из консоли.

2.2. Распределение ролей в бригаде

Котов Д.А. - реализация алгоритмов сортировки.

Ибатов Н.Э. - реализация логики взаимодействия модели и пользовательского интерфейса и тестирование алгоритмов.

Печеркин А.С. - реализация пользовательского интерфейса.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Описание графического интерфейса

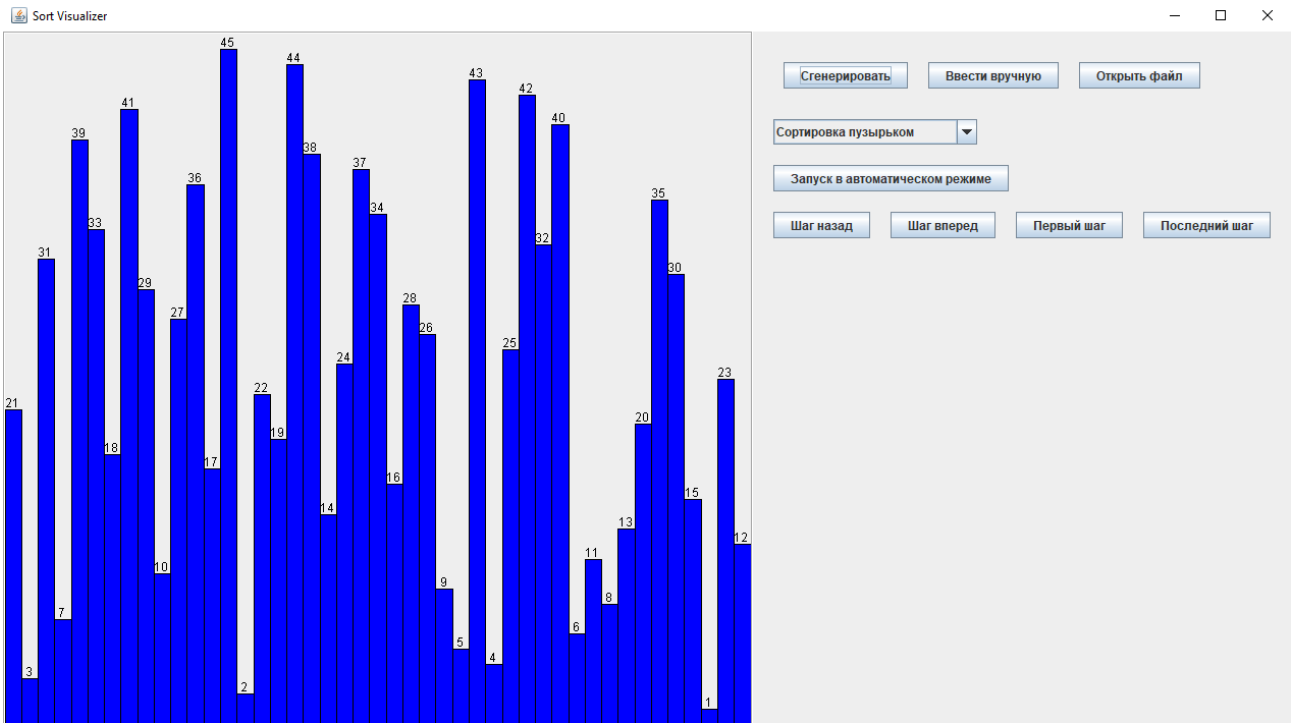

Рис. 2. Скриншот интерфейса

Кнопка “Сгенерировать” отвечает за создание перемешанного массива заданного размера. Размер задается во всплывающем окне при нажатии на кнопку.

Кнопка “Ввести вручную” вызывает окно, в которое пользователь может ввести элементы массива, разделенные пробелами.

Кнопка “Открыть файл” вызывает окно выбора файла, из которого будет считан массив.

Алгоритм сортировки выбирается в выпадающем списке. Доступные варианты: сортировка пузырьком, быстрая сортировка, битонная сортировка.

Кнопка “Запуск в автоматическом режиме” отвечает за запуск анимации сортировки.

Кнопки “Шаг назад”, “Шаг вперед”, “Первый шаг”, “Последний шаг” отвечают за переход по шагам алгоритма в пошаговом режиме.

3.2. Структура программы

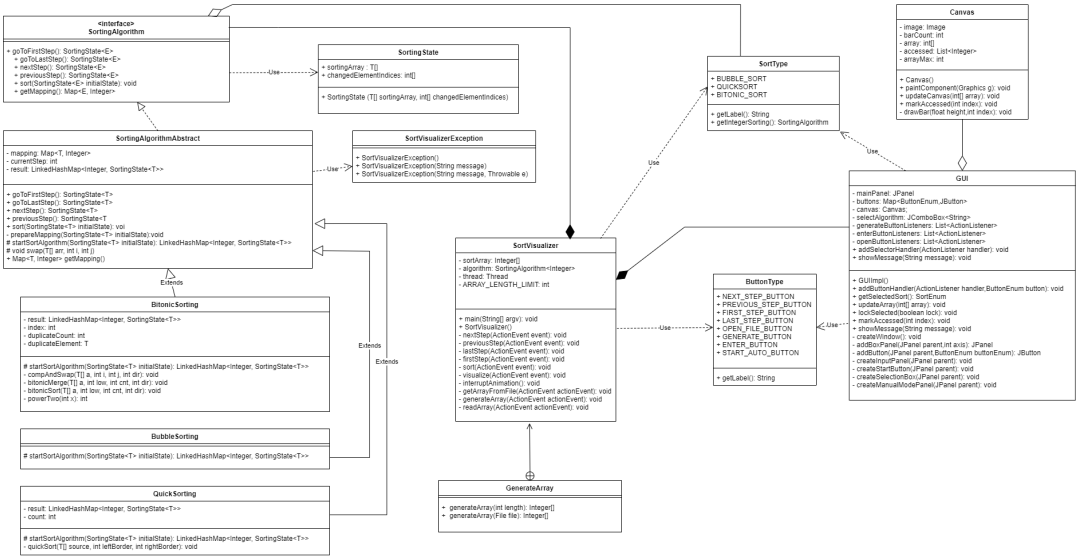

Рис. 3. Диаграмма классов

Основной класс программы SortVisualizer создает экземпляр класса GUI. Добавляет обработчики кнопок на интерфейсе и, с помощью класса GenerateArray, позволяет сгенерировать массив, прочитать из файла.

После создания массива происходит его сортировка выбранным способом, для этого используется интерфейс SortingAlgorithm и его реализации для различных сортировок. Результат сортировки сохраняется в SortingState, который затем отображается в автоматическом или пошаговом режиме.

За отрисовку интерфейса отвечает класс GUI, который реализует интерфейс GUI. При создании экземпляра класса GUI, создается основное окно программы. В этом классе реализована возможность добавления обработчиков кнопок интерфейса. Для удобного обращения к кнопкам на интерфейсе было создано перечисление с кнопками. Для отрисовки сортируемого массива был создан класс Canvas, который реализует интерфейс JComponent, в нем при помощи метода paintComponent происходит отрисовка массива.

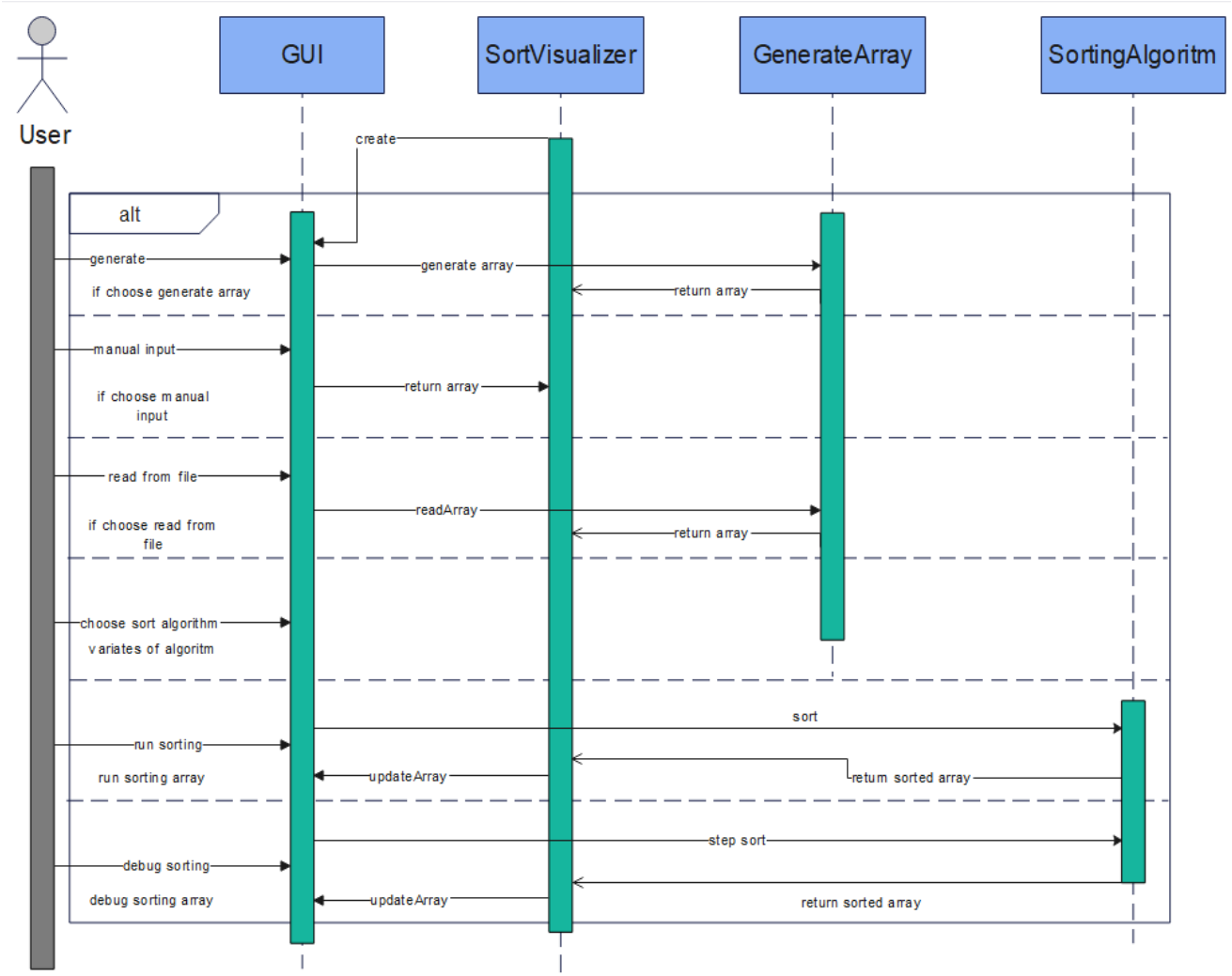

Рис. 4. Диаграмма последовательности.

У пользователя есть возможность выбрать один из трех форматов сгенерировать массив (автоматический, задав длину массива; ввести массив через пробел; считать массив из файла). Также у пользователя есть возможность выбрать, каким алгоритмом будет сортироваться массив и 2 режима запуска программы (в автоматическом, в режиме отладки). После ввода массива он отображается на экран. После нажатия запуска алгоритма, программа его сортирует с отображением действий.

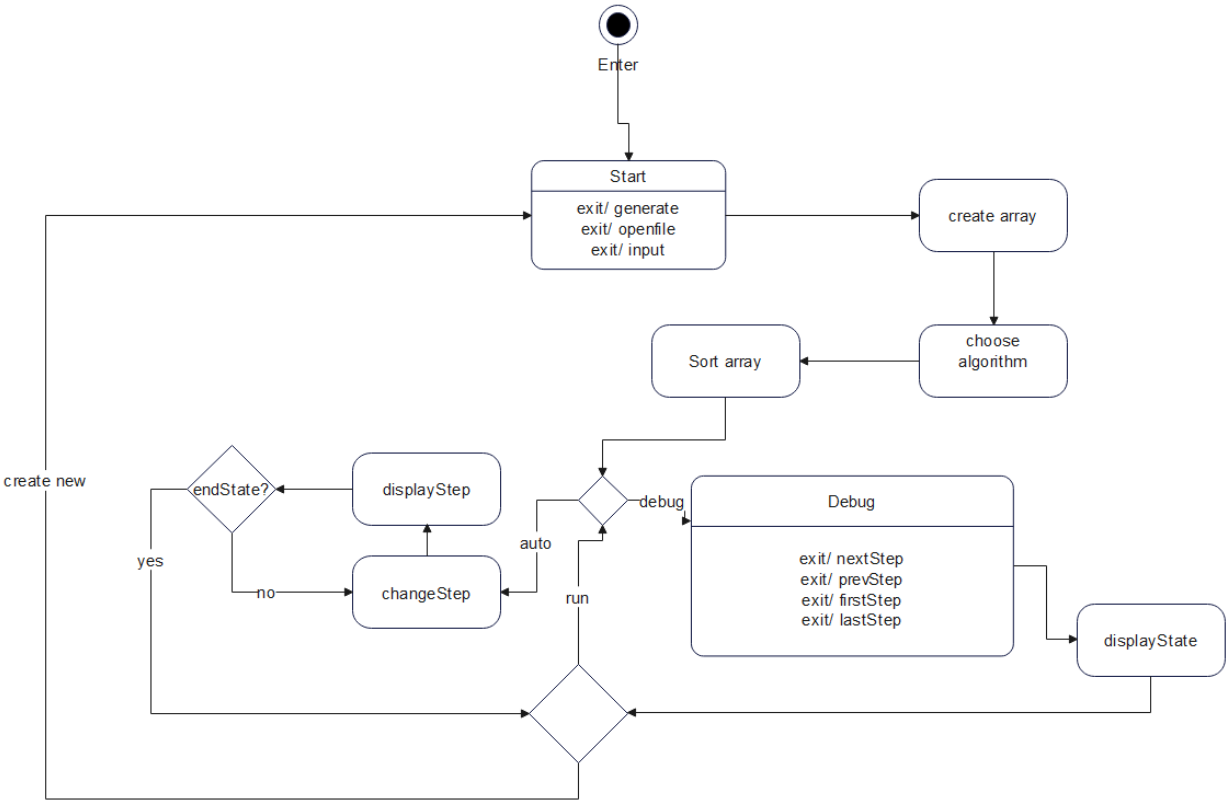

Рис. 5. Диаграмма состояний.

После запуска программы доступны функции генерации массива, загрузки из файла, и ручного ввода (состояние Start на диаграмме). При выборе одного из вариантов происходит создание массива, затем при помощи выбранного алгоритма этот массив сортируется, при этом после каждой перестановки значений состояние массива сохраняется в список. Далее пользователю предоставляется выбор: просмотреть анимацию сортировки или пройти по сохраненным состояниям в ручном режиме. При выборе анимации программа переходит в цикл, в котором последовательно отображает сохраненные состояния. В ручном режиме пользователь может выбрать на какой шаг он хочет перейти. При переходе на какой-то шаг из списка состояний берется состояние соответствующее данному шагу, затем это состояние выводится на экран.

4. ТЕСТИРОВАНИЕ

Для тестирования используется фреймворк JUnit 5.

4.1. Тестирование алгоритмов

Для каждого алгоритма сортировки реализован метод `sortArray()`, который проверяет корректность сортировки массива и `emptyArray()` , который сортирует пустой массив.

Для класса `SortingAlgorithm` реализованы тесты для пошаговых методов.

Метод `nextStepException()` проверяет на то, что если текущий шаг последний, то если перейти на следующий шаг, то вызывается исключение.

Метод `previousStepException()` проверяет на то, что если сразу перейти на шаг назад, то вызывается исключение.

Метод `firstStepException()` вызывает исключение, если массив еще не задан.

Метод `lastStepException()` вызывает исключение, если массив еще не задан.

Тест на сортировку необъявленного массива выкидывает NullPointerException. Алгоритмы сортируют массивы любой длины, но для нормального отображения массива на экране и времени выполнения алгоритмы размер массива ограничен 1-500.

Таблица 1. Тестирование алгоритмов.

Алгоритм	Ввели	Ожидалось	Результат	Комментарий
QuickSort	`{5,8,2,7,3,1,6,9,4}`	`{1,2,3,4,5,6,7,8,9}`	`{1,2,3,4,5,6,7,8,9}`	Корректность алгоритма
QuickSort	{}	{}	{}	Пустой массив
QuickSort	null	NullPointerException	NullPointerException	Не объявленный массив

BubbleSort	{5,8,2,7,3,1,6,9,4}	{1,2,3,4,5,6,7,8,9}	{1,2,3,4,5,6,7,8,9}	Корректность алгоритма
BubbleSort	{}	{}	{}	Пустой массив
BubbleSort	null	NullPointerException	NullPointerException	Не объявленный массив
BitonicSort	{1,3,4,6,2,7,5,8}	{1,2,3,4,5,6,7,8}	{1,2,3,4,5,6,7,8}	Корректность алгоритма
BitonicSort	{}	{}	{}	Пустой массив
BitonicSort	null	NullPointerException	NullPointerException	Не объявленный массив

ЗАКЛЮЧЕНИЕ

В ходе работы были выполнены следующие задачи:

- Изучены основы языка Java по онлайн-курсу.
- Реализован интерфейс приложения с использованием библиотеки Swing.
- Реализованы алгоритмы сортировок и проведено их тестирование при помощи фреймворка модульного тестирования JUnit.
- Реализована возможность пошагового просмотра работы алгоритмов.
- Был настроен maven-проект для сборки исполняемого файла из консоли.

Таким образом, задача практики была выполнена, а именно, было разработано приложение с графическим интерфейсом, позволяющее визуализировать работу алгоритмов сортировки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шилдт Г. Java. Полное руководство. Москва: “Диалектика-Вильямс”, 2018
2. Хорстманн К. Java. Библиотека профессионала. Том 1. Основы. Москва: “Вильямс”, 2018
3. Goetz B. Java Concurrency in Practice. Boston: “Addison-Wesley Professional”, 2006
4. Urma R. Modern Java in Action. New-York: “Manning”, 2018
5. Oaks S. Java Performance In-Depth Advice for Tuning and Programming. Newton: “O'Reilly Media”, 2020

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

https://github.com/lexapech/sort-visualizer