

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

ОТЧЕТ
по учебной практике
Тема: Различные алгоритмы сортировки

Студент гр. 9383	_____	Нистратов Д.Г.
Студент гр. 9383	_____	Звега А.Р.
Руководитель	_____	Ефремов М.А.

Санкт-Петербург
2021

ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ

Студент Нистратов Д.Г. группы 9383

Студент Звега А.Р. группы 9383

Тема практики: Различные алгоритмы сортировки

Задание на практику:

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

Алгоритмы: Сортировка пузырьком, Сортировка перемешиванием, Пирамидальная сортировка.

Сроки прохождения практики: 01.07.2021 – 14.07.2021

Дата сдачи отчета: 03.07.2021

Дата защиты отчета: 14.07.2021

Студент	_____	Нистратов Д.Г.
---------	-------	----------------

Студент	_____	Звега. А.Р.
---------	-------	-------------

Руководитель	_____	Ефремов М.А.
--------------	-------	--------------

АННОТАЦИЯ

Целью данной учебной практики является разработка приложения для визуализации работы алгоритмов сортировки: Сортировка пузырьком, Сортировка перемешиванием, Пирамидальная сортировка. Приложение разрабатывается на языке Java с графическим интерфейсом. Пользователю предоставляется возможность введения собственных элементов для сортировки, пошаговая визуализация алгоритмов, а также сравнение различных алгоритмов сортировки. Приложение должно быть простым и удобным в управлении.

Разработка выполняется командой, за участниками распределены определенные роли в создании приложения. Выполнение работы и составление отчета осуществляется поэтапно.

SUMMARY

The purpose of this training practice is to develop an application for visualizing the operation of sorting algorithms: Bubble sort, Cocktail sort, Heap sort. The application is developed in the Java language with a graphical interface. The user is given the opportunity to input their own elements to sort, step-by-step visualization of algorithms, as well as comparison of various sorting algorithms. The application should be simple and easy to manage.

Development is carried out by a team that has certain roles assigned to them. The work and the report are completed in stages.

СОДЕРЖАНИЕ

	Введение	5
1.	Требования к программе	6
1.1.	Исходные требования к программе	6
2.	План разработки и распределение ролей в бригаде	7
2.1.	План разработки	7
2.2.	Распределение ролей в бригаде	7

ВВЕДЕНИЕ

Целью учебной практики является создание приложения, визуализирующего работу различных алгоритмов сортировки. Приложение написано на языке Java. Пользователю предоставляется возможность ввести заданный массив элементов или сгенерировать случайный, пошагово запустить алгоритм, а также сравнить несколько алгоритмов одновременно. Информация о работе алгоритмов выводится на экран.

Задание выполняется командой, в которой за каждым участником закреплены задачи. Готовая программа собирается в один исполняемый jar-архив.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные Требования к программе

Исходные требования были описаны в виде use-case диаграммы, см.

Изображение 1. use-case диаграмма

1.1.1. Требования к вводу исходных данных.

Пользователь выбирает кол-во элементов для создания массива чисел, а также алгоритм сортировки. После указания кол-ва элементов, пользователю представляется выбор: генерация случайного расположения элементов от 0 до выбранного кол-ва элементов, либо ввод заданного кол-ва элементов.

1.1.2. Требования к визуализации

Результатом работы программы должно являться анимация сортировки выбранного алгоритма пользователем, с возможностью остановить визуализацию для просмотра пошаговой визуализации алгоритма. Пошаговая визуализация алгоритма предоставляет возможность перехода к следующему или предыдущему шагу анимации, а также возобновления анимации алгоритма сортировки.

1.1.3 Обоснование Use-case диаграммы.

1) Выбор размера массива, ограничен от 2 до 200

Пользователь выбирает размер массива для генерирования или ввода значений, значение ограничено в рамках от 2 до 200 для удобной визуализации работы алгоритма.

2) Создание элементов, от 1 до 1000

Пользователь создает массив заданным ранее размером. Значения ограничены в рамках от 1 до 1000 для удобной визуализации работы алгоритма.

3) Генерация элементов от 1 до размера массива

Программа генерирует массив от 1 до выбранного пользователем размера, элементы в массиве начинаются с 1, а каждый следующий элемент на 1 больше. Далее массив размещивается с помощью рандомизации.

4) Ввод элементов

Программа просит у пользователя ввести массив через вспомогательное окно.

5) Выбор алгоритма сортировки

Пользователь выбирает алгоритм сортировки из списка для визуализации. В списке присутствуют следующие алгоритмы: Сортировка пузырьком, Сортировка перемешиванием, Пирамидальная сортировка, Сравнение сортировок.

6) Визуализация алгоритма

Пользователь запускает визуализацию алгоритма с задержкой. На каждом шаге визуализации отмечаются измененные элементы массива.

7) Пошаговая визуализация алгоритма

Пользователю представляется выбор, следующий шагов алгоритма визуализации: Следующий шаг визуализации, Предыдущий шаг визуализации, Остановка работы визуализации, Продолжение работы визуализации, последний шаг визуализации, Первый шаг визуализации.



Изображение 1 - UML use-case диаграмма.

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

04.07.2021 Изучение и реализация алгоритмов сортировки Bubble sort, Cocktail Sort, Heap Sort.

04.07.2021 Изучение библиотеки Swing для реализации графического интерфейса, создание концепта GUI приложения.

06.07.2021 Тестирование алгоритмов.

07.07.2021 Имплементация визуализации алгоритмов в GUI приложении.

08.07.2021 Реализация пошаговой визуализации работы алгоритмов.

08.07.2021 Реализация многопоточности алгоритмов.

09.07.2021 Тестирование GUI приложения.

10.07.2021 Реализация визуализации сравнения алгоритмов.

11.07.2021 Исправление недочетов, подготовка к сдаче финальной версии программы.

2.2. Распределение ролей в бригаде

Нистратов Д.Г. – многопоточность, визуализация программы, тестирование визуализации.

Звега А.Р. – сортировка пузырьком, сортировка перемешиванием, пирамидальная сортировка, тестирование алгоритмов.

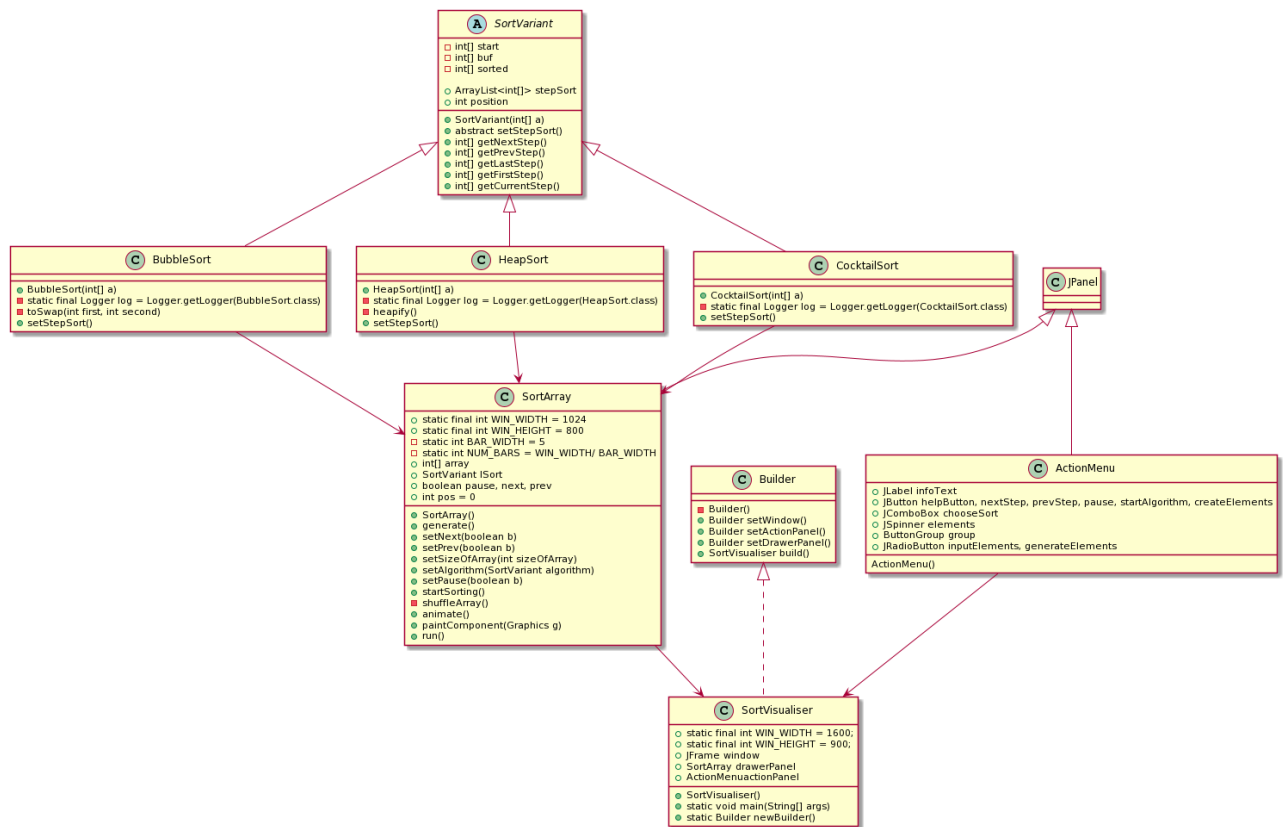
3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1 UML диаграмма классов.

Изображение 2 uml диаграмма классов

Описание:

- 1) Классы BubbleSort, HeapSort и CocktailSort наследуются от абстрактного класса SortVariant.
- 2) Классы SortArray и ActionMenu наследуются от класса JPanel.
- 3) Класс Builder является внутренним классом SortVisualiser.
- 4) Закрашенными стрелками показано использование класса в другом классе. К примеру BubbleSort, HeapSort и CocktailSort используются в SortArray.



Изображение 2 - UML диаграмма классов.

3.2 Описание алгоритмов и их реализация.

Сортировка пузырьком:

Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется перестановка элементов. Проходы по массиву повторяются $n-1$ раз или до тех пор, пока на очередном проходе не окажется, что перестановки больше не нужны, что означает — массив отсортирован.

Сортировка перемешиванием:

Усовершенствованный алгоритм пузырьковой сортировки. Отличия состоят в том, что если при движении по части массива перестановки не происходят, то эта часть массива уже отсортирована и, следовательно, её можно исключить из рассмотрения. Так же при движении от конца массива к началу минимальный элемент ставится на первую позицию, а максимальный элемент сдвигается только на одну позицию вправо. Массив просматривается поочередно справа налево и слева направо.

Пирамидальная сортировка:

Сортировка пирамидой использует бинарное сортирующее дерево. Сортирующее дерево — это такое дерево, у которого выполнены условия:

Каждый лист имеет глубину либо d , либо $d-1$, d — это максимальная глубина дерева.

Значение в любой вершине не меньше (другой вариант — не больше) значения её потомков.

Удобная структура данных для сортирующего дерева — такой массив `Array`, что `Array[0]` — элемент в корне, а потомки элемента `Array[i]` являются `Array[2i+1]` и `Array[2i+2]`.

Алгоритм сортировки будет состоять из двух основных шагов:

1. Выстраиваем элементы массива в виде сортирующего дерева:

$$\text{Array}[i] \geq \text{Array}[2i+1]$$
$$\text{Array}[i] \geq \text{Array}[2i+2]$$

при $i < n/2$.

2. Далее удаляются элементы из корня по одному за раз и перестраивается дерево. То есть на первом шаге меняются $\text{Array}[0]$ и $\text{Array}[n-1]$, преобразовываем $\text{Array}[0], \text{Array}[1], \dots, \text{Array}[n-2]$ в сортирующее дерево. Затем переставляем $\text{Array}[0]$ и $\text{Array}[n-2]$, преобразовываем $\text{Array}[0], \text{Array}[1], \dots, \text{Array}[n-3]$ в сортирующее дерево. Процесс продолжается до тех пор, пока в сортирующем дереве не останется один элемент. Тогда $\text{Array}[0], \text{Array}[1], \dots, \text{Array}[n-1]$ — упорядочен.

4. ТЕСТИРОВАНИЕ

4.1 Тестирование алгоритмов.

Тестирование алгоритмов реализовано в классе TestSort в котором используется Junit. Тесты алгоритмов, проверяют правильно ли работает алгоритм, то есть верно ли отсортирован массив.

- 1) Первый тест проверяет, верно ли сортируется случайный массив.
- 2) Второй тест проверяет, что будет если подать массив, в котором одинаковые элементы.
- 3) Третий тест проверяет, что будет если подать пустой массив.
- 4) Так же тестируются методы прохода по массиву перестановок.

Эти тесты покрывают все возможные ситуации так как, метод сортировки работает с внутренними полями класса, которые инициализируются при создании объекта. А при создании объекта класса исключительных ситуаций не так много. Даже если отойти от методов и классов, алгоритмы сортировок устроены так что они работают какой бы ты массив в них не передал, проблемой становится только некоторая функция сравнения (компаратор) элементов массива, но в конкретной реализации, в которой используются целые числа, даже в ней каких либо ошибок быть не может.

Тесты прохода по массиву перестановок, являются простейшими методами которые добавлены исключительно для упрощения обработки результатов сортировки, поэтому проверка заключается только в том чтобы эти функции, не выходили за границы массива перестановок.

ЗАКЛЮЧЕНИЕ

В ходе выполнения практической работы, были реализованы алгоритмы сортировок. Были изучены вспомогательные средства Junit и log4j. Был получен навык создания Maven файла, для включения зависимостей и параметров сборки. Был написан GUI интерфейс для взаимодействия с программой, в котором можно задавать параметры сортировок, так же он обладает функционалом, который наглядно показывает работу алгоритма в пошаговом режиме. Для написания GUI, была изучена библиотека Swing.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. JDK 11 Documentation. Date Views 2021.07.02

docs.oracle.com/en/java/javase/11/.

2. Package javax.swing. Date Views 2021.07.02

docs.oracle.com/en/java/javase/11/docs/api/java.desktop/javax/swing/package-summary.html

3. Maven Documentation. Date Views 2021.07.02 <https://maven.apache.org/>

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ