**Лабораторная работа №3**

**Абстрактные классы, интерфейсы, framework Vaadin**

Реализовать абстрактный базовый класс, определив в нем абстрактные функции. Эти функции определяются в производных классах. В базовых классах должны быть объявлены абстрактные функции ввода/вывода, которые реализуются в производных классах. Вызывающая программа должна продемонстрировать вызов всех абстрактных функций классов-наследников с помощью указателя на базовый класс.

Изменить программу, используя в ней в качестве базового класса соответствующий интерфейс.

Примечание: при необходимости, в программах использовать функции класса **Math**, находящегося в пакете **java.lang**.

В качестве интерфейса для ввода исходных данных и вывода результатов использовать web-интерфейс организованный с помощью Vaadin, можно, например, использовать компоненты: **TextField**, **Label**, **Button** и другие.

**Вариант 9:** cоздать абстрактный базовый класс **Array** с виртуальными методами **sort** и поэлементной обработки массива **foreach**. Разработать производные классы **Bubble** (пузырек) и **Selection** (выбор). В первом классе сортировка реализуется методом пузырька, а поэлементная обработка состоит в извлечении квадратного корня. Во втором классе сортировка реализуется методом выбора, а поэлементная обработка – вычисление логарифма.

**Теоретическая часть**

**Абстрактный класс** – это такой класс, который не может быть создан напрямую (нельзя создать объект этого класса), а служит для определения общего интерфейса и базовой функциональности для своих потомков. В абстрактном классе обязательно есть хотя бы один абстрактный метод. Объявляется ключевым словом **abstract**. Абстрактный класс может содержать как абстрактные методы (без реализации), так и методы с полной реализацией. Подклассы обязаны реализовать все абстрактные методы или тоже объявляться как абстрактные.

**Абстрактная функция** – это метод, который объявлен без реализации (без тела) в абстрактном классе или интерфейсе. Объявляется ключевым словом **abstract**. Такой метод не содержит тела и должен быть реализован в подклассах.

**Интерфейс** – это специальный класс, который описывает только набор методов, которые классы-наследники должны реализовывать, но без самой реализации. Объявляется ключевым словом **interface**. Классы, реализующие интерфейс, обязаны реализовать все его методы.

**Vaadin** – это фреймворк для разработки web-приложений на Java с удобным UI, который позволяет создавать современные интерфейсы без глубоких знаний фронтенда (HTML, JS, CSS).

**Описание методов сортировки**

Сортировка пузырьком – это простой алгоритм сортировки, который многократно проходит по списку, сравнивает соседние элементы и меняет их местами, если они стоят в неправильном порядке. Алгоритм повторяет проходы по списку до тех пор, пока не останется ни одной пары, которую нужно поменять местами.

Принцип работы:

1. Сравниваются два соседних элемента
2. Если левый элемент больше правого – меняются местами
3. Проход повторяется от начала до конца массива, но каждый раз на один элемент меньше (самый большой элемент «всплывает» в конец)
4. Повторяется до тех пор, пока весь массив не отсортирован

Характеристики:

* Сложность в худшем случае: O(n²) – когда массив изначально отсортирован в обратном порядке
* Сложность в среднем: O(n²)
* Сложность в лучшем случае: O(n) – если массив уже отсортирован

Сортировка выбором – это простой алгоритм сортировки, который делит массив на две части: отсортированную и неотсортированную. На каждом шаге он выбирает минимальный (или максимальный) элемент из неотсортированной части и ставит его в конец отсортированной.

Принцип работы:

1. Ищется минимальный элемент в неотсортированной части массива
2. Этот минимальный элемент меняется местами с первым элементом неотсортированной части
3. Граница отсортированной части сдвигается на один элемент вправо
4. Повторяется, пока весь массив не станет отсортированным

Характеристики:

* Сложность в худшем, лучшем и среднем случае: O(n²)
* Независимо от того, отсортирован массив или нет, поиск минимального элемента требует обхода почти всех оставшихся элементов

**Текст программы**

1. Через абстрактный класс

**Array.java**:

package com.northpole.snow.logic;  
  
public abstract class Array {  
 protected double[] data;  
  
 public abstract void input(String input); *// ввод данных* public abstract String output(); *// вывод данных* public abstract void sort(); *// сортировка* public abstract void foreach(); *// применение операции*}

**Bubble.java**:

package com.northpole.snow.logic;  
  
public class Bubble extends Array {  
  
 *// Реализация метода ввода данных* @Override  
 public void input(String input) {  
 *// Разбиваем входную строку по пробелам (с учетом нескольких пробелов)* String[] tokens = input.trim().split("\\s+");  
 *// Создаем массив чисел нужного размера* data = new double[tokens.length];  
 *// Преобразуем каждую строку в число и сохраняем в массив* for (int i = 0; i < tokens.length; i++) {  
 data[i] = Double.*parseDouble*(tokens[i]);  
 }  
 }  
  
 *// Реализация метода вывода данных* @Override  
 public String output() {  
 *// Используем StringBuilder для эффективной конкатенации строк* StringBuilder sb = new StringBuilder();  
 *// Форматируем каждое число с 3 знаками после запятой* for (double d : data) {  
 sb.append(String.*format*("%.3f ", d));  
 }  
 *// Возвращаем результат, убирая последний пробел* return sb.toString().trim();  
 }  
  
 *// Реализация алгоритма пузырьковой сортировки* @Override  
 public void sort() {  
 *// Внешний цикл - проходим по всем элементам* for (int i = 0; i < data.length - 1; i++) {  
 *// Внутренний цикл - сравниваем соседние элементы* for (int j = data.length - 1; j > i; j--) {  
 *// Если текущий элемент больше следующего - меняем их местами* if (data[j] < data[j - 1]) {  
 double tmp = data[j]; *// Временная переменная для обмена* data[j] = data[j - 1];*// Перемещаем меньший элемент влево* data[j - 1] = tmp; *// Перемещаем больший элемент вправо* }  
 }  
 }  
 }  
  
 *// Реализация операции для каждого элемента массива* @Override  
 public void foreach() {  
 *// Применяем квадратный корень к каждому элементу массива* for (int i = 0; i < data.length; i++) {  
 data[i] = Math.*sqrt*(data[i]); *// Math.sqrt - стандартная функция Java* }  
 }  
}

**Selection.java**:

package com.northpole.snow.logic;  
  
public class Selection extends Array {  
  
 *// Реализация метода ввода данных* @Override  
 public void input(String input) {  
 *// Удаляем лишние пробелы и разбиваем строку на отдельные числа* String[] tokens = input.trim().split("\\s+");  
 *// Создаем массив для хранения чисел* data = new double[tokens.length];  
 *// Преобразуем каждую подстроку в число типа double* for (int i = 0; i < tokens.length; i++) {  
 data[i] = Double.*parseDouble*(tokens[i]);  
 }  
 }  
  
 *// Реализация метода вывода данных* @Override  
 public String output() {  
 *// Используем StringBuilder для эффективного построения строки* StringBuilder sb = new StringBuilder();  
 *// Форматируем каждое число с тремя знаками после запятой* for (double d : data) {  
 sb.append(String.*format*("%.3f ", d));  
 }  
 *// Возвращаем результат, обрезая последний пробел* return sb.toString().trim();  
 }  
  
 *// Реализация алгоритма сортировки выбором* @Override  
 public void sort() {  
 *// Проходим по всем элементам массива, кроме последнего* for (int i = 0; i < data.length - 1; i++) {  
 *// Предполагаем, что минимальный элемент находится на текущей позиции* int minIdx = i;  
 *// Ищем минимальный элемент в оставшейся части массива* for (int j = i + 1; j < data.length; j++) {  
 if (data[j] < data[minIdx]) {  
 minIdx = j; *// Нашли новый минимальный элемент* }  
 }  
 *// Обмен местами текущего элемента с найденным минимальным* double tmp = data[i];  
 data[i] = data[minIdx];  
 data[minIdx] = tmp;  
 }  
 }  
  
 *// Реализация операции для каждого элемента массива* @Override  
 public void foreach() {  
 *// Применяем натуральный логарифм (Math.log) к каждому элементу массива* for (int i = 0; i < data.length; i++) {  
 data[i] = Math.*log*(data[i]);  
 }  
 }  
}

1. Через интерфейс

**Array.java**:

package com.northpole.snow.logic;  
  
public interface Array {  
 void input(String input); *// ввод данных* String output(); *// вывод данных* void sort(); *// сортировка* void foreach(); *// применение операции*}

**Bubble.java**:

package com.northpole.snow.logic;  
  
public class Bubble implements Array {  
 private double[] data; *// Массив для хранения чисел  
  
 // Преобразует строку чисел в массив double* @Override  
 public void input(String input) {  
 *// Разбиваем строку по пробелам и удаляем пустые элементы* String[] tokens = input.trim().split("\\s+");  
 data = new double[tokens.length];  
  
 *// Парсим каждую подстроку в число* for (int i = 0; i < tokens.length; i++) {  
 data[i] = Double.*parseDouble*(tokens[i]);  
 }  
 }  
  
 *// Форматирует массив в строку с 3 знаками после запятой* @Override  
 public String output() {  
 StringBuilder sb = new StringBuilder();  
  
 *// Добавляем каждое число с форматированием* for (double d : data) {  
 sb.append(String.*format*("%.3f ", d));  
 }  
  
 *// Удаляем последний пробел и возвращаем результат* return sb.toString().trim();  
 }  
  
 *// Реализация алгоритма пузырьковой сортировки* @Override  
 public void sort() {  
 *// Внешний цикл - проходим по всем элементам* for (int i = 0; i < data.length - 1; i++) {  
 *// Внутренний цикл - сравниваем соседние элементы* for (int j = data.length - 1; j > i; j--) {  
 *// Если текущий элемент больше следующего - меняем их местами* if (data[j] < data[j - 1]) {  
 double tmp = data[j]; *// Временная переменная для обмена* data[j] = data[j - 1];*// Перемещаем меньший элемент влево* data[j - 1] = tmp; *// Перемещаем больший элемент вправо* }  
 }  
 }  
 }  
  
 *// Применяет квадратный корень к каждому элементу* @Override  
 public void foreach() {  
 *// Обрабатываем каждый элемент массива* for (int i = 0; i < data.length; i++) {  
 data[i] = Math.*sqrt*(data[i]);  
 }  
 }  
}

**Selection.java**:

package com.northpole.snow.logic;  
  
public class Selection implements Array {  
 private double[] data; *// Внутренний массив для хранения данных  
  
 // Преобразует строку чисел в массив double* @Override  
 public void input(String input) {  
 *// Удаляем пробелы по краям и разбиваем строку по пробелам* String[] tokens = input.trim().split("\\s+");  
 *// Создаем массив нужного размера* data = new double[tokens.length];  
 *// Конвертируем каждую строку в число* for (int i = 0; i < tokens.length; i++) {  
 data[i] = Double.*parseDouble*(tokens[i]);  
 }  
 }  
  
 *// Возвращает отформатированную строку элементов массива* @Override  
 public String output() {  
 StringBuilder sb = new StringBuilder();  
 *// Форматируем каждое число с 3 знаками после запятой* for (double d : data) {  
 sb.append(String.*format*("%.3f ", d));  
 }  
 *// Удаляем последний пробел и возвращаем* return sb.toString().trim();  
 }  
  
 *// Реализация сортировки выбором* @Override  
 public void sort() {  
 *// Проходим по всем элементам кроме последнего* for (int i = 0; i < data.length - 1; i++) {  
 int minIdx = i; *// Индекс минимального элемента  
 // Ищем минимальный элемент в оставшейся части* for (int j = i + 1; j < data.length; j++) {  
 if (data[j] < data[minIdx]) {  
 minIdx = j;  
 }  
 }  
 *// Меняем местами текущий и минимальный элементы* double tmp = data[i];  
 data[i] = data[minIdx];  
 data[minIdx] = tmp;  
 }  
 }  
  
 *// Применяет натуральный логарифм к каждому элементу* @Override  
 public void foreach() {  
 *// Обрабатываем каждый элемент массива* for (int i = 0; i < data.length; i++) {  
 data[i] = Math.*log*(data[i]); *// Math.log - натуральный логарифм* }  
 }  
}

**MainView.java** (одинаковый для абстрактного класса и интерфейса):

package com.northpole.snow;  
  
import com.northpole.snow.logic.Array;  
import com.northpole.snow.logic.Bubble;  
import com.northpole.snow.logic.Selection;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.html.H2;  
import com.vaadin.flow.component.html.Label;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.Route;  
  
@Route("")  
public class MainView extends VerticalLayout {  
  
 *// Конструктор класса MainView* public MainView() {  
 *// Создание заголовка страницы* H2 title = new H2("Сортировка и обработка массива");  
  
 *// Создание текстового поля для ввода чисел* TextField inputField = new TextField("Введите числа через пробел");  
 inputField.setWidth("500px"); *// Установка ширины поля ввода  
  
 // Создание кнопок для разных видов сортировки* Button bubbleBtn = new Button("Bubble: sort + sqrt");  
 Button selectionBtn = new Button("Selection: sort + log");  
  
 *// Создание метки для вывода результата* Label sortedLabel = new Label();  
 Label processedLabel = new Label();  
  
 *// Обработчик клика для кнопки Bubble sort* bubbleBtn.addClickListener(e -> {  
 Array arr = new Bubble();  
 arr.input(inputField.getValue());  
 arr.sort();  
 sortedLabel.setText("Отсортированный массив (Bubble): " + arr.output());  
 arr.foreach();  
 processedLabel.setText("После sqrt: " + arr.output());  
 });  
  
 *// Обработчик клика для кнопки Selection sort* selectionBtn.addClickListener(e -> {  
 Array arr = new Selection();  
 arr.input(inputField.getValue());  
 arr.sort();  
 sortedLabel.setText("Отсортированный массив (Bubble): " + arr.output());  
 arr.foreach();  
 processedLabel.setText("После log: " + arr.output());  
 });  
  
 *// Добавляем все компоненты на страницу в вертикальном layout* add(title, inputField, bubbleBtn, selectionBtn, sortedLabel, processedLabel);  
 }  
}

**Описание программы**

Класс **Array** – абстрактный класс/интерфейс:

* В реализации через абстрактный класс имеет поле **data** – общий массив для хранения чисел. В реализации через интерфейс данное поле отсутствует
* **input()** – принимает строку с числами, абстрактный/не реализован
* **output()** – возвращает строку с элементами массива, абстрактный/не реализован
* **sort()** – сортирует массив, абстрактный/не реализован
* **foreach()** – применяет к каждому элементу массива определённую функцию, абстрактный/не реализован

Класс **Bubble** – наследует/реализует **Array**:

* В реализации через интерфейс имеет поле **data** – общий массив для хранения чисел. В реализации через абстрактный класс данное поле отсутствует
* **input()** – разбивает входную строку по пробелам. Преобразует каждое число в **double** и сохраняет в массив **data**
* **output()** – формируется строка из элементов массива
* **sort()** – реализована пузырьковая сортировка массива **data**
* **foreach()** – к каждому элементу массива применяется функция квадратного корня **Math.sqrt**

Класс **Selection** – наследует/реализует **Array**:

* В реализации через интерфейс имеет поле **data** – общий массив для хранения чисел. В реализации через абстрактный класс данное поле отсутствует
* **input()** – разбивает входную строку по пробелам. Преобразует каждое число в **double** и сохраняет в массив **data**
* **output()** – формируется строка из элементов массива
* **sort()** – реализована сортировка выбором массива **data**
* **foreach()** – к каждому элементу массива применяется натуральный логарифм **Math.log**

Класс **MainView** – Vaadin UI:

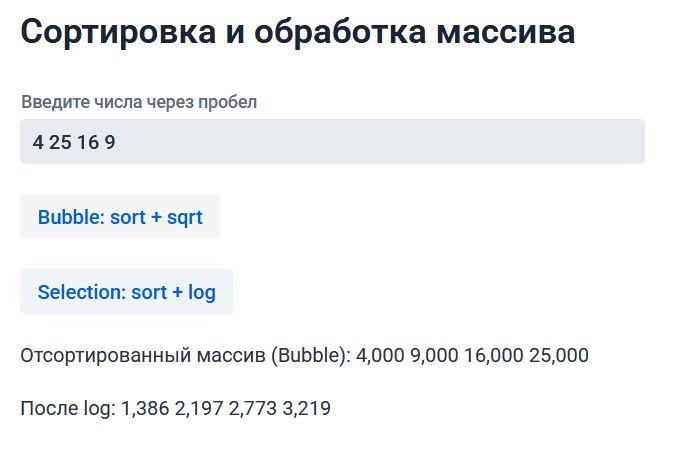
* Создаются UI-компоненты: заголовок, поле ввода, две кнопки, метка для вывода
* При нажатии на кнопку **bubbleBtn**:
  + Создаётся объект **Bubble**
  + Вызываются методы: **input** → **sort** → **foreach** → **output**
  + Результат выводится в **outputLabel**.
* При нажатии на кнопку **selectionBtn**:
  + Аналогично, создаётся объект **Selection**
  + Вызываются те же методы, но с логикой сортировки выбором и логарифмом
  + Результат отображается

**Результаты тестов**

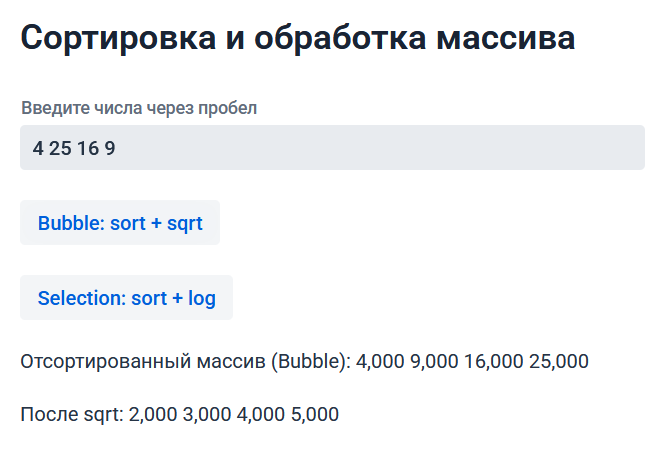
Через абстрактный класс пузырьком:



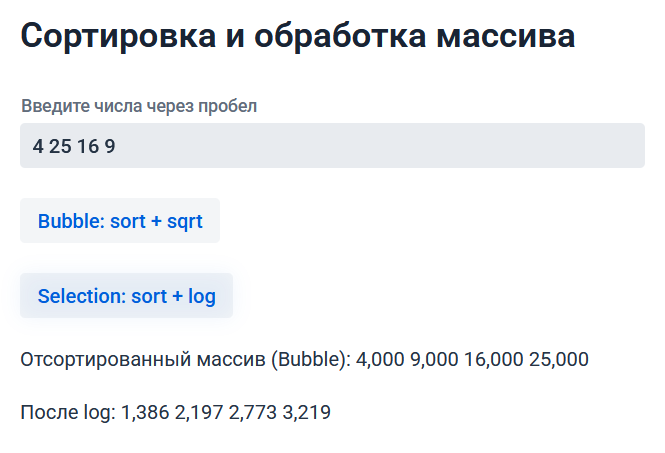
Через абстрактный класс выбором:



Через интерфейс пузырьком:



Через интерфейс выбором:



**Вывод**

В ходе выполнения лабораторной работы были созданы абстрактный класс, интерфейс и классы-потомки. В качестве абстрактного класса/интерфейса был реализован массив, а в качестве его потомков классы для сортировки методом пузырька и выбора. Интерфейс программы выполнен через фреймворк Vaadin.