

**МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**

(национальный исследовательский университет)»

Кафедра 319 «Системы интеллектуального мониторинга»

**ЛАБОРАТОРНАЯ РАБОТА №3**

по дисциплине «Программирование»

Тема: динамические структуры данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | | |  | Серебренников А.А. |
| Группа | | | М3О-135Б-20 |  |
| Проверил | | |  | Шилов В.В. |
| Оценка |  | Дата защиты « \_ » 2021 г. | | |

**Москва 2021**

**МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**

(национальный исследовательский университет)»

Кафедра 319 «Системы интеллектуального мониторинга»

**З А Д А Н И Е**

на лабораторную работу №3 по дисциплине

«Программирование»

Студент М3О-135Б-20 Серебренников Андрей Алексеевич

(№ группы, Ф. И. О.)

Тема \_\_\_\_\_\_\_\_\_\_\_\_динамические структуры данных\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перечень вопросов, подлежащих разработке в лабораторной работе

Изучение принципов работы динамических структур данных\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Реализация предложенных интерфейсов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Разработка пояснительной записки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рекомендуемая литература

1. Герберт Шилдт. Java. Полное руководство, 10-е изд. : Пер. с англ. – СПб. ООО "Альфакнига', 2018. – 1488 с. ISBN 978-5-6040043-6-4 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Бертран Мейер. Объектно-ориентированное конструирование программных системю: Русская Редакция; 2015. – 768 с. ISBN 5-7502-0255-0\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Лафоре Р. Структуры данных и алгоритмы в Java. Классика Computers Science. 2-е изд. – СПб.:Питер, 2013. – 704 с. ISBN 978-5-496-007405 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Задание выдано «17» апрель 2021 г.

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*Шилов В.В* \_\_\_\_

(Ф. И. О., подпись)

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc73747997)

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc73747998)

[Array 5](#_Toc73747999)

[Linked 5](#_Toc73748000)

[Map 5](#_Toc73748001)

[2. ПРОЦЕСС РАЗРАБОТКИ 6](#_Toc73748002)

[2.1. Ключевые части кода 6](#_Toc73748003)

[2.2. Диаграмма классов 14](#_Toc73748004)

[3. ПРОЦЕСС ТЕСТИРОВАНИЯ 17](#_Toc73748005)

[ЗАКЛЮЧЕНИЕ 19](#_Toc73748006)

[СПИСОК ЛИТЕРАТУРЫ 20](#_Toc73748007)

[ПРИЛОЖЕНИЕ А 21](#_Toc73748008)

[ПРИЛОЖЕНИЕ Б 26](#_Toc73748009)

[ПРИЛОЖЕНИЕ В 32](#_Toc73748010)

[ПРИЛОЖЕНИЕ Г 39](#_Toc73748011)

# ВВЕДЕНИЕ

Задача заключается в написании собственных классов Array, Linked, Map, которые являются динамическими структурами данных. Задача выполнена на языке программирования Java. Каждый из классов содержит свои методы, соответствующие тем методам, которые заложены в самом языке программирования. Каждый метод использует динамическое выделение памяти. Написанные классы необходимо реализовать в задаче, написанной в методе Main.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Array – автоматически расширяемый массив. Array может менять свой размер во время исполнения программы, при этом не обязательно указывать размерность при создании объекта. Есть возможность вставить элемент в произвольное место коллекции. А также спокойно удалить элемент из любого места. Элементы Array могут быть абсолютно любых типов в том числе и null[[1]](#java).

Преимущество: быстрая вставка в конец массива, быстрый поиск по индексу.  
Недостатки: копирование из старого массива в новый, дорогая операция вставки не в конец массива, дорогая операция удаления не из конца списка.

Linked – односвязный список, где каждый элемент структуры содержит указатели на следующий элемент. Linked реализует методы получения, удаления и вставки в начало, середину и конец списка, а также позволяет добавлять любые элементы, в том числе и null[[1]](#java).

Преимущество: быстрая вставка в конец массива и в начало массива  
Недостатки: дорогая операция поиска по индексу.

Map – структура в контейнерах, которой хранятся два объекта: ключ и связанное с ним значение. Map позволяет искать объекты по ключу. Объект, ассоциированный с ключом, называется значением. И ключи, и значения являются объектами. Ключи могут быть уникальными, а значения могут дублироваться. Некоторые отображения допускают пустые ключи и пустые значения[[1]](#java).

# ПРОЦЕСС РАЗРАБОТКИ

## 2.1. Ключевые части кода

Задача №1 – Написать собственный класс, основанный на интерфейсе Array. Основной код программы представлен в приложении А. Диаграмма для класса Array представлена на рисунке 2.1. Ключевые части кода задачи №1:

Добавление нового элемента в Array представлено в листинге 1

Листинг 1 – Добавление нового элемента

public boolean add(String o) {

if (arraySize > elementsNumber) {

array[elementsNumber] = o;

} else {

arraySize \*= 2;

array = Arrays.copyOf(array, arraySize);

array[elementsNumber] = o;

}

elementsNumber++;

return true;

}

Добавление нового элемента в указанное место в Array представлено в листинге 2

Листинг 2 – Добавление нового элемента в указанное место

public void add(int index, String element) {

if (index > elementsNumber || index < 0) {

throw new IndexOutOfBoundsException("Invalid indexes");

} else if (index == elementsNumber) {

add(element);

} else {

if (elementsNumber == arraySize) {

array = Arrays.copyOf(array, arraySize \* 2);

arraySize \*= 2;

}

System.arraycopy(array, index, array, index + 1, elementsNumber - index);

array[index] = element;

++elementsNumber;

}

}

Удаление элемента по указанному индексу в Array представлено в листинге 3

Листинг 3 – удаление элемента по указанному индексу

public String remove(int index) {

if (index < 0 || index > (elementsNumber - 1)) {

throw new IndexOutOfBoundsException("Invalid indexes");

} else {

String buffer = array[index];

--elementsNumber;

System.arraycopy(array, index + 1, array, index, elementsNumber - index);

array[elementsNumber] = null;

return buffer;

}

}

Получение элемента по указанному индексу в Array представлено в листинге 4

Листинг 4 – Получение элемента по указанному индексу

public String get(int index) {

if (index < 0 || (index > elementsNumber - 1)) {

throw new IndexOutOfBoundsException("Invalid indexes");

} else {

return array[index];

}

}

Замена элемента по указанному индексу в Array представлено в листинге 5

Листинг 5 – Замена элемента по указанному индексу

public String set(int index, String element) {

if (index < 0 || index > (elementsNumber - 1)) {

throw new IndexOutOfBoundsException("Invalid indexes");

} else {

String buffer = array[index];

array[index] = element;

return buffer;

}

}

Задача №2 – Написать собственный класс, основанный на интерфейсе Linked. Основной код программы представлен в приложении Б. Диаграмма класса Linked представлена на рисунке 2.2. Ключевые части кода задачи №2:

Добавление нового элемента в Linked представлено в листинге 6

Листинг 6 – Добавление нового элемента

public boolean add(String o) {

Node currentNode = new Node();

currentNode.value = o;

if (first == null) {

last = currentNode;

first = currentNode;

} else {

last.next = currentNode;

last = currentNode;

}

++elementsNumber;

return true;

}

Добавление нового элемента в указанное место в Linked представлено в листинге 7

Листинг 7 – Добавление нового элемента по индексу

public void add(int index, String element) {

checkAddIndex(index);

if (index == elementsNumber) {

add(element);

} else if (index == 0) {

Node currentNode = new Node();

currentNode.value = element;

currentNode.next = first;

first = currentNode;

++elementsNumber;

} else {

Node currentNode = first;

int nodePos = 0;

while (currentNode != null) {

if (nodePos == index - 1) {

Node nextCurrentNode = currentNode.next;

Node newNode = new Node();

newNode.value = element;

newNode.next = nextCurrentNode;

currentNode.next = newNode;

++elementsNumber;

}

nodePos++;

currentNode = currentNode.next;

}

}

}

Удаление элемента по указанному индексу в Linked представлено в листинге 8

Листинг 8 – удаление элемента по индексу

public String remove(int index) {

checkIndex(index);

Node currentNode = first;

Node preNode = null;

if (index == 0) {

String buffer = first.value;

first = first.next;

--elementsNumber;

if (size() == 0) {

last = null;

}

return buffer;

} else {

for (int i = 0; i < elementsNumber; ++i) {

if (i == index) {

--elementsNumber;

preNode.next = currentNode.next;

if (index == elementsNumber) {

last = preNode;

}

return currentNode.value;

}

preNode = currentNode;

currentNode = currentNode.next;

}

}

return null;

}

Получение элемента по указанному индексу в Linked представлено в листинге 9

Листинг 9 – Получение элемента по указанному индексу

public String get(int index) {

checkIndex(index);

Node currentNode = first;

for (int i = 0; i < index; ++i) {

if (i == index) {

return currentNode.value;

}

currentNode = currentNode.next;

}

if (currentNode.value != null) {

return currentNode.value;

}

return null;

}

Замена элемента по указанному индексу в Linked представлено в листинге 10

Листинг 10 – Замена элемента по указанному индексу

public String set(int index, String element) {

String buffer;

checkIndex(index);

Node currentNode = first;

int i = 0;

while (currentNode != null) {

if (i == index) {

buffer = currentNode.value;

currentNode.value = element;

return buffer;

}

currentNode = currentNode.next;

i++;

}

return null;

}

Задача №3 – Написать собственный класс, основанный на интерфейсе Map. Основной код программы представлен в приложении В. Диаграмма для класса Map представлена на рисунке 2.3. Ключевые части код задачи №3:

Вычисление hash с помощью функции hashCalc представлено в листинге 11

Листинг 11 – Вычисление hash

public int hashCalc(String key) {

int hashCode = 0;

if (key != null) {

for (int i = 0; i < key.length(); ++i) {

hashCode += (int) (key.charAt(i));

}

}

return hashCode;

}

Добавление нового элемента в Map по ключу и значению представлено в листинге 12

Листинг 12 – Добавление нового элемента по ключу и значению

public Integer put(String key, Integer value) {

int hash = hashCalc(key);

if (nodeBasket[hash % basketNumbers] != null) {

Node currentNode = nodeBasket[hash % basketNumbers];

do {

if (Objects.equals(currentNode.key, key)) {

int buffer = currentNode.value;

currentNode.value = value;

return buffer;

}

if (currentNode.next == null) {

Node nextNode = new Node(hash, key, value);

currentNode.next = nextNode;

elementsNumber++;

break;

}

currentNode = currentNode.next;

} while (currentNode != null);

} else {

Node newNode = new Node(hash, key, value);

newNode.next = null;

elementsNumber++;

nodeBasket[hash % basketNumbers] = newNode;

}

if (elementsNumber >= basketNumbers \* loadFactor) {

nodeBasket = resize();

}

return null;

}

Удаление элемента в Map по ключу представлено в листинге 13

Листинг 13 – Удаление элемента по ключу

public Integer remove(String key) {

int hash = hashCalc(key);

if (nodeBasket[hash % basketNumbers] != null) {

Node currentNode = nodeBasket[hash % nodeBasket.length];

Node bufferNode = null;

while (currentNode != null) {

if (Objects.equals(currentNode.key, key)) {

Integer returnInt = currentNode.value;

if (bufferNode == null) {

nodeBasket[hash % basketNumbers] = currentNode.next;

} else {

bufferNode.next = currentNode.next;

}

--elementsNumber;

return returnInt;

}

bufferNode = currentNode;

currentNode = currentNode.next;

}

}

return null;

}

Получение значения по ключу в Map представлено в листинге 14

Листинг 14 – Получение значения по ключу

public Integer get(String key) {

int hash = hashCalc(key);

if (nodeBasket[hash % nodeBasket.length] == null) {

return null;

}

Node currentNode = nodeBasket[hash % nodeBasket.length];

while (currentNode != null) {

if (Objects.equals(currentNode.key, key)) {

return currentNode.value;

}

currentNode = currentNode.next;

}

return null;

}

Задача №4 – «Написать класс, который позволяет через командную строчку создавать на выбор динамическую структуру и заполнять ее. Вывести количество элементов в структуре и спросить у пользователя хочется ли он вывести все элементы. YES - означает, вывести, NO - не выводить и перейти к следующему шагу. Если пользователь ввел команду YES, то выводится список элементов, каждый элемент в новой строке. Спросить у пользователя, хочется ли он получить конкретный элемент по индексу или по ключу. YES означает, что хочет. NO, что не хочет. Если пользователь ввел YES, то необходимо дождаться, что он введет индекс или ключ (в зависимости от типа данных для map это ключ, для остальных - индекс). Далее вывести полученных элемент, а также необходимо количество миллисекунд для этого действия.» Основной код программы представлен в приложении Г. Ключевые части кода программы №4:

Выбор коллекции для добавления элементов представлен в листинге 15

Листинг 15 – Выбор коллекции для добавления

do {

selectCollection = input.nextLine();

check = true;

if (selectCollection.matches("^ARRAY$|^LINKED$|^MAP$")) {

check = false;

} else {

System.out.println("Incorrect input");

}

} while (check);

ArrayMethods array = new ArrayMethods();

LinkedMethods linked = new LinkedMethods();

MapMethods map = new MapMethods();

String inputInColletction;

do {

inputInColletction = input.nextLine();

check = true;

if (inputInColletction.equals("END")) {

check = false;

break;

}

switch (selectCollection) {

case arraySelect:

array.add(inputInColletction);

break;

case linkedSelect:

linked.add(inputInColletction);

break;

case mapSelect:

boolean secCheck;

int value = EMPTY;

do {

System.out.print("input value: ");

if (input.hasNextInt()) {

value = input.nextInt();

secCheck = true;

if (value <= 0) {

System.out.println("the value must be greater than 0");

secCheck = false;

}

} else {

input.nextLine();

System.out.println("You input not an integer value.");

secCheck = false;

}

} while (!secCheck);

input.nextLine();

map.put(inputInColletction, value);

break;

}

} while (check);

## 2.2. Диаграмма классов

1. Диаграмма класса Array (Рис 2.1):

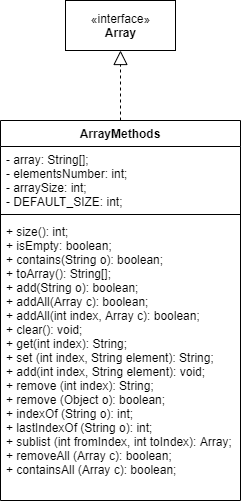


Рис 2.1 – Диаграмма класса Array

1. Диаграмма класса Linked (Рис 2.2):

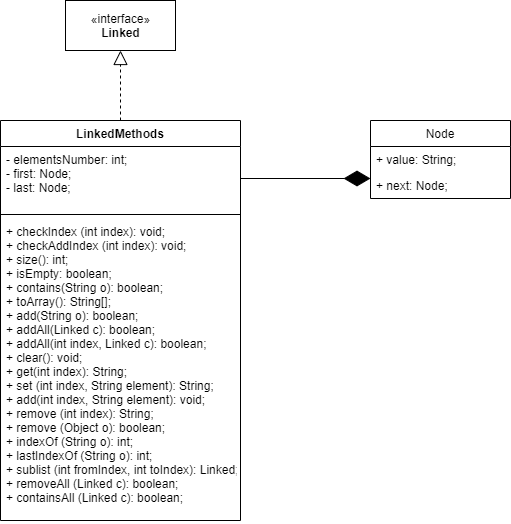


Рис 2.2 – Диаграмма класса Linked

1. Диаграмма класса Map (Рис 2.3)

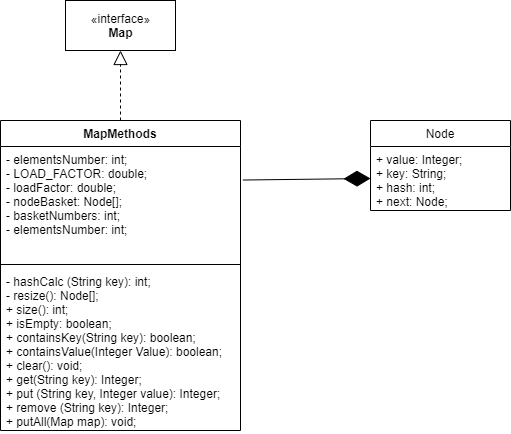


Рис 2.3 – Диаграмма класса Map

# ПРОЦЕСС ТЕСТИРОВАНИЯ

Тестирование программы, написанной в классе Main представлено в таблице 1

Таблица 1 – Тестирование программы, написанной в классе Main

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назначение теста | Входные данные | Реакция программы |
| 1 | Проверка правильности выбора коллекции(1) | 123  Маша  Lin  Map  Array | «Incorrect input» |
| 2 | Проверка правильности выбора коллекции(2) | LINKED  ARRAY  MAP | Переход к добавлению в выбранную коллекцию |
| 3 | Проверка ввода элементов в выбранную коллекцию | Маша  Паша  Дима  END | Запись элементов «Маша», «Паша», «Дима» в выбранную коллекцию (кроме MAP) |
| 4 | Проверка ввода элементов в MAP(1) | Саша  Вика | «You input not an integer value.» |
| 5 | Проверка ввода элементов в MAP(2) | Дима  -1 | «the value must be greater than 0» |
| 6 | Проверка ввода элементов в MAP(3) | Дима  15  END | Создание пары key – «Дима», value – «15». |
| 7 | Проверка вывода всех элементов массива(1) | YES | Вывод элементов «Маша», «Паша», «Дима»(кроме MAP) |
| 8 | Проверка вывода всех элементов массива для MAP | YES | Вывод элементов MAP в формате «Ключ –> Значение» - «Дима -> 15» |
| 9 | Проверка вывода всех элементов массива | NO | Переход к выводу элемента по индексу |
| 10 | Получение значения по индексу(1) | 1 | Вывод «Миша» (для LINKED и ARRAY). Вывод «Дима -> 15» (для MAP) |
| 11 | Получение значения по индексу(2) | NO | Завершение работы программы. |
| 12 | Получение значения по индексу(3) | 10 | Вывод null. |
| 13 | Получение значения по индексу(4) | Паша  ^  ? | «Not an integer.» |
| 14 | Получение значения по индексу(5) | -2 | "Enter index > 0." |

# ЗАКЛЮЧЕНИЕ

Создание собственных динамических структур данных позволяет понять принцип реализации этих структур в стандартных библиотеках Java. Это необходимо для последующего использованию этих классов в своих программах.

# СПИСОК ЛИТЕРАТУРЫ

1. Герберт Шилдт. Java. Полное руководство, 10-е изд. : Пер. с англ. – СПб. ООО "Альфакнига', 2018. – 1488 с. ISBN 978-5-6040043-6-4

# ПРИЛОЖЕНИЕ А

Листинг 15 – Код программы класса ArrayMethods:

package mai;

import mai.interfaces.Array;

import java.util.Arrays;

import java.util.Objects;

public class ArrayMethods implements Array {

String[] array;

int arraySize;

int elementsNumber = 0;

private static final int DEFAULT\_SIZE = 5;

public ArrayMethods() {

this(DEFAULT\_SIZE);

}

public ArrayMethods(int size) {

if (size > 0) {

array = new String[size];

arraySize = size;

} else {

throw new IllegalArgumentException("Size must be greater than 0" + size);

}

}

public int size() {

return elementsNumber;

}

public boolean isEmpty() {

return elementsNumber == 0;

}

public boolean contains(String o) {

return indexOf(o) != -1;

}

public String[] toArray() {

return Arrays.copyOf(array, elementsNumber);

}

public boolean add(String o) {

if (arraySize > elementsNumber) {

array[elementsNumber] = o;

} else {

arraySize \*= 2;

array = Arrays.copyOf(array, arraySize);

array[elementsNumber] = o;

}

elementsNumber++;

return true;

}

public boolean addAll(Array c) {

if (c == null) {

throw new IllegalArgumentException("Array can not be null");

} else if (c.size() == 0) {

return false;

} else {

for (int i = 0; i < c.size(); ++i) {

add(c.get(i));

}

return true;

}

}

public boolean addAll(int index, Array c) {

int prevSize = elementsNumber;

if (c == null) {

throw new IllegalArgumentException("Array can not be null");

}

if (index < 0 || index > elementsNumber) {

throw new IndexOutOfBoundsException("Invalid indexes");

} else {

if (c.size() == 0) {

return false;

}

for (int i = 0; i < c.size(); ++i) {

add(index + i, c.get(i));

}

}

return elementsNumber != prevSize;

}

public void clear() {

array = new String[DEFAULT\_SIZE];

arraySize = DEFAULT\_SIZE;

elementsNumber = 0;

}

public String get(int index) {

if (index < 0 || (index > elementsNumber - 1)) {

throw new IndexOutOfBoundsException("Invalid indexes");

} else {

return array[index];

}

}

public String set(int index, String element) {

if (index < 0 || index > (elementsNumber - 1)) {

throw new IndexOutOfBoundsException("Invalid indexes");

} else {

String buffer = array[index];

array[index] = element;

return buffer;

}

}

public void add(int index, String element) {

if (index > elementsNumber || index < 0) {

throw new IndexOutOfBoundsException("Invalid indexes");

} else if (index == elementsNumber) {

add(element);

} else {

if (elementsNumber == arraySize) {

array = Arrays.copyOf(array, arraySize \* 2);

arraySize \*= 2;

}

System.arraycopy(array, index, array, index + 1, elementsNumber - index);

array[index] = element;

++elementsNumber;

}

}

public String remove(int index) {

if (index < 0 || index > (elementsNumber - 1)) {

throw new IndexOutOfBoundsException("Invalid indexes");

} else {

String buffer = array[index];

--elementsNumber;

System.arraycopy(array, index + 1, array, index, elementsNumber - index);

array[elementsNumber] = null;

return buffer;

}

}

public boolean remove(Object o) {

for (int i = 0; i < elementsNumber; ++i) {

if (Objects.equals(o, array[i])) {

remove(i);

return true;

}

}

return false;

}

public int indexOf(String o) {

for (int i = 0; i < elementsNumber; ++i) {

if (Objects.equals(o, array[i])) {

return i;

}

}

return -1;

}

public int lastIndexOf(String o) {

for (int i = elementsNumber - 1; i >= 0; --i) {

if (Objects.equals(o, array[i])) {

return i;

}

}

return -1;

}

public Array subList(int fromIndex, int toIndex) {

if (toIndex > (elementsNumber) || fromIndex < 0 || fromIndex >= toIndex || fromIndex >= elementsNumber) {

throw new IndexOutOfBoundsException("Invalid indexes");

} else {

Array newArray = new ArrayMethods(toIndex - fromIndex);

for (int i = fromIndex; i < toIndex; ++i) {

newArray.add(array[i]);

}

return newArray;

}

}

public boolean removeAll(Array c) {

if (c == null) {

throw new IllegalArgumentException("Array can not be null");

}

int prevSize = elementsNumber;

if (c.size() <= 0) {

return false;

} else {

for (int i = 0; i < c.size(); ++i) {

remove(c.get(i));

}

}

return elementsNumber != prevSize;

}

public boolean containsAll(Array c) {

if (c == null) {

throw new IllegalArgumentException("Array can not be null");

}

int check = 0;

for (int i = 0; i < c.size(); ++i) {

for (int j = 0; j < elementsNumber; ++j) {

if (Objects.equals(c.get(i), array[j])) {

++check;

}

}

}

return check == c.size();

}

}

# ПРИЛОЖЕНИЕ Б

Листинг 16 – Код программы класса LinkedMethods:

package mai;

import mai.interfaces.Linked;

import java.util.Objects;

public class LinkedMethods implements Linked {

int elementsNumber = 0;

public static class Node {

String value;

Node next;

}

Node first = null;

Node last = null;

public LinkedMethods() {

}

public void checkIndex(int index) {

if (index >= elementsNumber || index < 0) {

throw new IllegalArgumentException("Invalid indexes");

}

}

public void checkAddIndex(int index) {

if (index > elementsNumber || index < 0) {

throw new IllegalArgumentException("Invalid indexes");

}

}

public int size() {

return elementsNumber;

}

public boolean isEmpty() {

return elementsNumber == 0;

}

public boolean contains(String o) {

return indexOf(o) != -1;

}

public String[] toArray() {

String[] stringForReturn = new String[elementsNumber];

if (first != null) {

Node currentNode = first;

for (int i = 0; i < elementsNumber

&& currentNode.next != null; i++, currentNode = currentNode.next) {

stringForReturn[i] = currentNode.value;

}

stringForReturn[elementsNumber - 1] = currentNode.value;

}

return stringForReturn;

}

public boolean add(String o) {

Node currentNode = new Node();

currentNode.value = o;

if (first == null) {

last = currentNode;

first = currentNode;

} else {

last.next = currentNode;

last = currentNode;

}

++elementsNumber;

return true;

}

public boolean addAll(Linked c) {

if (c == null) {

throw new IllegalArgumentException("Linked can not be null");

} else if (c.size() <= 0) {

return false;

}

addAll(elementsNumber, c);

return true;

}

public boolean addAll(int index, Linked c) {

checkAddIndex(index);

if (c == null) {

throw new IllegalArgumentException("Linked can not be null");

} else {

if (c.size() == 0) {

return false;

}

for (int i = index; i < c.size() + index; ++i) {

add(i, c.get(i - index));

}

return true;

}

}

public void clear() {

first = null;

last = null;

elementsNumber = 0;

}

public String get(int index) {

checkIndex(index);

Node currentNode = first;

for (int i = 0; i < index; ++i) {

if (i == index) {

return currentNode.value;

}

currentNode = currentNode.next;

}

if (currentNode.value != null) {

return currentNode.value;

}

return null;

}

public String set(int index, String element) {

String buffer;

checkIndex(index);

Node currentNode = first;

int i = 0;

while (currentNode != null) {

if (i == index) {

buffer = currentNode.value;

currentNode.value = element;

return buffer;

}

currentNode = currentNode.next;

i++;

}

return null;

}

public void add(int index, String element) {

checkAddIndex(index);

if (index == elementsNumber) {

add(element);

} else if (index == 0) {

Node currentNode = new Node();

currentNode.value = element;

currentNode.next = first;

first = currentNode;

++elementsNumber;

} else {

Node currentNode = first;

int nodePos = 0;

while (currentNode != null) {

if (nodePos == index - 1) {

Node nextCurrentNode = currentNode.next;

Node newNode = new Node();

newNode.value = element;

newNode.next = nextCurrentNode;

currentNode.next = newNode;

++elementsNumber;

}

nodePos++;

currentNode = currentNode.next;

}

}

}

public String remove(int index) {

checkIndex(index);

Node currentNode = first;

Node preNode = null;

if (index == 0) {

String buffer = first.value;

first = first.next;

--elementsNumber;

if (size() == 0) {

last = null;

}

return buffer;

} else {

for (int i = 0; i < elementsNumber; ++i) {

if (i == index) {

--elementsNumber;

preNode.next = currentNode.next;

if (index == elementsNumber) {

last = preNode;

}

return currentNode.value;

}

preNode = currentNode;

currentNode = currentNode.next;

}

}

return null;

}

public boolean remove(Object o) {

Node currentNode;

int i = 0;

for (currentNode = first; (currentNode != null && i < elementsNumber); currentNode = currentNode.next) {

if (Objects.equals(get(i), o)) {

remove(i);

return true;

}

++i;

}

return false;

}

public int indexOf(String o) {

for (int i = 0; i < elementsNumber; ++i) {

if (Objects.equals(o, get(i))) {

return i;

}

}

return -1;

}

public int lastIndexOf(String o) {

for (int i = (elementsNumber - 1); i >= 0; --i) {

if (Objects.equals(o, get(i))) {

return i;

}

}

return -1;

}

public Linked subList(int fromIndex, int toIndex) {

checkIndex(fromIndex);

checkAddIndex(toIndex);

if (fromIndex >= toIndex) {

throw new IllegalArgumentException("invalid indexes");

}

Linked buffer = new LinkedMethods();

Node currentNode = first;

if (fromIndex == toIndex) {

return buffer;

} else {

for (int i = 0; i < toIndex && currentNode != null; ++i) {

if (i >= fromIndex) {

buffer.add(currentNode.value);

}

currentNode = currentNode.next;

}

}

return buffer;

}

public boolean removeAll(Linked c) {

int check = elementsNumber;

if (c == null) {

throw new IllegalArgumentException("Array can not be null");

} else if (c.size() <= 0) {

return false;

} else {

for (int i = 0; i < c.size(); ++i) {

remove(c.get(i));

}

}

return check != elementsNumber;

}

public boolean containsAll(Linked c) {

int check = 0;

if (c == null) {

throw new IllegalArgumentException("Array can not be null");

} else if (c.size() == 0) {

return false;

}

for (int i = 0; i < c.size(); ++i) {

if (contains(c.get(i))) {

++check;

}

}

return check == c.size();

}

}

# ПРИЛОЖЕНИЕ В

Листинг 17 – Код программы класса MapMethods:

package mai;

import mai.interfaces.Map;

import java.util.Objects;

public class MapMethods implements Map {

public static class Node {

String key;

Integer value;

int hash;

Node next;

public Node(int hash, String key, int value) {

this.hash = hash;

this.key = key;

this.value = value;

Node next;

}

}

private static final double LOAD\_FACTOR = 0.75;

private final double loadFactor;

private static final int DEFAULT\_SIZE = 5;

Node[] nodeBasket;

int basketNumbers = 0;

int elementsNumber = 0;

public int hashCalc(String key) {

int hashCode = 0;

if (key != null) {

for (int i = 0; i < key.length(); ++i) {

hashCode += (int) (key.charAt(i));

}

}

return hashCode;

}

private Node[] resize() {

Node[] prevBasket = nodeBasket;

basketNumbers \*= 2;

Node[] newBasket = new Node[basketNumbers];

for (int i = 0; i < prevBasket.length; i++) {

if (prevBasket[i] != null) {

Node currentNode = prevBasket[i];

do {

if (newBasket[currentNode.hash % newBasket.length] == null) {

newBasket[currentNode.hash % newBasket.length] = new Node(currentNode.hash, currentNode.key, currentNode.value);

} else {

Node nodeBuffer = newBasket[currentNode.hash % newBasket.length];

do {

if (nodeBuffer.next == null) {

nodeBuffer.next = new Node(currentNode.hash, currentNode.key, currentNode.value);

break;

}

nodeBuffer = nodeBuffer.next;

} while (nodeBuffer != null);

}

currentNode = currentNode.next;

} while (currentNode != null);

}

}

return newBasket;

}

public MapMethods(int basketNumbers, double loadFactor) {

if (basketNumbers < 1 || loadFactor > 1 || loadFactor <= 0) {

throw new IllegalArgumentException("illegal arguments");

} else {

this.basketNumbers = basketNumbers;

nodeBasket = new Node[basketNumbers];

this.loadFactor = loadFactor;

elementsNumber = 0;

}

}

public MapMethods() {

this(DEFAULT\_SIZE, LOAD\_FACTOR);

}

public int size() {

return elementsNumber;

}

public boolean isEmpty() {

return elementsNumber == 0;

}

public boolean containsKey(String key) {

int hash = hashCalc(key);

if (nodeBasket[hash % basketNumbers] != null) {

Node currentNode = nodeBasket[hash % basketNumbers];

while (currentNode != null) {

if (Objects.equals(currentNode.key, key)) {

return true;

}

currentNode = currentNode.next;

}

}

return false;

}

public boolean containsValue(Integer value) {

if (this.isEmpty()) {

return false;

}

for (Node currentNode : nodeBasket) {

if (currentNode != null) {

do {

if (Objects.equals(currentNode.value, value)) {

return true;

}

currentNode = currentNode.next;

} while (currentNode != null);

}

}

return false;

}

public Integer get(String key) {

int hash = hashCalc(key);

if (nodeBasket[hash % nodeBasket.length] == null) {

return null;

}

Node currentNode = nodeBasket[hash % nodeBasket.length];

while (currentNode != null) {

if (Objects.equals(currentNode.key, key)) {

return currentNode.value;

}

currentNode = currentNode.next;

}

return null;

}

public Integer put(String key, Integer value) {

int hash = hashCalc(key);

if (nodeBasket[hash % basketNumbers] != null) {

Node currentNode = nodeBasket[hash % basketNumbers];

do {

if (Objects.equals(currentNode.key, key)) {

int buffer = currentNode.value;

currentNode.value = value;

return buffer;

}

if (currentNode.next == null) {

Node nextNode = new Node(hash, key, value);

currentNode.next = nextNode;

elementsNumber++;

break;

}

currentNode = currentNode.next;

} while (currentNode != null);

} else {

Node newNode = new Node(hash, key, value);

newNode.next = null;

elementsNumber++;

nodeBasket[hash % basketNumbers] = newNode;

}

if (elementsNumber >= basketNumbers \* loadFactor) {

nodeBasket = resize();

}

return null;

}

public Integer remove(String key) {

int hash = hashCalc(key);

if (nodeBasket[hash % basketNumbers] != null) {

Node currentNode = nodeBasket[hash % nodeBasket.length];

Node bufferNode = null;

while (currentNode != null) {

if (Objects.equals(currentNode.key, key)) {

Integer returnInt = currentNode.value;

if (bufferNode == null) {

nodeBasket[hash % basketNumbers] = currentNode.next;

} else {

bufferNode.next = currentNode.next;

}

--elementsNumber;

return returnInt;

}

bufferNode = currentNode;

currentNode = currentNode.next;

}

}

return null;

}

public void putAll(Map map) {

if (map != null) {

MapMethods secondMap = (MapMethods) map;

for (Node node : secondMap.nodeBasket) {

if (node != null) {

do {

put(node.key, node.value);

node = node.next;

} while (node != null);

}

}

}

}

public void clear() {

elementsNumber = 0;

basketNumbers = DEFAULT\_SIZE;

nodeBasket = new Node[DEFAULT\_SIZE];

}

}

# ПРИЛОЖЕНИЕ Г

Листинг 17 – Код класса Main

package mai;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

String selectCollection;

final String arraySelect = "ARRAY";

final String linkedSelect = "LINKED";

final String mapSelect = "MAP";

final int EMPTY = 0;

boolean check = true;

do {

selectCollection = input.nextLine();

check = true;

if (selectCollection.matches("^ARRAY$|^LINKED$|^MAP$")) {

check = false;

} else {

System.out.println("Incorrect input");

}

} while (check);

ArrayMethods array = new ArrayMethods();

LinkedMethods linked = new LinkedMethods();

MapMethods map = new MapMethods();

String inputInColletction;

do {

inputInColletction = input.nextLine();

check = true;

if (inputInColletction.equals("END")) {

check = false;

break;

}

switch (selectCollection) {

case arraySelect:

array.add(inputInColletction);

break;

case linkedSelect:

linked.add(inputInColletction);

break;

case mapSelect:

boolean secCheck;

int value = EMPTY;

do {

System.out.print("input value: ");

if (input.hasNextInt()) {

value = input.nextInt();

secCheck = true;

if (value <= 0) {

System.out.println("the value must be greater than 0");

secCheck = false;

}

} else {

input.nextLine();

System.out.println("You input not an integer value.");

secCheck = false;

}

} while (!secCheck);

input.nextLine();

map.put(inputInColletction, value);

break;

}

} while (check);

switch (selectCollection) {

case arraySelect:

System.out.println(array.size());

break;

case linkedSelect:

System.out.println(linked.size());

break;

case mapSelect:

System.out.println(map.size());

break;

}

System.out.println("show all elements of the collection?");

String choice = input.nextLine();

if (choice.equals("YES")) {

switch (selectCollection) {

case arraySelect:

for (String element : array.toArray()) {

System.out.println(element);

}

break;

case linkedSelect:

for (String element : linked.toArray()) {

System.out.println(element);

}

break;

case mapSelect:

for (MapMethods.Node node : map.nodeBasket) {

if (node != null) {

System.out.println(node.key + " -> " + node.value);

}

}

break;

}

}

long time = EMPTY;

System.out.println("Do you want get element?");

if (input.nextLine().equals("YES")) {

int index = 0;

time = System.currentTimeMillis();

switch (selectCollection) {

case arraySelect:

check = false;

do {

System.out.print("Enter index: ");

if (input.hasNextInt()) {

index = input.nextInt();

check = true;

if (index <= 0) {

System.out.println("Enter index > 0.");

check = false;

}

} else {

input.nextLine();

System.out.println("Not an integer.");

check = false;

}

} while (!check);

System.out.println(array.get(index));

break;

case linkedSelect:

do {

System.out.print("Enter index: ");

if (input.hasNextInt()) {

index = input.nextInt();

check = true;

if (index <= 0) {

System.out.println("Enter index > 0.");

check = false;

}

} else {

input.nextLine();

System.out.println("Not an integer.");

check = false;

}

} while (!check);

System.out.println(linked.get(index));

break;

case mapSelect:

System.out.println("Enter key: ");

String key = input.nextLine();

System.out.println(map.get(key));

break;

}

}

if (time != 0) {

double currentTime = System.currentTimeMillis();

System.out.println("Поиск по элементам занял " + (currentTime - time) + " миллисекунд.");

}

}

}