Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут атомної і теплової енергетики

Кафедра цифрових технологій в енергетиці

**Проектування та аналіз обчислювальних алгоритмів**

ЗВІТ ДО

КОМП’ЮТЕРНОГО ПРАКТИКУМУ № 3

«Списки»

*(Тема роботи)*

Варіант № 16

Виконав: студент 2-го курсу

гр. ТР-12

Каркушевський В.Л.

*(П.І.Б.)*

Оцінка \_\_\_\_\_\_\_\_\_ Перевірив: Крячок О.С.

Дата « 14 » листопала 2022 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(П.І.Б., підпис)*

Київ – 2022

**ЗАВДАННЯ:**

1. Дослідити особливості створення однонаправлених і двонаправлених списків.

2. Вивчити і реалізувати механізми додавання нових записів у список, пошуку записів у списку за певними полями, видалення записів зі списку та редагування знайдених записів, а також збереження всього списку у файлі та зчитування списку із файлу до пам’яті з відновленням всіх зв’язків.

3. Розробити Блок-схему програмного алгоритму.

4. Оформити ЗВІТ до лабораторної роботи згідно вимог та методичних рекомендацій.

РЕЗУЛЬТАТ РОБОТИ ПОТРІБНО:

1. Роздрукувати (вивести на екран) попередньо сформовані та підготовлені для запису в файл дані.

2. Роздрукувати (вивести на екран) результат виконання операції читання даних із файлу.

3. Відкритий для редагування програмний код розмістити на сайті https://replit.com/ (посилання через кнопку «+ Invite»).

4. ЗВІТ до комп’ютерного практикуму № 3 для перевірки додати в свій Клас на ресурсі https://classroom.google.com/.

Індивідуальне завдання

16. Структура «Файл»: назва, розмір, дата створення, тип.

**Теоретична частина**

Зв'язний список – це структура даних, яка складається з вузлів (Node). Списки досить ефективнi щодо операцiй додавання або видалення елементу в довiльному мiсцi списку, виконуючи їх за постiйний час. Списки використовуються замiсть масивiв для зберiгання й опрацювання однотипних даних, кiлькiсть яких заздалегiдь є невiдома й може змiнюватися у процесi роботи. У списках просто органiзувати процеси видалення елементу чи його вставку на довiльне мiсце .

Вузли (Node) є фундаментальними будівельними блоками багатьох структур даних. Вони є основою для зв’язаних списків, стеків, черг, дерев тощо.

Окремий вузол містить поля даних та посилань на інші вузли. Кожна структура даних додає до цих функцій додаткові обмеження або поведінку, щоб створити потрібну структуру .

За кількістю полів вузла розрізняють :

* + Однозв’язний список (посилання на наступний елемент)
  + Двозв’язний список (посилання на наступний і попередній елемент)

За способом зв'язку вузлів розрізняють:

* Лінійні списки (останній елемент вказує на NULL)
* Циклічні списки (останній елемент вказує на перший)

Однозв’язний список – це структура даних, яка складається з двох вузлів, перший – частина даних, друга – посилання на наступний елемент (рисунок 1.1).

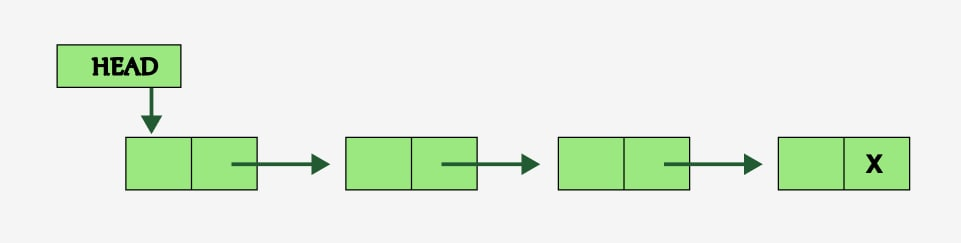


Рисунок 1.1 – Однозв’язний список

В однозв’язному списку доступ можна отримати лише в одному напрямку. Він використовує менше місця в пам'яті та є менш ефективний у порівнянні з двозв’язаним списком. Для цієї структури потрібна лише одна змінна вказівника списку, тобто головний покажчик, що вказує на перший вузол. У однозв’язаному списку тимчасова складність для вставки та видалення елемента зі списку O(n) .

Двозв’язний список – це структура даних, яка складається з трьох вузлів, перший – частина даних, друга – посилання на наступний елемент, третя – посилання на попередній елемент (рисунок 1.2).

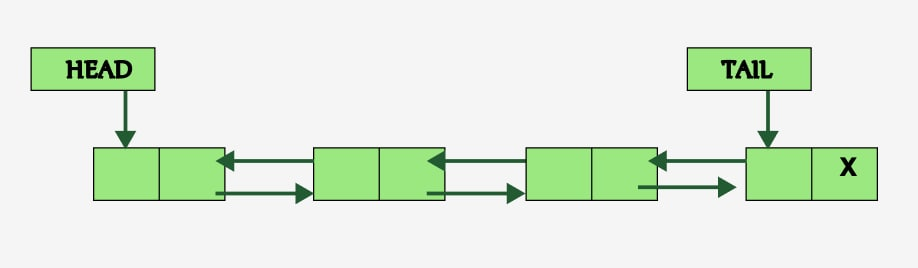


Рисунок 1.2 – Двозв’язний список

До двозв’язаного списку можна отримати доступ в обох напрямках. Він використовує більше місця в пам'яті та є більш ефективним у порівняні з однозв’язним списком. Для цієї структури потрібні дві змінні вказівника списку, які вказують на перший вузол та останній. У двозв’язаному списку тимчасова складність для вставки та видалення елемента дорівнює O(1).

**Описання програми**

У цій програмі розроблений двозв’язний список. Список створено для параметризованого типу даних У ньому присутні такі методи як:

* size() - метод для виведення розміру
* isEmpty() - метод для перевірки списку на наявність елементів
* addFirst() - метод для додавання елементу до списку у початок
* addLast() - метод для додавання елементу до списку у кінець
* removeFirst() - метод для видалення першого елементу списку
* removeLast() - метод для видалення останнього елементу списку
* removeByIndex() - метод для видалення елементу за індексом
* clear() - метод для очищення списку
* get() - метод для знаходження елемента списку за заданим індексом
* set() - метод для заміни елемента листа за заданим індексом
* indexOf() - пошук індекса елементу списку

Розроблено структуру у вигляді файлу під назвою класу MyFile (рисунок 1.3), яка містить у собі поля такі як:

* Ім’я
* Дата створення
* Формат
* Розмір

Також у цій структурі створено гетери та сетери, та перевизначено метод toString() який буде виводить зміст одного файлу.

Cтворено клас MyOS у якій буде зберігатися список файлів MyFile, та розроблено мінімальний функціонал, для додавання файлів, вилучення, пошуку та заміни деяких полів. Час створення файлу записується автоматично за допомогою Date. При використанні метода addFile використовується затримка в 1 секунду, для зміни часу при додаванні різних файлів. Також створенно перегрузку на метод searchSize() що дозволяє шукати файли за точно вказаним розміром, та за розміром в діапазоні. Також створенно перевантажений метод toString() для виводу усіх файлів у вигляді таблиці.

У головному методі створено об’єкт типу MyOS. Також зчитування даних відбувається з текстового файлу. Реалізовано за допомогою FileReader та Scanner. За допомогою циклу додано файли. Також для наглядності використано деякі методи, такі як deleteFile(), renameFile(), changeFormat(), changeSize(), searchName(), searchFormat(), searchSize(). Усі проміжні результати виведені у консоль.

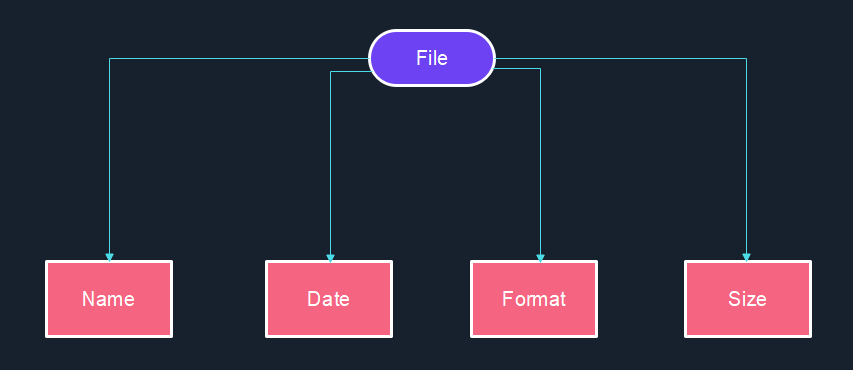
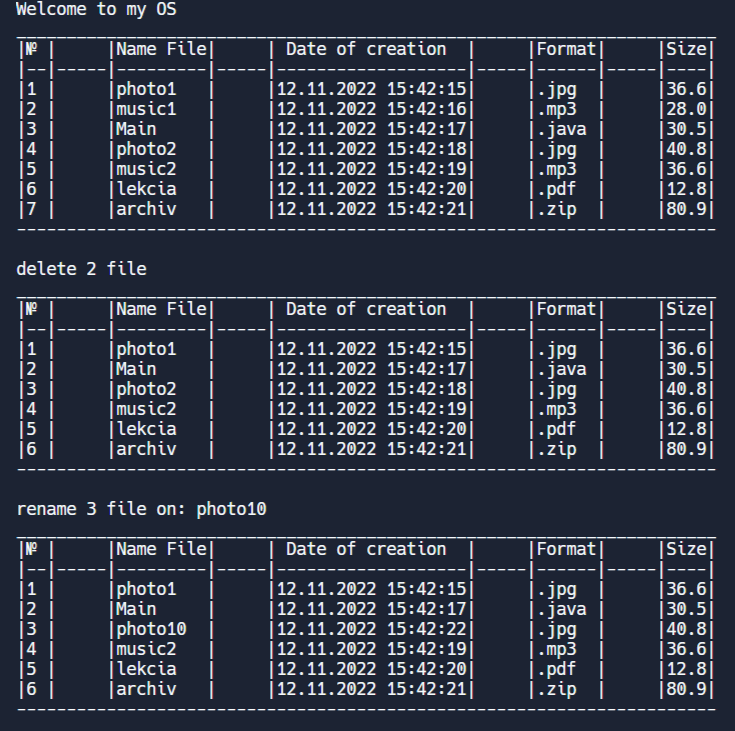
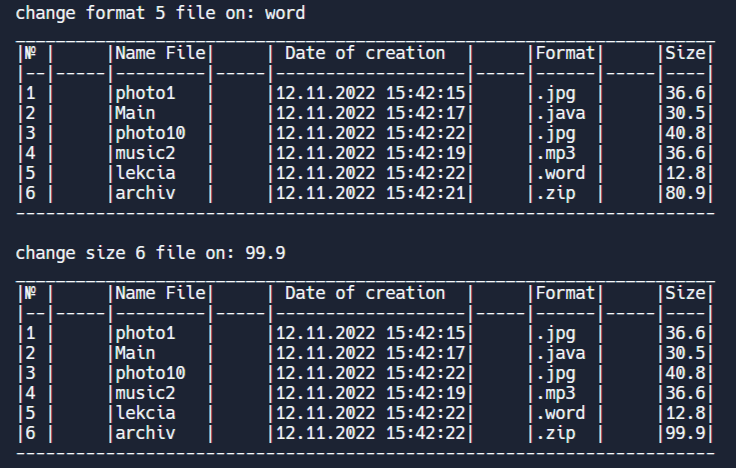
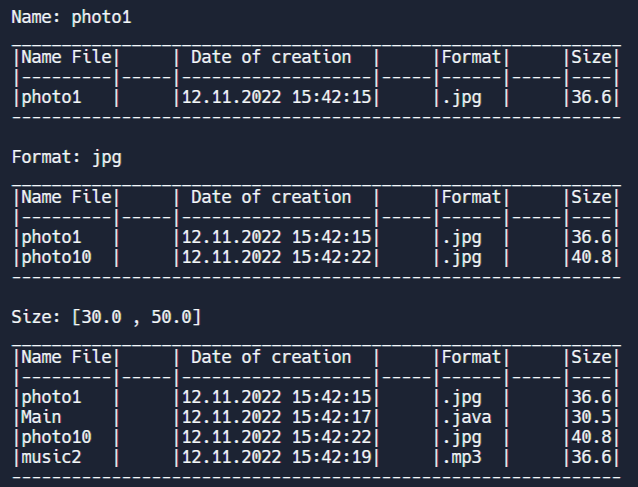
****

Рисунок 1.3 – Структура «Файл»

**Результати роботи**

****

****

****

**Висновки**

На цій лабораторній роботі було ознайомлено з теоретичним матеріалом. Досліджено особливості створення однонаправлених і двонаправлених списків. Розроблено двонаправлений список, структура “Файл”, структура “Операційна система”. Утворено ввід даних з файлу. . Вивчино і реалізовано механізми додавання нових записів у список, пошуку записів у списку за певними полями, видалення записів зі списку та редагування знайдених записів, а також збереження всього списку у файлі та зчитування списку із файлу до пам’яті з відновленням всіх зв’язків.

**Додатки**

**Посилання на Replit.com**

<https://replit.com/@TR-12-Karkushie/Lab3#Main.java>

**Код програми**

**DoublyLinkedList**

*import* java.util.Iterator;  
*import* java.util.NoSuchElementException;  
  
  
*public class* DoublyLinkedList<E> *implements* Iterable<E> {  
  
 *private* Node<E> firstNode;  
 *private* Node<E> lastNode;  
 *private int* size = 0;  
  
  
 *public* DoublyLinkedList() {*//конструктор* lastNode = *new* Node<E>(*null*, firstNode, *null*);  
 firstNode = *new* Node<E>(*null*, *null*, lastNode);  
 }  
  
  
 *public int* size() {  
 *return* size;  
 }*//метод для виведення розміру  
  
 public boolean* isEmpty() {*//метод для перевірки листа на наявність елементів  
 return* size == 0;  
 }  
  
  
  
  
 *public void* addFirst(E e) {*//метод для додавання елементу до списку у початок* Node<E> next = firstNode;  
 next.setElem(e);  
 firstNode = *new* Node<>(*null*, *null*, next);  
 *if* (isEmpty()) lastNode.setPrev(next);  
  
 next.setPrev(firstNode);  
 size++;  
 }  
  
  
 *public void* addLast(E e) {*//метод для додавання елементу до списку у кінець* Node<E> prev = lastNode;  
 prev.setElem(e);  
 lastNode = *new* Node<E>(*null*, prev, *null*);  
 *if* (isEmpty()) firstNode.setNext(prev);  
 prev.setNext(lastNode);  
 size++;  
 }  
  
  
 *public void* removeFirst() {*//метод для видалення першого елементу списку  
 if* (isEmpty()) *throw new* NoSuchElementException("List is clear");  
 firstNode = firstNode.getNext();  
 size--;  
  
 }  
  
 *public void* removeLast() {*//метод для видалення останнього елементу списку  
 if* (isEmpty()) *throw new* NoSuchElementException("List is clear");  
  
 lastNode = lastNode.getPrev();  
 size--;  
 }  
  
 *public void* removeByIndex(*int* index) {*//метод для видалення елементу списку за індексом  
 if* (index < 0 || index >size) *throw new* NoSuchElementException("No element index");  
 *if* (index == 0) {  
 removeFirst();  
  
 } *else if*(index==size-1){  
 removeLast();  
  
 }*else* {  
  
 Node<E> temp = firstNode.getNext();  
 *for* (*int* i = 0; i < index - 1; i++) {  
 temp.getNext();*//10* }  
 Node<E> temp2 = temp.getNext();*//15* temp.setNext(temp2.getNext());  
 temp2.getNext().setPrev(temp);  
 temp2.setElem(*null*);  
 size--;  
  
 }  
 }  
  
  
  
 *public void* clear() {*//метод для очищення списку* size = 0;  
 lastNode = *new* Node<E>(*null*, firstNode, *null*);  
 firstNode = *new* Node<E>(*null*, *null*, lastNode);  
  
 }  
  
 *public* E get(*int* index) {*//метод для знаходження елемента списку за заданим індексом  
  
 if* (index < 0 || index >= size) *throw new* NoSuchElementException("No element index");  
 Node<E> temp = firstNode.getNext();  
 *for* (*int* i = 0; i < index; i++) {  
 temp = getNext(temp);  
 }  
 *return* temp.getElem();  
 }  
  
 *public void* set(*int* index, E value) {*//метод для заміни елемента листа за заданим індексом  
 if* (index < 0 || index >= size) *throw new* NoSuchElementException("No element index");  
 Node<E> temp = firstNode.getNext();  
 *for* (*int* i = 0; i < index; i++) {  
 temp = getNext(temp);  
 }  
 temp.setElem(value);  
 }  
  
  
 *public int* indexOf(E elem) {*//пошук індекса елементу списку* Node<E> temp = firstNode.getNext();  
 *int* index = -1;  
 *for* (*int* i = 0; i < size; i++) {  
 *if* (elem.toString().equals(temp.getElem().toString())) {  
 index = i;  
 *break*;  
 }  
 temp = getNext(temp);  
 }  
  
 *if* (index == -1) *throw new* NoSuchElementException("No element index");  
 *return* index;  
 }  
  
  
 @Override  
 *public* String toString() {*//Перевизначимо метод toString  
  
 if* (size == 0) *return* "List is clear";  
 StringBuilder sb = *new* StringBuilder();  
 sb.append("[");  
  
 Node<E> current = firstNode.getNext();  
  
 *while* (current != *null*) {  
  
 sb.append(current.getElem());  
 *if* (current.next == lastNode) {  
 *break*;  
 }  
 current = current.next;  
 *if* (current != *null*) sb.append(", ");  
 }  
  
 sb.append("]");  
 *return* sb.toString();  
 }  
  
  
 @Override  
 *public* Iterator<E> iterator() {*//визначимо ітератор  
 return new* Iterator<E>() {  
 *int* counter = 0;  
  
 @Override  
 *public boolean* hasNext() {  
 *return* counter < size;  
 }  
  
 @Override  
 *public* E next() {  
 *return* get(counter++);  
 }  
 };  
 }  
  
  
 *private* Node<E> getNext(Node<E> current) {  
 *return* current.getNext();  
 }  
  
 *private* Node<E> getPrev(Node<E> current) {  
 *return* current.getPrev();  
 }  
  
 *private class* Node<E> {*//створимо клас Node  
  
  
 private* E elem;  
 *private* Node<E> prev;  
 *private* Node<E> next;  
  
  
 *private* Node(E elem, Node<E> prev, Node<E> next) {*//конструктор  
 this*.elem = elem;  
 *this*.prev = prev;  
 *this*.next = next;  
 }  
  
 *public* E getElem() {  
 *return* elem;  
 }*//гетери і сетери  
  
 public void* setElem(E elem) {  
 *this*.elem = elem;  
 }  
  
 *public* Node<E> getNext() {  
 *return* next;  
 }  
  
 *public void* setNext(Node<E> next) {  
 *this*.next = next;  
 }  
  
 *public* Node<E> getPrev() {  
 *return* prev;  
 }  
  
 *public void* setPrev(Node<E> prev) {  
 *this*.prev = prev;  
 }  
  
  
 }  
}

**MyFile**

*public class* MyFile {  
  
 *public* String Name;  
 *public float* Size;  
 *public* String Date;  
 *public* String Format;  
  
 *public* MyFile(String name, *float* size, String date, String format) {  
 Name = name;  
 Size = size;  
 Date = date;  
 Format = format;  
 }  
  
 @Override  
 *public* String toString() {  
 *return* "|" + String.*format*("%-9s", getName()) + "| |" + getDate() + "| |" + String.*format*(".%-5s", getFormat()) + "| |" + String.*format*("%-4.1f", getSize()) + "|\n";  
 }  
  
  
 *public* String getName() {  
 *return* Name;  
 }  
  
 *public void* setName(String name) {  
 Name = name;  
 }  
  
 *public float* getSize() {  
 *return* Size;  
 }  
  
 *public void* setSize(*float* size) {  
 Size = size;  
 }  
  
 *public* String getDate() {  
 *return* Date;  
 }  
  
 *public void* setDate(String date) {  
 Date = date;  
 }  
  
 *public* String getFormat() {  
 *return* Format;  
 }  
  
 *public void* setFormat(String format) {  
 Format = format;  
 }  
}

**MyOS**

*import* java.text.DateFormat;  
*import* java.text.SimpleDateFormat;  
*import* java.util.Date;  
  
*public class* MyOS {  
 *public* DoublyLinkedList<MyFile> Files;  
 *public* Date date ;  
 DateFormat df = *new* SimpleDateFormat("dd.MM.y k:mm:ss");  
 *public* MyOS() {  
 System.out.println("Welcome to my OS");  
 Files = *new* DoublyLinkedList<>();  
 }  
  
  
 *public void* addFile(String name, *float* size,String format ) *throws* InterruptedException {  
 date = *new* Date();  
 Files.addLast(*new* MyFile(name,size,df.format(date),format));  
 Thread.sleep(1000);  
 }  
  
 *public void* deleteFile(*int* position ) *throws* Exception {  
 *if* (position < 1 || position > Files.size()) *throw new* Exception("position error");  
 System.out.println("delete " + position + " file");  
 Files.removeByIndex(position-1);  
 }  
  
 *public void* renameFile(*int* position, String name) *throws* Exception {  
 *if* (position < 1 || position > Files.size()) *throw new* Exception("position error");  
 System.out.println("rename " + position + " file on: " + name);  
 date = *new* Date();  
 MyFile temp = Files.get(position-1);  
 temp.setName(name);  
 temp.setDate(df.format(date));  
 Files.set(position-1,temp);  
 }  
  
 *public void* changeFormat(*int* position, String format) *throws* Exception {  
 *if* (position < 1 || position > Files.size()) *throw new* Exception("position error");  
 System.out.println("change format " + position + " file on: " + format);  
 date = *new* Date();  
 MyFile temp = Files.get(position-1);  
 temp.setFormat(format);  
 temp.setDate(df.format(date));  
 Files.set(position-1,temp);  
 }  
  
 *public void* changeSize(*int* position, *float* size) *throws* Exception {  
 *if* (position < 1 || position > Files.size()) *throw new* Exception("position error");  
 System.out.println("change size " + position + " file on: " + size);  
 date = *new* Date();  
 MyFile temp = Files.get(position-1);  
 temp.setSize(size);  
 temp.setDate(df.format(date));  
 Files.set(position-1,temp);  
 }  
  
 *public void* searchName(String name){  
 System.out.println("Name: " + name);  
 *int* i = 0;  
 System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 System.out.println("|Name File| | Date of creation | |Format| |Size|");  
 System.out.println("|---------|-----|-------------------|-----|------|-----|----|");  
 *for* (MyFile temp: Files) {  
 *if* (name.equals(temp.Name)){  
 System.out.print(temp);  
 i++;  
 }  
 }  
 *if* (i==0){  
 System.out.println("No files found");  
 }  
 System.out.println("-------------------------------------------------------------\n");  
 }  
  
 *public void* searchFormat(String format){  
 System.out.println("Format: " + format);  
 *int* i = 0;  
 System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 System.out.println("|Name File| | Date of creation | |Format| |Size|");  
 System.out.println("|---------|-----|-------------------|-----|------|-----|----|");  
 *for* (MyFile temp: Files) {  
 *if* (format.equals(temp.Format)){  
 System.out.print(temp);  
 i++;  
 }  
 }  
 *if* (i==0){  
 System.out.println("No files found");  
 }  
 System.out.println("-------------------------------------------------------------\n");  
  
 }  
  
 *public void* searchSize(*float* size){  
 System.out.println("Size: " + size);  
 *int* i = 0;  
 System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 System.out.println("|Name File| | Date of creation | |Format| |Size|");  
 System.out.println("|---------|-----|-------------------|-----|------|-----|----|");  
  
 *for* (MyFile temp: Files) {  
 *if* (size == (temp.Size)){  
  
 System.out.print(temp);  
 i++;  
 }  
 }  
 *if* (i==0){  
 System.out.println("No files found");  
 }  
 System.out.println("-------------------------------------------------------------\n");  
 }  
  
  
 *public void* searchSize(*float* startSize,*float* finishSize){  
 System.out.printf("Size: [%.1f , %.1f]\n",startSize,finishSize);  
 *int* i = 0;  
 System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 System.out.println("|Name File| | Date of creation | |Format| |Size|");  
 System.out.println("|---------|-----|-------------------|-----|------|-----|----|");  
  
 *for* (MyFile temp: Files) {  
  
 *if* (temp.Size >= startSize && temp.Size <= finishSize){  
  
 System.out.print(temp);  
 i++;  
 }  
 }  
 *if* (i==0){  
 System.out.println("No files found");  
 }  
 System.***out***.println("-------------------------------------------------------------\n");  
 }  
  
  
 @Override  
 *public* String toString() {  
  
 StringBuilder sb = *new* StringBuilder();  
 *int* i = 1 ;  
 sb.append("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");  
 sb.append("|№ | |Name File| | Date of creation | |Format| |Size|\n");  
 sb.append("|--|-----|---------|-----|-------------------|-----|------|-----|----|\n");  
  
 *for* (MyFile temp: Files) {  
 sb.append("|").append(String.*format*("%-2d", i)).append("| ").append(temp);  
 i++;  
 }  
 sb.append("----------------------------------------------------------------------\n");  
  
 *return* sb.toString();  
 }  
}

**Main**

/\*

\* Лабораторна робота №3

1. Дослідити особливості створення однонаправлених і двонаправлених списків.

2. Вивчити і реалізувати механізми додавання нових записів у список, пошуку записів у списку за певними полями, видалення записів зі списку та редагування знайдених записів, а також збереження всього списку у файлі та зчитування списку із файлу до пам’яті з відновленням всіх зв’язків.

3. Розробити Блок-схему програмного алгоритму.

4. Оформити ЗВІТ до лабораторної роботи згідно вимог та методичних рекомендацій.

РЕЗУЛЬТАТ РОБОТИ ПОТРІБНО:

1. Роздрукувати (вивести на екран) попередньо сформовані та підготовлені для запису в файл дані.

2. Роздрукувати (вивести на екран) результат виконання операції читання даних із файлу.

3. Відкритий для редагування програмний код розмістити на сайті https://replit.com/ (посилання через кнопку «+ Invite»).

4. ЗВІТ до комп’ютерного практикуму № 3 для перевірки додати в свій Клас на ресурсі https://classroom.google.com/.

Завдання за варіантом:

16. Структура «Файл»: назва, розмір, дата створення, тип.

\*

\* Виконав студент групи ТР-12

\* Каркушевський Владислав

\* Номер варіанту: 16

\* \*/

import java.io.FileReader;

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws Exception {

MyOS os = new MyOS();

FileReader fr = new FileReader("info.txt");

Scanner scan = new Scanner(fr);

System.out.println("Зміст файлу info.txt");

while (scan.hasNextLine()) {

String name = scan.nextLine();

String format = scan.nextLine();

String size = scan.nextLine();

os.addFile(name,Float.parseFloat(size),format);

System.out.println(name + "\n" + format + "\n" + size);

System.out.println(scan.nextLine());

}

System.out.println(os);

os.deleteFile(2);

System.out.println(os);

os.renameFile(3,"photo10");

System.out.println(os);

os.changeFormat(5,"word");

System.out.println(os);

os.changeSize(6,99.9f);

System.out.println(os);

os.searchName("photo1");

os.searchFormat("jpg");

os.searchSize(30,50);

}

}

**info.txt**

photo1

jpg

36.6

-------

music1

mp3

28

-------

Main

java

30.5

-------

photo2

jpg

40.8

-------

music2

mp3

36.6

-------

lekcia

pdf

12.8

-------

archiv

zip

80.9

!!!!!!!end!!!!!!!