Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

Кафедра ІПЗ

**Лабораторна робота №2**

Виконав

ст.гр. ІП-22-1

Токарєв С.О.

Перевірив:

Григорчук Л.І.

Івано-Франківськ

2023

**Постановка задачі:**

-Створити програму зчитування цілих чисел з термінала й занесення їх

у список.

5.4.2 Створити програму перетворення рядка в список символів.

5.4.3 Створити програму перетворення рядка в список атомів.

5.4.4 На основі наявних даних створити ~~Пролог-Програму~~ **Java програму\***, що складається з

елементів списку.

Дано базу даних у вигляді списку елементів - "Родина". Кожна родина складається із трьох об'єктів: чоловіка, дружин і дітей. Оскільки кількість дітей у різних родинах може бути різним, тому їх необхідно представити у вигляді списку,

що складає з довільного числа елементів. Кожного члена родини у свою чергу потрібно представити структурою, що складається із чотирьох компонентів: ім'я,

прізвища, дати народження й роботи. Інформація про роботу - це або "не працює",

або вказівки коду роботи й окладу.

А) На основі створеної програми знайти всі родини із трьома дітьми.

Б) Знайти імена й прізвища всіх жінок, що мають принаймні трьох дітей

В) На основі відредагованої програми знайти імена й прізвища всіх людей з

бази даних.

Г) Знайти всіх працюючих дружин.

Д) Знайти прізвища людей, чий дохід менше ніж 100

**Теоретичні відомості:**

Зв'язані списки є важливою темою в програмуванні і структурах даних. Вони використовуються для представлення послідовних даних, таких як списки, черги та інші структури. Ось кілька теоретичних відомостей для лабораторної роботи зі зв'язаними списками на Java:

- Визначення зв'язаного списку:

Зв'язаний список - це структура даних, що складається з вузлів, де кожен вузол містить дані та посилання (або вказівник) на наступний вузол в списку.

**Основні операції:**

-Додавання елемента в початок списку (push).

-Додавання елемента в кінець списку (append).

-Видалення елемента з початку списку (pop).

-Видалення елемента з кінця списку.

-Пошук елемента в списку.

-Вставка елемента в певну позицію списку.

-Видалення елемента з певної позиції списку.

**Види зв'язаних списків:**

-Однонаправлені зв'язані списки, де кожен вузол вказує лише на наступний вузол.

-Двонаправлені зв'язані списки, де кожен вузол має посилання на попередній і наступний вузол.

-Циклічні зв'язані списки, де останній вузол вказує на перший, створюючи цикл.

**Реалізація зв'язаних списків на Java:**

Для реалізації зв'язаних списків на Java можна використовувати класи і внутрішні класи. Кожен елемент списку представляється об'єктом класу, який містить дані та посилання на наступний вузол.

**Часова складність операцій:**

Часова складність операцій над зв'язаними списками може бути різною. Додавання та видалення елементів з початку списку зазвичай має часову складність O(1), тоді як додавання або видалення з середини списку може мати складність O(n), де n - кількість елементів у списку.

**Код:**

**-Файл LinkedList.java** у якому описано структуру зв`язаного списку для виконання перших 3 завдань

package lab2;

import java.time.LocalDate;

import java.util.Scanner;

import java.util.ArrayList;

public class LinkedList {

Scanner scan = new Scanner(System.***in***);

Node head;

Node tail;

int length=0;

//Node class

public class Node{

int value;

char letter;

Node next;

public Node(int value) {

this.value = value;

}

public Node(char letter) {

this.letter = letter;

}

}

//Getters, we don't need setters

public Node getHead() {

return head;

}

public Node getTail() {

return tail;

}

public int getLength() {

return length;

}

public LinkedList() {

}

//Method to add elements in the end of LL

public void append(int value) {

Node newNode = new Node(value);

if (length ==0) {

head = newNode;

tail = newNode;

}else {

tail.next=newNode;

tail=newNode;

}

length++;

}

//Method that adds chars if we deal with chars

public void append(char letter) {

Node newNode = new Node(letter);

if (length ==0) {

head = newNode;

tail = newNode;

}else {

tail.next=newNode;

tail=newNode;

}

length++;

}

//Method to print out a LL

public void printList() {

Node temp = head;

while (temp != null) {

System.***out***.print(temp.value + " ");

temp=temp.next;

}

System.***out***.println('\n');

}

public void printCharList() {

Node temp = head;

while (temp != null) {

System.***out***.print(temp.letter + " ");

temp=temp.next;

}

System.***out***.println('\n');

}

//Task #1 - "Create a program to add the numbers, entered from the keyboard

public void numAdding() {

System.***out***.println("Enter the amount of number: ");

int amount = scan.nextInt();

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

System.***out***.println("Enter the number: ");

int num = scan.nextInt();

this.append(num);

}

}

//Task 3

public void stringToLinkedList() {

System.***out***.println("Enter the word: ");

String word = scan.nextLine();

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

this.append(word.charAt(i));

}

}

//Task 2

public void stringToList() {

System.***out***.println("Enter the word: ");

String word = scan.nextLine();

ArrayList<Character> letters = new ArrayList<Character>();

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

letters.add(word.charAt(i));

}

}

}

**-Файл LinkedListFamily.java** потрібен для виконання 4 і 5 завдання, пов`язаних з сім`ями. Список families має в собі кілька сімей (дві), об`єкти представлені класом **Family** кожна сім`я має чоловіка дружину і масив з усіма дітьми.

package lab2;

public class LinkedListFamily {

int length =0;

Family head;

Family tail;

public void append(Family fam) {

if(length==0) {

head=fam;

tail=fam;

}

else {

tail.next=fam;

tail=tail.next;

}

length++;

}

public void printFamilies() {

Family temp=head;

if(length==0) System.***out***.println("The list is empty");

else{

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

temp.printFamily();

System.***out***.println();

temp=temp.next;

}

}

}

public void checkEmployedWives() {

Family temp=head;

System.***out***.println("Employed wives: ");

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

if(temp.getWife().getJob()!="Unemployed") {

System.***out***.println(temp.getWife().getFirstName() +" "+ temp.getWife().getSecondName()+ " Works as: "+ temp.getWife().getJob());

temp=temp.next;

}

}

}

public void checkIncome() {

Family temp=head;

System.***out***.println("People with income less than 10000: ");

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

if(temp.getHusband().getIncome()<10000) {

System.***out***.println(temp.getHusband().getFirstName() +" "+ temp.getHusband().getSecondName());

}

if(temp.getWife().getIncome()<10000){

System.***out***.println(temp.getWife().getFirstName() +" "+ temp.getWife().getSecondName());

}

for(Person c: temp.children) {

if(c.getIncome()<10000) {

System.***out***.println(c.getFirstName() +" "+ c.getSecondName());

}

}

temp=temp.next;

}

}

public void childrenAmount() {

Family temp=head;

System.***out***.println("");

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

if(temp.children.size()>=3) {

System.***out***.println(temp.getHusband().getSecondName()+ "s family has "+temp.children.size()+" kids");

}

temp=temp.next;

}

}

}

**-Файл family.java** описує клас сім’ї, має методи для вивду окремо всіх членів сім`ї, дітей, кожен з об`єктів представлено класом **Person**

package lab2;

import java.util.ArrayList;

class Family {

Family next;

Person husband;

Person wife;

ArrayList<Person> children = new ArrayList<Person>();

public Family() {

}

public Person getHusband() {

return husband;

}

public void setHusband(Person husband) {

this.husband = husband;

}

public Person getWife() {

return wife;

}

public void setWife(Person wife) {

this.wife = wife;

}

public void addChild(Person child) {

children.add(child);

}

public void getChildren() {

for(Person child:children) {

System.***out***.print(child.firstName +" "+child.secondName+ ", ");

}

}

public void printFamily() {

System.***out***.print("Husband: "+husband.firstName +" "+husband.secondName+"| Wife: "+wife.firstName+" "+wife.secondName+"| Children: ");

this.getChildren();

}

}

**-Файл Person.java** описує людину загалом, кожна людина має ім`я, прізвище, день народження, роботу і дохід.

package lab2;

import java.time.LocalDate;

public class Person {

String firstName;

String secondName;

LocalDate birthday;

String job;

int income;

public Person(String firstName, String secondName, LocalDate birthday, String job, int income) {

this.firstName = firstName;

this.secondName = secondName;

this.birthday = birthday;

this.job = job;

this.income=income;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getSecondName() {

return secondName;

}

public void setSecondName(String secondName) {

this.secondName = secondName;

}

public LocalDate getBirthday() {

return birthday;

}

public void setBirthday(LocalDate birthday) {

this.birthday = birthday;

}

public String getJob() {

return job;

}

public void setJob(String job) {

this.job = job;

}

public int getIncome() {

return income;

}

public void setIncome(int income) {

this.income = income;

}

}

-**Файл Main.java** включає в себе 2 меню, одне меню загальне і меню сім`ї.

package lab2;

import java.time.LocalDate;

import java.util.Scanner;

public class Main {

public static void menu() {

int i=0;

Scanner sc = new Scanner(System.in);

do {

LinkedList list = new LinkedList();

LinkedList charList = new LinkedList();

System.out.println("1. Task #1, creating a LL from numbers from keyboard");

System.out.println("2. Task #2, making a list from a word ");

System.out.println("3. Task #3, making a linked list from a word ");

System.out.println("4. Task #4, Family");

System.out.println("5.Exit");

i=sc.nextInt();

switch(i) {

case 1:

list.numAdding();

list.printList();

break;

case 2:

charList.stringToList();

break;

case 3:

charList.stringToLinkedList();

charList.printCharList();

break;

case 4:

families();

break;

case 5: break;

}

}

while(i!=5);

sc.close();

}

public static void families() {

LinkedListFamily families = new LinkedListFamily();

//Petrenko Family

Family petrenko = new Family();

//public Person(String firstName, String secondName, LocalDate birthday, String job)

Person anna = new Person("Anna", "Petrenko",LocalDate.of(1985, 11, 9),"Manager", 20000);

Person oleksii = new Person("Oleksii", "Petrenko", LocalDate.of(1983, 5, 21),"Lawyer",34500);

Person oleksandr = new Person("Oleksandr", "Petrenko", LocalDate.of(2006, 3, 15),"Unemployed",0);

petrenko.setHusband(oleksii);

petrenko.setWife(anna);

petrenko.addChild(oleksandr);

//Johnsons family

Family johnsons = new Family();

Person robert = new Person("Robert", "Johnson", LocalDate.of(1975, 9, 4), "CEO", 40000);

Person suzan = new Person("Suzan", "Johnson", LocalDate.of(1986, 6, 7), "Unemployed",0);

Person roger = new Person("Roger", "Johnson", LocalDate.of(2000, 9, 29), "Software engineer", 25000);

Person victor = new Person("Victor", "Johnson", LocalDate.of(2004, 1,5), "Unemployed",0);

Person ginny = new Person ("Ginny", "Johnson", LocalDate.of(2007, 12, 1), "Unemployed",0);

johnsons.setHusband(robert);

johnsons.setWife(suzan);

johnsons.addChild(ginny);

johnsons.addChild(roger);

johnsons.addChild(victor);

families.append(petrenko);

families.append(johnsons);

Scanner scan = new Scanner(System.in);

int k;

do {

System.out.println("1.Print all the families");

System.out.println("2.Print all employed wives");

System.out.println("3.Print all people with income less than 10000");

System.out.println("4.Print all families with 3 children or more");

System.out.println("5.Exit");

k=scan.nextInt();

switch(k) {

case 1: families.printFamilies();

System.out.println();

break;

case 2:

families.checkEmployedWives();

System.out.println();

break;

case 3:

families.checkIncome();

System.out.println();

break;

case 4:

families.childrenAmount();

System.out.println();

break;

case 5:

break;

}

}while(k!=5);

scan.close();

}

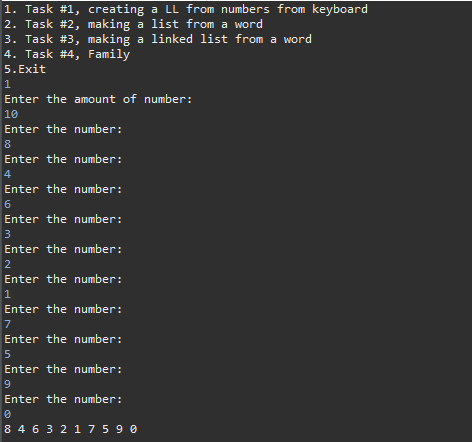
public static void main(String[] args) {

menu();

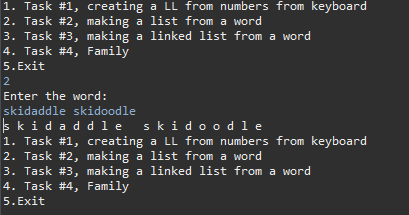
}

}

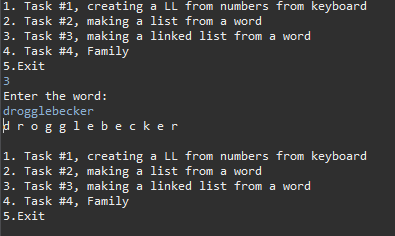
**Приклад виконання**



Завдання №1, додавання чисел до зв`язаного списку з клавіатури



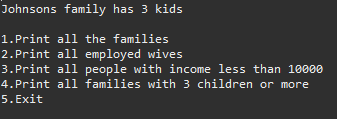
Завдання №2, додавання кожної літери слова в список



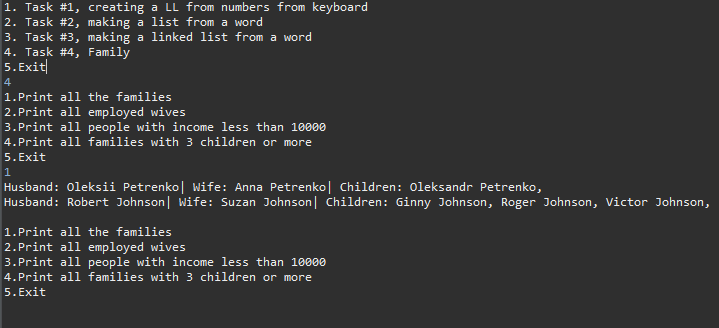
Завдання №3, перетворення рядка на атоми(Ноди)



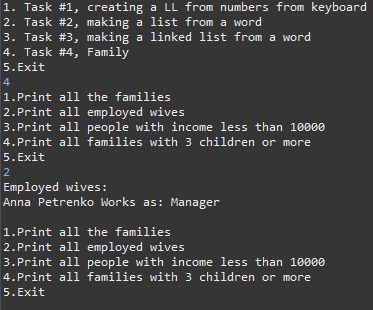
Вхідні дані сімей для подальших завдань



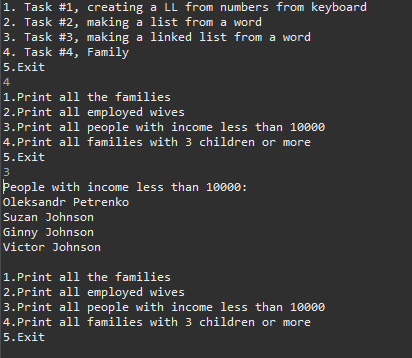
Завдання №4, вивід усіх сімей, де є 3 дитини



Завдання №5, вивести усі сім`ї



Завдання №6, знайти всіх працюючих дружин



Завдання №7 знайти всіх людей, чий дохід менше ніж 10.000

**Часова складність**

В процесі виконання лабораторної роботи було використано методи зв`язаних списків, такі як append і printList. Append в парадигмі концепції big O займає O(1), так як не були використані цикли і зі сторони програми виконується визначена кількість дій. PrintList в свою чергу – O(n), так як присутній цикл, в якому виконується певна кількість подій певну кількість повторень.

**Висновки**

Під час виконання лабораторної роботи я навчився працювати із зв`язаними списками, створювати клас Node. Який має в собі лише один рядок value, поле next і конструктор. А також більш комплексні елементи списку такий як клас Family.