Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

Кафедра ІПЗ

**Лабораторна робота №4**

**Тема: “Особливі види черг”**

Виконав

ст.гр. ІП-22-1

Токарєв С.О.

Перевірив:

Григорчук Л.І.

Івано-Франківськ

2023

**Теоретичні відомості:**

**Черга (Queue):**

Черга - це структура даних, яка працює за принципом "першим прийшов - першим вийшов" (First-In-First-Out, FIFO). Елемент, який доданий до черги першим, також вилучається першим.

Черга використовується для збереження та керування елементами в такому порядку, в якому вони надходять.

Черги використовуються в різних областях програмування, таких як обробка завдань у черговому режимі, обробка подій, обчислення шляху в графах і багато іншого.

**Реалізація черг:**

В мові програмування Java чергу можна реалізувати за допомогою інтерфейсу Queue. Один з найпоширеніших класів для реалізації черги - LinkedList.

Інші мови програмування також надають вбудовані структури даних для черг, такі як Queue у Python.

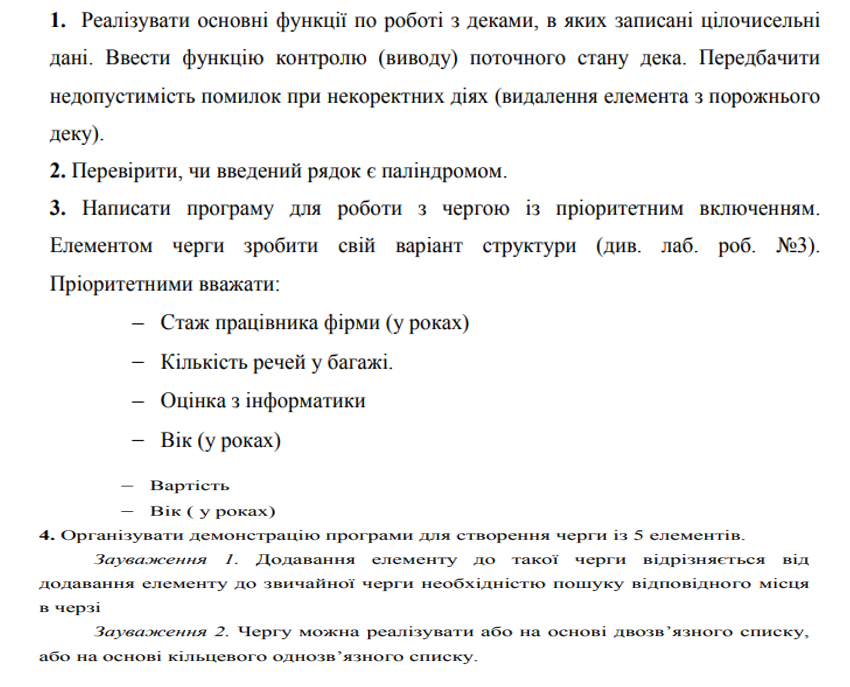
Для деяких особливих випадків і задач можуть використовуватися спеціалізовані типи черг, такі як пріоритетні черги.

**Використання черг:**

Черги використовуються в алгоритмах обробки завдань, наприклад, в системах керування завданнями, де завдання виконуються в порядку їхнього надходження.

Вони також застосовуються в алгоритмах пошуку шляху, обробці подій в інтерфейсах користувача та інших областях, де потрібно зберігати та обробляти дані у визначеному порядку.

Пріоритетні черги використовуються для обробки завдань в порядку їхнього пріоритету.

**Постановка задачі:**

**Код:**

**Person.java:**

package lab4;

import java.time.LocalDate;

public class Person {

protected String fName;

protected String sName;

protected LocalDate birthday;

protected String occupation;

protected int income;

public Person(String fName, String sName, LocalDate birthday, String occupation, int income) {

this.fName = fName;

this.sName = sName;

this.birthday = birthday;

this.occupation = occupation;

this.income = income;

}

public String getfName() {

return fName;

}

public String getsName() {

return sName;

}

*@Override*

public String toString() {

return "[Name = " + fName + ", Surname = " + sName + ", birthday = " + birthday + ", occupation = "

+ occupation + ", income = " + income + " ]";

}

public LocalDate getBirthday() {

return birthday;

}

public String getOccupation() {

return occupation;

}

public int getIncome() {

return income;

}

}

**Family.java:**

package lab4;

import java.util.ArrayList;

public class Family {

Person husband;

Person wife;

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

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 printChildren() {

for(Person c: children) {

c.toString();

}

}

public void addChild(Person c) {

children.add(c);

}

public void printFamily() {

System.***out***.println(husband.getsName() + " Family:");

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

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

for(Person c: children) {

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

}

}

}

**Main.java:**

package lab4;

import java.time.LocalDate;

import java.util.\*;

import java.util.Deque;

public class Main {

public static void families() {

PriorityQueue<Family> families = new PriorityQueue<>((f1, f2) -> f1.husband.getsName().compareTo(f2.husband.getsName()));

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);

Family smiths = new Family();

smiths.setHusband(new Person("William", "Smith", LocalDate.of(1982,03,15), "Architect", 55000));

smiths.setWife(new Person("Emma", "Smith", LocalDate.of(1984,07,2), "Designer", 50000));

smiths.addChild(new Person("Oliver", "Smith", LocalDate.of(2010,05,10), "Student", 0));

smiths.addChild(new Person("Sophia", "Smith", LocalDate.of(2015,12,03), "Unemployed", 0));

johnsons.setHusband(robert);

johnsons.setWife(suzan);

johnsons.addChild(ginny);

johnsons.addChild(roger);

johnons.addChild(victor);

families.add(smiths);

families.add(johnsons);

families.add(petrenko);

for(Family f:families) {

f.printFamily();

}

families.remove();

families.remove();

System.out.println("\nQueue after deleting family from the start and from the end:\n");

for(Family f:families) {

f.printFamily();

}

}

public static void deque() {

Deque<Integer> deque = new ArrayDeque<>();

// Додавання елементів на початок і в кінець дека

deque.addFirst(1);

deque.addLast(2);

deque.addLast(3);

deque.addFirst(4);

System.out.println("Deque: " + deque);

// Метод для видалення елемента з початку дека (pop\_front)

if (!deque.isEmpty()) {

int front = deque.removeFirst();

System.out.println("Removed from the start: " + front);

}

// Метод для видалення елемента з кінця дека (pop\_last)

if (!deque.isEmpty()) {

int last = deque.removeLast();

System.out.println("Removed from the end: " + last);

}

System.out.println("First element: " + deque.getFirst());

System.out.println("First element: " + deque.getLast());

System.out.println("Size: " + deque.size());

System.out.println("Updated Deque: " + deque);

}

public static boolean isPalindrome(String input) ;

String cleanInput = input.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();

String reversed = new StringBuilder(cleanInput).reverse().toString();

return cleanInput.equals(reversed);

}

public static void menu() {

Scanner sc = new Scanner(System.in);

int k = 0;

do {

System.out.println("1.Task #1: ");

System.out.println("2.Task #2: ");

System.out.println("3.Task #3: ");

System.out.println("4.Exit: ");

k=sc.nextInt();

switch(k){

case 1: deque();

break;

case 2:

Scanner s = new Scanner(System.in);

System.out.print("Enter the word: ");

String word;

word = s.nextLine();

boolean isPalindromic = isPalindrome(word);

System.out.println("Is Palindrome: " + isPalindromic);

break;

case 3:

families();

break;

}

}while(k!=4);

sc.close();

}

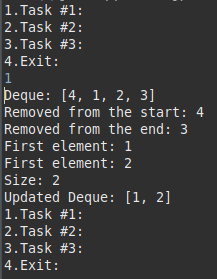
public static void main(String[] args) {

menu();

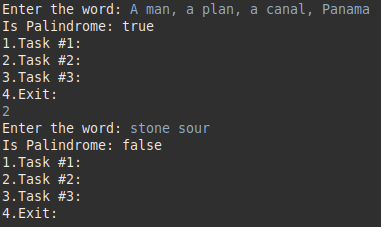
}

}

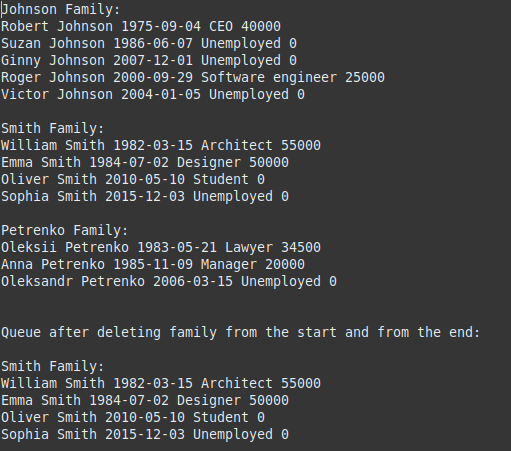
**Результат виконання:**



**Завдання №1: демонстрація основнихих методів деку**



Завдання №2: перевірка рядка чи є він паліндромом(обидва випадки)



Завдання №3: створення пріорітетної черги з сімей, видалення 2 елементів з черги, з початку і з кінця

**Висновок**

В ході виконання лабораторної роботи я навчився працювати з деками, чергами, насамперед пріорітетними чергами, вивчив їхні основні методи і створював їх на основі інших структур.