Ministerul Educației al Republicii Moldova

Universitatea Tehnică a Moldovei
Facultatea CIM

Departamentul Ingineria Software și Automatică

RAPORT

Lucrare de laborator Nr. 2
La APPOO
Tema: Biblioteca standartă a șabloanelor.

A efectuat: st. Gr. TI-142
Cojucari Dan

A verificat: Gavrișco Alexandru
Andrei Postaru

Chișinău 2017

Lucrarea de laborator nr.2

Tema: *Biblioteca standartă a şabloanelor*

Scopul lucrării: *Studierea tehnologiei de programare folosind Standard Template Library(STL) a limbajului Java.*

1. Sarcina lucrării

De elaborat 3 programe. Primul program demostrează folisrea contaierilor şi a tipurilor de date standart. Programul 2 demonstrează folosirea containerilor şi a claselor definite de utilizator. Programul 3 demonstrează folosirea algoritmilor STL.

Programul 1 va efectua:

1. Să se creeze contanier conform variantei şi să se umple cu date conform variantei.
2. Să se vizualizeze containerul.
3. Să se modifice containerul, prin ştergerea unui element şi inserarea altui.
4. Să se vizualizeze containerul utilizînd iteratorul.
5. Să se creeze al 2-lea container cu date de acelaşi tip.
6. Să se modifice primul container, ştergînd elemente de la al n-lea element şi să se adauge toate elementele containerului 2.
7. Să se vizualizeze primul şi al 2-lea container.

Programul 2 va efectua aceleaşi funcţii ca şî primul program, dar cu date de tip utilizator.

Programul 3 va efectua:

1. Să se creeze contanier conform variantei şi să se umple cu date de tip utilizator.
2. Să se sorteze containerul descrescător.
3. Să se vizualizeze containerul.
4. Să se găsească un element în container conform condiţiei căutate.
5. Elementele care satisfac condiţia să fie copiete în container de al 2-lea tip.
6. Să se vizualizeze containerul 2.
7. Să se sorteze containerele descrescător.
8. Să se vizualizeze ambele containere.
9. Să se obţine al 3-lea container prin fuziunea primelor două.
10. Să se afişeze containerul 3.
11 .Să se afişeze cîte elemente satisfac condiţia.
12.Să se determine dacă contanerul coţine elementul căutat.

3. Realizarea sarcinii

Codul sursă este anexat în anexă.

În cadrul acestei lucrări de laborator am creat același program ca și în laboratorul precedent doar că utilizînd Java. În cadrul acestei lucrări am creat mia multe clase deoarece Java este un lombaj mai mult orientat pe obiecte, și fiecare subprogram reprezintă o clasă aparte.

Pentru al dolea și al treilea program am definit o clasă Carte ce conține o variabilă integer cu numărul paginilor, dar și numele cărții. Am definit în cadrul acestei clase constructorii dar și am creat metodele necesare pentru citirea și afișarea datelor acestei clase, deoarece în Java operatorii nu pot fi supraîncărcați.

```
public class Carte implements Comparable<Carte> {
    private int pagesNumber;
    private String name;

    public int compareTo(Carte that) {
        return this.pagesNumber - that.pagesNumber;
    }

    Carte() {
        pagesNumber = 0;
        name = "Fara_nume";
    }
    Carte(int newPagesNumber, String newName) {
        pagesNumber = newPagesNumber;
        name = newName;
    }
    public void setPagesNumber(int nr) {
        pagesNumber = nr;
    }
    public void setName(String newName) {
        name = newName;
    }
    public int getPagesNumber() {
        return pagesNumber;
    }
    public String getName() {
        return name;
    }
    public void print() {
        System.out.println("Nume   : " + name);
        System.out.println("Pagini : " + pagesNumber);
    }
    public void read() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Introduceti numele cartii    : ");
        name = scanner.nextLine();
        System.out.print("Introduceti numarul de pagini: ");
        pagesNumber = scanner.nextInt();
    }
}
```

Concluzie

În urma efectuării aceste lucrări de laborator am făcut un program cu meniu, care îndeplinește cele 3 condiții. Am lucrat cu biblioteca STL a limbajului Java. Am lucrat cu containerele Queue și Vector. Am observat că containerele queue si vector nu pot fi sortate. De asemenea, am observat ca coada nu are iterator, de aceea folosim un vector temporar, in care efectuam parcurgerea, apoi copiem inapoi in coada dupa finisarea parcurgerii.

Anexa

Codul sursă:

Carte.java

```
import java.util.Scanner;

/**
 * Created by Nexus on 17-Mar-17.
 */
public class Carte implements Comparable<Carte> {
    private int pagesNumber;
    private String name;

    public int compareTo(Carte that) {
        return this.pagesNumber - that.pagesNumber;
    }

    Carte() {
        pagesNumber = 0;
        name = "Fara_nume";
    }
    Carte(int newPagesNumber, String newName) {
        pagesNumber = newPagesNumber;
        name = newName;
    }
    public void setPagesNumber(int nr) {
        pagesNumber = nr;
    }
    public void setName(String newName) {
        name = newName;
    }
    public int getPagesNumber() {
        return pagesNumber;
    }
    public String getName() {
        return name;
    }
    public void print() {
        System.out.println("Nume   : " + name);
        System.out.println("Pagini : " + pagesNumber);
    }
    public void read() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Introduceti numele cartii    : ");
        name = scanner.nextLine();
        System.out.print("Introduceti numarul de pagini: ");
        pagesNumber = scanner.nextInt();
    }
}
```

Program1.java

```
import java.util.*;

/**
 * Created by Nexus on 17-Mar-17.
 */
public class Program1 {
    private void pushDataInQueue(Queue<Character> data) {
        String tempString;
        Scanner scanner = new Scanner(System.in);
```

```
        System.out.println("Introduceti un sir de caractere caractere:");
        tempString = scanner.nextLine();

        for (char c: tempString.toCharArray()) {
            data.offer(c);
        }
    }
    private void showQueue(Queue<Character> data) {
        Queue<Character> tempData = new LinkedList<Character>(data);
        while(!tempData.isEmpty()) {
            System.out.print(tempData.element() + " ");
            tempData.remove();
        }
        System.out.println();
    }

    public void run() {
        Scanner scanner = new Scanner(System.in);
        int elementsNr, elementsForDelete;
        Queue<Character> queue1 = new LinkedList<Character>();

        pushDataInQueue(queue1);
        elementsNr = queue1.size();

        System.out.println("Coada initiala este:");
        showQueue(queue1);
        System.out.println();

        queue1.poll();
        queue1.offer('s');
        System.out.println("(A fost sters primul element si a fost adaugat altul la sfirsitul
cozii)");
        System.out.println("Coada modificata este:");
        showQueue(queue1);
        System.out.println();

        Queue<Character> queue2 = new LinkedList<Character>(queue1);
        System.out.println("(S-au copiat elementele din coada 1 in coada 2)");
        System.out.println("Coada 2:");
        showQueue(queue2);
        System.out.println();

        System.out.print("Introduceti numarul de elemente ce vor fi sterse (nr < " + elementsNr
+ ") : ");
        elementsForDelete = scanner.nextInt();
        for(int i = 0; i < elementsForDelete; i++) {
            queue1.poll();
        }
        while(!queue2.isEmpty()) {
            queue1.offer(queue2.element());
            queue2.poll();
        }
        System.out.println("(S-au copiat elementele din coada 2 in coada 1)");
        System.out.print("Coada 1: ");
        showQueue(queue1);
        System.out.print("Coada 2: ");
        showQueue(queue2);
        System.out.println();
    }
}
```

Program2.java

```
import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

/**
 * Created by Nexus on 17-Mar-17.
 */
public class Program2 {
    private void pushDataInQueue(Queue<Carte> data, int size){
        System.out.println("Introduceti" + size + " elemente: ");
        for(int i = 0; i < size; i++) {
            Carte tmp = new Carte();
            System.out.println("Introduceti cartea " + (i+1) + " :");
            tmp.read();
            data.offer(tmp);
        }
    }
    private void showQueue(Queue<Carte> data) {
        Queue<Carte> tempData = new LinkedList<Carte>(data);
        while(!tempData.isEmpty()) {
            tempData.element().print();
            tempData.remove();
        }
        System.out.println();
    }

    public void run() {
        int elementsNr, elementsForDelete;
        Queue<Carte> queue1 = new LinkedList<Carte>();;
        Scanner scanner = new Scanner(System.in);

        System.out.print("Introduceti numarul de elemente: ");
        elementsNr = scanner.nextInt();
        pushDataInQueue(queue1, elementsNr);

        System.out.println("Coada initiala este:");
        showQueue(queue1);
        System.out.println();

        Carte temp = new Carte(295, "Totul despre C/C++");

        queue1.poll();
        queue1.offer(temp);
        System.out.println("(A fost sters primul element si a fost adaugat altul la sfirsitul
cozii)");
        System.out.println("Coada modificata este:");
        showQueue(queue1);
        System.out.println();

        Queue<Carte> queue2 = new LinkedList<Carte>(queue1);
        System.out.println("(S-au copiat elementele din coada 1 in coada 2)");
        System.out.println("Coada 2:");
        showQueue(queue2);
        System.out.println();

        System.out.print("Introduceti numarul de elemente ce vor fi sterse (nr < " + elementsNr
+ ") : ");
        elementsForDelete = scanner.nextInt();
        for(int i = 0; i < elementsForDelete; i++) {
            queue1.poll();
        }
        while(!queue2.isEmpty()) {
            queue1.offer(queue2.element());
```

```
            queue2.poll();
        }
        System.out.println("(S-au copiat elementele din coada 2 in coada 1)");
        System.out.print("Coada 1: ");
        showQueue(queue1);
        System.out.print("Coada 2: ");
        showQueue(queue2);
        System.out.println();
    }
}
```

Program3.java

```
import java.util.*;

/**
 * Created by Nexus on 17-Mar-17.
 */
public class Program3 {
    private void sortQueue(Queue<Carte> data) {
        Vector<Carte> tmp = new Vector<Carte>();

        while(!data.isEmpty()) {
            tmp.add(data.element());
            data.remove();
        }

        Collections.sort(tmp);

        for (int i = tmp.size()-1; i >= 0; i--) {
            data.offer(tmp.elementAt(i));
        }
    }

    private Boolean numarPar(Carte itm) {
        return ((itm.getPagesNumber() % 2) == 0);
    }

    private Carte findCarte(Queue<Carte> data) {
        Vector<Carte> tmp = new Vector<Carte>();
        Queue<Carte> tempData = new LinkedList<Carte>(data);

        while(!tempData.isEmpty()) {
            tmp.add(tempData.element());
            tempData.remove();
        }

        for(int i = 0; i < tmp.size(); i++) {
            if(numarPar(tmp.elementAt(i))) {
                return tmp.elementAt(i);
            }
        }
        return null;
    }

    private void copyElements(Queue<Carte> data, Vector<Carte> vct) {
        Queue<Carte> tempData = new LinkedList<Carte>(data);
        while(!tempData.isEmpty()) {
            if(numarPar(tempData.element())) {
                vct.add(tempData.element());
            }
            tempData.remove();
        }
    }
    private void showVector(Vector<Carte> data) {
        for (int i = 0; i < data.size(); i++) {
```

```
            data.elementAt(i).print();
        }
        System.out.println();
    }

    private void sortAsc(Queue<Carte> queue1, Vector<Carte> vector1) {
        Vector<Carte> tmp = new Vector<Carte>();

        while(!queue1.isEmpty()) {
            tmp.add(queue1.element());
            queue1.remove();
        }

        Collections.sort(tmp);
        Collections.sort(vector1);

        for (int i = 0; i < tmp.size(); i++) {
            queue1.offer(tmp.elementAt(i));
        }
    }

    private Queue<Carte> concat(Queue<Carte> queue1, Vector<Carte> vct) {
        Queue<Carte> queue2= new LinkedList<Carte>(queue1);
        for (int i = 0; i < vct.size(); i++) {
            queue2.offer(vct.elementAt(i));
        }
        return queue2;
    }

    private void pushDataInQueue(Queue<Carte> data, int size){
        System.out.println("Introduceti" + size + " elemente: ");
        for(int i = 0; i < size; i++) {
            Carte tmp = new Carte();
            System.out.println("Introduceti cartea " + (i+1) + " :");
            tmp.read();
            data.offer(tmp);
        }
    }
    private void showQueue(Queue<Carte> data) {
        Queue<Carte> tempData = new LinkedList<Carte>(data);
        while(!tempData.isEmpty()) {
            tempData.element().print();
            tempData.remove();
        }
        System.out.println();
    }

    public void run() {
        int elementsNr;
        Queue<Carte> queue1 = new LinkedList<Carte>();;
        Scanner scanner = new Scanner(System.in);

        System.out.print("Introduceti numarul de elemente: ");
        elementsNr = scanner.nextInt();
        pushDataInQueue(queue1, elementsNr);

        System.out.println("Coada initiala este:");
        showQueue(queue1);

        sortQueue(queue1);
        System.out.println("Coada sortata:");
        showQueue(queue1);

        Carte tmpCarte = findCarte(queue1);
        if (tmpCarte!=null) {
```

```
            System.out.println("Elementul par este:");
            tmpCarte.print();
        }
        System.out.println();

        Vector<Carte> vct = new Vector<Carte>();
        copyElements(queue1, vct);
        System.out.println("Vectorul initial:");
        showVector(vct);
        System.out.println();

        sortAsc(queue1, vct);
        System.out.println("Coada sortata:");
        showQueue(queue1);
        System.out.println("Vectorul sortat:");
        showVector(vct);

        Queue<Carte> queue2 = concat(queue1, vct);
        System.out.println("Coada + Vector :");
        showQueue(queue2);

        int length = 0;
        while(!queue2.isEmpty()) {
            if(numarPar(queue2.element())){
                length++;
            }
            queue2.remove();
        }
        if(length == 0) {
            System.out.println("Nu sunt valori pare in coada");
        } else {
            System.out.println("Sunt " + length + " valori pare");
        }
        System.out.println();
    }
}
```

Main.java

```
import java.util.Scanner;

/**
 * Created by Nexus on 17-Mar-17.
 */
public class MainClass {
    public static void main( String[] args ) {
        int command;
        while(true) {
            System.out.println("     Menu:");
            System.out.println();
            System.out.println("(1) Program 1");
            System.out.println("(2) Program 2");
            System.out.println("(3) Program 3");
            System.out.println("(0) Iesire");
            System.out.print("Commanda: ");
            Scanner scanner = new Scanner(System.in);
            command = scanner.nextInt();
            switch(command) {
                case 0:
                    System.exit(0);
                    break;
                case 1:
                    Program1 p1 = new Program1();
                    p1.run();
                    break;
                case 2:
```

```
                    Program2 p2 = new Program2();
                    p2.run();
                    break;
                case 3:
                    Program3 p3 = new Program3();
                    p3.run();
                    break;
                default:
                    System.out.println("Comanda gresita! Incearca din nou!");
                    System.out.println();
                    scanner.nextLine();
                    break;
            }
        }
    }
}
```

Bibliografie:

1. Standard Template Library (STL) [Resursă electronică]. – Regim de access: http://www.infoarena.ro/stl
2. Standard Template Library (STL) [Resursă electronică]. – Regim de access: http://vega.unitbv.ro/~cataron/Courses/PCLPII/PCLP2_Capitolul9.pdf
3. Queue Java [Resursă electronică]. – Regim de access: https://docs.oracle.com/javase/7/docs/api/java/util/Queue.html
4. Vector Java [Resursă electronică]. – Regim de access: http://docs.oracle.com/javase/7/docs/api/java/util/Vector.html