**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

Факультет инфокоммуникационных сетей и систем (ИКСС)

Лабораторная работа №2

по курсу

«Разработка Java-приложений управления телекоммуникациями»

Группа: ИКПИ-11

Выполнил студент: Дунаев В.Е.

Принял преподаватель: Белая Т.И.

Место для подписи \_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2023 г.

# **Цель работы**

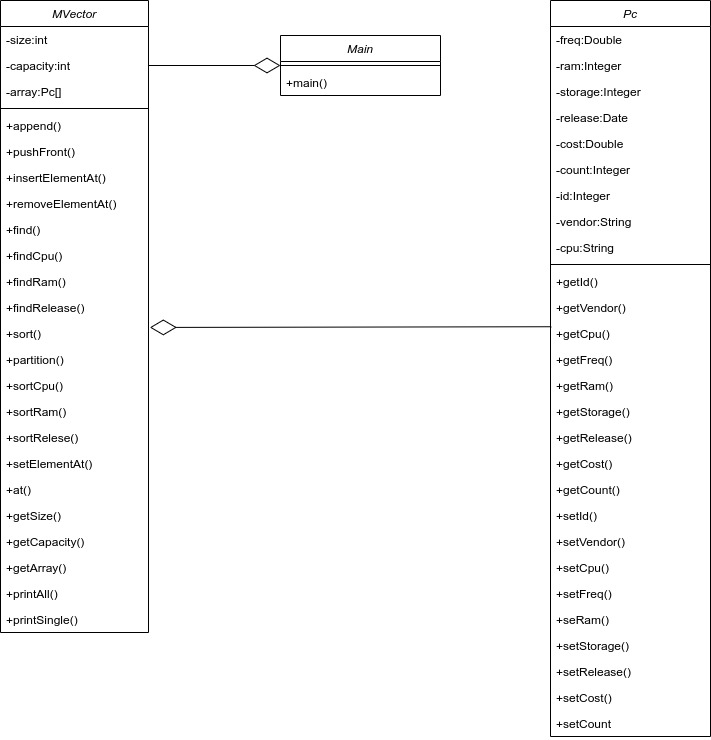
Написать программу на языке Java, реализующую контейнер, который описывает копмпьютер. Модель компьютера характеризуется кодом и маркой компьютера, типом процессора(может содержать цифры и буквы), частотой работы процессора, объемом озу, объемом жесткого диска, датой выпуска на рынок, стоимостью компьютера в рублях и кол-вом экземпляров имеющихся в наличии.

Контейнер обязан реализовать выполнение следующих функций:

1. Добавление элемента в конец, начало, середину;
2. Удаление элемента из начала, середины, конца;
3. Печати содержимого списка;
4. Поиск и сортировка по типу процессора, объему ОЗУ, дате выпуска компьютера на рынок

Для решения задач обязательно использовать функции. Все необходимые данные для функции должны передаваться им в качестве параметров. Использование глобальных переменных в функциях недопускается.

# **Структура кода**

****

## **Листинг**

## **Main.java**

import java.util.function.Function;

import java.util.function.BiConsumer;

import java.util.Date;

import java.text.\*;

class Main {

public static void main(String[] args){

SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");

Pc a = new Pc();

a.id = 1;

a.vendor = "a";

a.cpu = "i3";

a.freq = 4.2;

a.ram = 8;

a.storage = 256;

try {

a.release = format.parse("2021-01-12");

} catch (ParseException e) { e.printStackTrace(); }

a.cost = 32.5;

a.count = 5;

Pc b = new Pc();

b.id = 2;

b.vendor = "b";

b.cpu = "i5";

b.freq = 4.7;

b.ram = 16;

b.storage = 512;

try {

b.release = format.parse("2022-01-12");

} catch (ParseException e) { e.printStackTrace(); }

b.cost = 55.5;

b.count = 4;

Pc c = new Pc();

c.id = 3;

c.vendor = "c";

c.cpu = "i7";

c.freq = 5.1;

c.ram = 64;

c.storage = 1024;

try {

c.release = format.parse("2022-09-12");

} catch (ParseException e) { e.printStackTrace(); }

c.cost = 99.9;

c.count = 1;

MVector arr = new MVector(10);

arr.append(c);

arr.append(b);

arr.append(a);

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

arr.printAll();

arr.removeElementAt(0);

arr.printAll();

arr.append(c);

Function<Pc, String> g = x -> Pc.getVendor(x);

BiConsumer<Pc, String> s = (pc, val) -> Pc.setVendor(pc, val);

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

Integer index = arr.findCpu("i7");

System.out.println("Поиск по процессору: i7");

arr.printSingle(index);

// arr.sort(arr.getArray(), 0, arr.getSize() - 1, g, s);

arr.sortRelease();

arr.printAll();

index = arr.findCpu("i7");

System.out.println(index);

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

}

}

## **MVector.java**

import java.util.function.Function;

import java.util.function.BiConsumer;

import java.util.\*;

import java.text.\*;

public class MVector {

public MVector(int len) {

this.capacity = len;

this.size = 0;

this.array = new Pc[this.capacity];

}

public void append(Pc obj) {

ensureCapacity(this.size + 1);

this.array[this.size++] = obj;

}

public void pushFront(Pc obj) {

ensureCapacity(this.size + 1);

Pc[] temp = new Pc[this.capacity];

temp[0] = obj;

for (int i = 1; i <= this.size; i++)

temp[i] = this.array[i - 1];

this.array = temp;

this.size++;

}

public void insertElementAt(Pc obj, int index) {

if (index < 0 || index >= this.size)

throw new NoSuchElementException("Index out of array");

ensureCapacity(this.size + 1);

Pc[] temp = new Pc[this.capacity];

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

temp[i] = this.array[i];

temp[index] = obj;

for(int i = index + 1; i <= this.size; ++i)

temp[i] = this.array[i - 1];

this.array = temp;

this.size++;

}

public void removeElementAt(int index) {

if (index < 0 || index >= this.size)

throw new NoSuchElementException("Index out of array");

Pc[] temp = new Pc[this.capacity];

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

temp[i] = this.array[i];

for(int i = index + 1; i < this.size; ++i)

temp[i - 1] = this.array[i];

this.array = temp;

this.size--;

}

public <J> Integer find(Pc arr[], J value, Function<Pc, J> getter) {

for (int i = 0; i < arr.length - 1; ++i) {

if (getter.apply(arr[i]) == value)

return i;

}

return -1;

}

public Integer findCpu(String value) {

Function<Pc, String> getter = x -> Pc.getCpu(x);

return this.find(this.array, value, getter);

}

public Integer findRam(Integer value) {

Function<Pc, Integer> getter = x -> Pc.getRam(x);

return this.find(this.array, value, getter);

}

public Integer findRelease(Date value) {

Function<Pc, Date> getter = x -> Pc.getRelease(x);

return this.find(this.array, value, getter);

}

public void sortCpu(){

Function<Pc, String> getter = x -> Pc.getCpu(x);

BiConsumer<Pc, String> setter = (pc, val) -> Pc.setCpu(pc, val);

this.sort(this.array, 0, size - 1, getter, setter);

}

public void sortRam(){

Function<Pc, Integer> getter = x -> Pc.getRam(x);

BiConsumer<Pc, Integer> setter = (pc, val) -> Pc.setRam(pc, val);

this.sort(this.array, 0, size - 1, getter, setter);

}

public void sortRelease(){

Function<Pc, Date> getter = x -> Pc.getRelease(x);

BiConsumer<Pc, Date> setter = (pc, val) -> Pc.setRelease(pc, val);

this.sort(this.array, 0, size - 1, getter, setter);

}

public <J extends Comparable<J>> void sort(Pc arr[], int begin, int end, Function<Pc, J> getter, BiConsumer<Pc, J> setter) {

if (begin < end) {

int partitionIndex = partition(arr, begin, end, getter, setter);

sort(arr, begin, partitionIndex-1,getter, setter);

sort(arr, partitionIndex+1, end,getter, setter);

}

}

private <J extends Comparable<J>> int partition(Pc arr[], int begin, int end, Function<Pc, J> getter, BiConsumer<Pc, J> setter) {

var pivot = getter.apply(arr[end]);

int i = (begin-1);

for (int j = begin; j < end; j++) {

Integer compare = getter.apply(arr[j]).compareTo(pivot);

if (compare <= 0) {

i++;

var swapTemp = arr[i];

arr[i] = arr[j];

arr[j] = swapTemp;

}

}

var swapTemp = arr[i+1];

arr[i+1] = arr[end];

arr[end] = swapTemp;

return i+1;

}

public void setElementAt(Pc obj, int index) {

if (index >= this.size || index < 0)

throw new IllegalArgumentException("Wrong index");

this.array[index] = obj;

}

public Pc at(int index) {

if (index < 0 || index >= this.size)

throw new NoSuchElementException("Wrong index");

return this.array[index];

}

public int getSize() {

return this.size;

}

public int getCapacity() {

return this.capacity;

}

public Pc[] getArray() {

return array;

}

public void ensureCapacity(int cap) {

if (this.capacity >= cap)

return;

if (cap < 0)

throw new IllegalArgumentException("Wrong capacity");

else if (cap <= 500)

this.capacity = cap \* 2;

else

this.capacity = (int)(cap \* 1.3);

Pc[] temp = new Pc[this.capacity];

for(int i = 0; i < size; ++i)

temp[i] = this.array[i];

this.array = temp;

}

public void printAll() {

SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");

System.out.println("\* Характеристики компьютера:");

System.out.println("\* 1)ID 2)Производитель 3)Процессор 4)Частота 5)ОЗУ 6)HDD 7)Дата выхода 8)Цена 9)Кол-во ");

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

System.out.print("1)" + this.array[i].id + " 2)" + this.array[i].vendor + " 3)" + this.array[i].cpu + " 4)" );

System.out.print(this.array[i].freq + " 5)" + this.array[i].ram + " 6)" + this.array[i].storage + " 7)");

if (this.array[i].release == null)

System.out.print("null 8)");

else

System.out.print(format.format(this.array[i].release) + " 8)");

System.out.println(this.array[i].cost + " 9)" + this.array[i].count);

}

}

public void printSingle(int index) {

if (index < 0 || index >= this.size)

throw new NoSuchElementException("Wrong index");

SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");

System.out.println("\n\* Характеристики компьютера:");

System.out.println("\* 1)ID 2)Производитель 3)Процессор 4)Частота 5)ОЗУ 6)HDD 7)Дата выхода 8)Цена 9)Кол-во ");

System.out.println("\* At index: " + index);

System.out.print("1)" + this.array[index].id + " 2)" + this.array[index].vendor + " 3)" + this.array[index].cpu + " 4)" );

System.out.print(this.array[index].freq + " 5)" + this.array[index].ram + " 6)" + this.array[index].storage + " 7)");

if (this.array[index].release == null)

System.out.print("null 8)");

else

System.out.print(format.format(this.array[index].release) + " 8)");

System.out.println(this.array[index].cost + " 9)" + this.array[index].count);

}

private int size;

private int capacity;

private Pc[] array;

}

## **Pc.java**

import java.util.Date;

class Pc {

public Integer id;

public String vendor;

public String cpu;

public Double freq;

public Integer ram;

public Integer storage;

public Date release;

public Double cost;

public Integer count;

public static Integer getId(Pc a) {

return a.id;

}

public static String getVendor(Pc a) {

if (a.vendor == null)

return "null";

return a.vendor;

}

public static String getCpu(Pc a) {

if (a.cpu == null)

return "null";

return a.cpu;

}

public static Double getFreq(Pc a) {

return a.freq;

}

public static Integer getRam(Pc a) {

return a.ram;

}

public static Integer getStorage(Pc a) {

return a.storage;

}

public static Date getRelease(Pc a) {

return a.release;

}

public static Double getCost(Pc a) {

return a.cost;

}

public static Integer getCount(Pc a) {

return a.count;

}

public static void setId(Pc a, Integer id) {

a.id = id;

}

public static void setVendor(Pc a, String vendor) {

a.vendor = vendor;

}

public static void setCpu(Pc a, String cpu) {

a.cpu = cpu;

}

public static void setFreq(Pc a, Double freq) {

a.freq = freq;

}

public static void setRam(Pc a, Integer ram) {

a.ram = ram;

}

public static void setStorage(Pc a, Integer storage) {

a.storage = storage;

}

public static void setRelease(Pc a, Date release) {

a.release = release;

}

public static void setCost(Pc a, Double cost) {

a.cost = cost;

}

public static void setCount(Pc a, Integer count) {

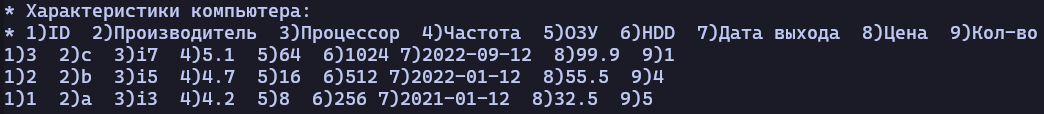
a.count = count;

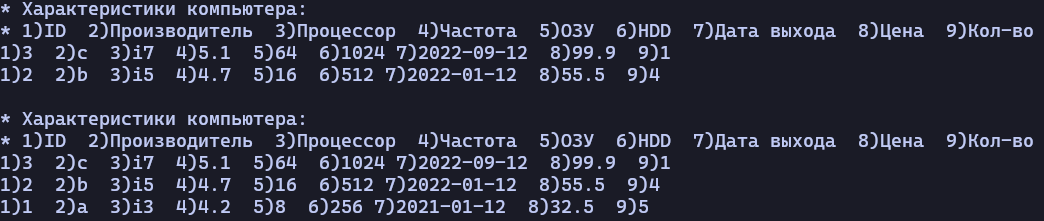
}

}

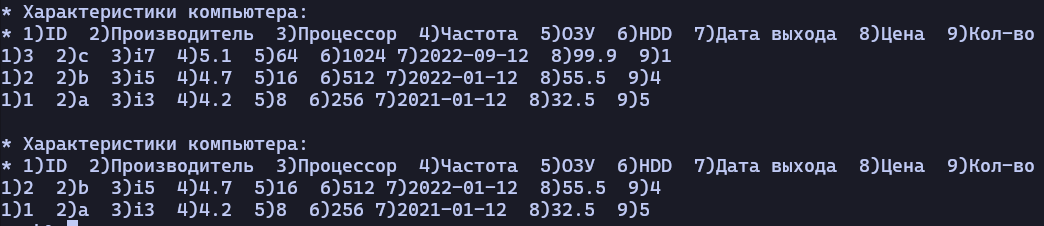
# **Тестирование**

**Вывод содержимого списка**

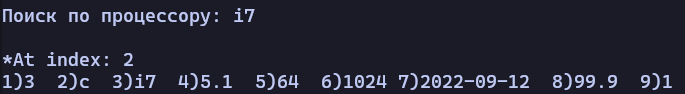
 **Добавление элемента в конец**



**Удаление нулевого элемента**



**Поиск по процессору**

**

# **Выводы**

В результате выполнения лабораторной работы была написана программа на языке Java, реализующая список для управления складом копмьютеров; изучен принципы работы классов и интерфейсов, изучена работа с методами и дженериками.