Третьяков І.С.

КІТ-118а

Лабораторна робота №15

# Колекції в java

***Мета:***

* Розширення функціональності параметризованих класів.

## **Вимоги**

1. Розробити консольну програму для реалізації завдання обробки даних згідно [прикладної області](https://oop-khpi.gitlab.io/%D0%B7%D0%B0%D0%B2%D0%B4%D0%B0%D0%BD%D0%BD%D1%8F/task07/" \l "_4).
2. Для розміщення та обробки даних використовувати контейнери (колекції) і алгоритми з [Java Collections Framework](https://docs.oracle.com/javase/8/docs/technotes/guides/collections/).
3. Забезпечити обробку колекції об'єктів: додавання, видалення, пошук, сортування згідно розділу [Прикладні задачі л.р. №10](https://oop-khpi.gitlab.io/%D0%B7%D0%B0%D0%B2%D0%B4%D0%B0%D0%BD%D0%BD%D1%8F/task10/" \l "_3).
4. Передбачити можливість довготривалого зберігання даних: 1) за допомогою стандартної серіалізації; 2) не використовуючи протокол серіалізації.
5. Продемонструвати розроблену функціональність в діалоговому та автоматичному режимах за результатом обробки параметрів командного рядка.

ПРИКЛАДНА ЗАДАЧА:

Планувальник. Знайти всі конференції, що пройшли за останні три роки в Харкові та області з тривалістю не менше доби.

ОПИС ПРОГРАМИ

***2.1 Опис змінних:***

ArrayList<Event> arrayList = **new** ArrayList<Event>()

// об’єкт параметризованого контейнера

Scanner scan = new Scanner(System.in); // змінна для активування зчитування з консолі

***2.2 Ієрархія та структура класів.***

Main class – головний клас. Містить метод main(точку входу у програму) та методи по роботі з програмою для реалізації індивідуального завдання.

interface Linked - інтерфейс контенеру

class Event - клас прикладної задачі планувальника

class LinkedContainer - параметризований клас-контейнер, котрий зберігає інформацію планувальника

**ТЕКСТ ПРОГРАМИ**

Main.java

package ua.khpi.oop.tretiakov15;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.TransformerException;

import java.io.IOException;

public class Main {

public static void main(String args[]) throws ClassNotFoundException, ParserConfigurationException, TransformerException, IOException {

String choose = args[0];

if(choose.equals("-auto"))

{

//automode();

Auto a = new Auto();

a.auto();

}else

{

Manual m = new Manual();

m.switchCase();

//manual();

}

}

}

Manual.java

package ua.khpi.oop.tretiakov15;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.TransformerException;

import javax.xml.transform.TransformerFactoryConfigurationError;

import java.io.\*;

import java.util.ArrayList;

import java.util.Scanner;

public class Manual {

public void switchCase()throws IOException, ClassNotFoundException, ParserConfigurationException, TransformerFactoryConfigurationError, TransformerException {

System.out.println("U have chosen manual mode");

int choose;

ArrayList<Event> arrayList = new ArrayList<Event>();

Event event1 = null;

do{

System.out.println("Choose action ");

Scanner in = new Scanner(System.in);

System.out.println("0. Exit");

System.out.println("1. Create new element");

System.out.println("2. Add elem ");

System.out.println("3. Del elem");

System.out.println("4. Clear container ");

System.out.println("5. Convert to Array ");

System.out.println("6. Serialize ");

System.out.println("7. Deserialize ");

System.out.println("8. Xml serialize");

System.out.println("9. Xml deserialize");

System.out.println("10. Kharkov events");

choose = in.nextInt();

switch (choose) {

case 1:

Participant[] par2 = new Participant[2];

par2[0] = new Participant("Игнат", "Третьяков", 18);

par2[1] = new Participant("Ростислав", "Малахов", 19);

//par2[2] = new Participant("Данил", "Макаров", 17);

Scanner din = new Scanner(System.in);

Scanner cin = new Scanner(System.in);

System.out.println("Enter date(format yyyy-mm-dd)");

String date = din.nextLine();

if(date.matches("[0-9]{4}-(0[1-9]|1[012])-(0[1-9]|1[0-9]|2[0-9]|3[01])")==true) {

System.out.println("OK");

}else {System.out.println("NOT OK CHANGE DATE");date=din.nextLine();}

System.out.println("Enter start time(format HH:MM)");

String startTime=din.nextLine();

if(startTime.matches("^([0-1]\\d|2[0-3])(:[0-5]\\d)$")==true) {

System.out.println("OK");

}else {System.out.println("NOT OK CHANGE TIME");startTime=din.nextLine();}

System.out.println("Enter duration");

int duration=cin.nextInt();

System.out.println("Enter venue");

String venue=din.nextLine();

System.out.println("Enter description");

String description=din.nextLine();

event1 = new Event(date,startTime,duration,venue,description,par2);

break;

case 2:

System.out.println(arrayList.size());

arrayList.add(event1);

System.out.println(arrayList.size());

for(Event tmp : arrayList)

{

System.out.println(tmp);

}

break;

case 3:

System.out.println(arrayList.size());

System.out.println("Еnter index: ");

int index = in.nextInt();

arrayList.remove(index);

System.out.println(arrayList.size());

break;

case 4:

arrayList.clear();

System.out.println(arrayList.size());

break;

case 5:

Object []arr = arrayList.toArray();

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

{

System.out.println(arr[i]);

}

break;

case 6:

ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream("event.txt"));

objectOutputStream.writeObject(arrayList);

objectOutputStream.close();

break;

case 7:

ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream("event.txt"));

ArrayList<Event> newEvents = (ArrayList<Event>) objectInputStream.readObject();

objectInputStream.close();

for (Event e : newEvents) {

System.out.println(e);

}

break;

case 8:

XmlWrite xmlWrite = new XmlWrite();

xmlWrite.write(arrayList,"XML.xml");

break;

case 9:

ArrayList<Event> newXml = XmlRead.read("XML.xml");

for(Event e : newXml )

{

System.out.println(e);

}

break;

case 10:

for(Event ev : arrayList)

{

if (ev.getData().compareTo("2017-01-01") > 0 && ev.getVenue() == "Kharkov" && ev.getDuration() >= 1440) {

System.out.println(ev);

}

}

break;

default:

System.out.println("Exit");

break;

}}while(choose!=0);

}

}

Auto.java

package ua.khpi.oop.tretiakov15;

import java.io.File;

import java.io.FileNotFoundException;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Scanner;

public class Auto {

public void auto()throws FileNotFoundException{

File file = new File("file.txt");

Scanner scanner = new Scanner(file);

String line = scanner.nextLine();

String[] array = line.split(" ");

// Участники

Participant par1[] = new Participant[2];

par1[0] = new Participant("Роман","Жолин",25);

par1[1] = new Participant("Евгений","Розумовский",24);

Participant[] par2 = new Participant[2];

par2[0] = new Participant("Игнат", "Третьяков", 18);

par2[1] = new Participant("Ростислав", "Малахов", 19);

String date = null;

String startTime = null;

int duration = 0;

String venue = null;

String description = null;

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

date = array[0].toString();

startTime = array[1].toString();

duration = Integer.parseInt(array[2]);

venue = array[3].toString();

description = array[4].toString();

}

Event firstEvent = new Event(date, startTime, duration, venue, description, par1);

Event secondEvent = new Event("2019-12-22", "10:00", 1000, "Kharkov", "Corporate", par2);

Event thirdEvent = new Event("2017-12-22", "10:00", 1440, "Kharkov","Corporate", par1);

Event[] events = {firstEvent, secondEvent, thirdEvent};

System.out.println("SORT BY DATE");

Arrays.sort(events);

for (Event e : events) {

System.out.println(e);

}

System.out.println("SORT BY DURATION");

ComparatorDuration comparatorDuration = new ComparatorDuration();

Arrays.sort(events, comparatorDuration);

for(Event tmpss : events)

{

System.out.println(tmpss);

}

System.out.println("SORT BY PARTICIPANT");

ComparatorParticipant comparatorParticipant = new ComparatorParticipant();

Arrays.sort(events, comparatorParticipant);

for(Event tmpp : events)

{

System.out.println(tmpp);

}

ArrayList arrayList = new ArrayList();

arrayList.add(firstEvent);

arrayList.add(secondEvent);

arrayList.add(thirdEvent);

//linkedContainer.addFirst(firstEvent);

//linkedContainer.addLast(secondEvent);

//linkedContainer.addLast(thirdEvent);

System.out.println("Container size");

System.out.println(arrayList.size());

}

}

ComparatorDuraion.java

**package** ua.khpi.oop.tretiakov15;

**import** java.util.Comparator;

**public** **class** ComparatorDuration **implements** Comparator<Event> {

@Override

**public** **int** compare(Event o1, Event o2) {

**if** (o1.getDuration() > o2.getDuration()) {

**return** 1;

}

**else** **if** (o1.getDuration() == o2.getDuration()) {

**return** 0;

}

{

**return** -1;

}

}

}

ComparatorParticipant.java

**package** ua.khpi.oop.tretiakov15;

**import** java.util.Comparator;

**public** **class** ComparatorParticipant **implements** Comparator<Event>

{

@Override

**public** **int** compare(Event o1, Event o2)

{

**if** (o1.getParticipants().length > o2.getParticipants().length) {

**return** 1;

}

**else** **if** (o1.getParticipants().length == o2.getParticipants().length) {

**return** 0;

}

{

**return** -1;

}

}

}

Event.java

**package** ua.khpi.oop.tretiakov15;

**import** java.io.Serializable;

**public** **class** Event **implements** Comparable<Event>, Serializable {

**private** String data; // Дата

**private** String startTime; // Время начала

**private** **int** duration; // Продолжительность

**private** String venue; // Место проведения

**private** String description; // Описание

**private** Participant[] participants; // Участники

**public** Event(String data, String startTime, **int** duration,

String venue, String description, Participant[] participants) {

**this**.data = data;

**this**.startTime = startTime;

**this**.duration = duration;

**this**.venue = venue;

**this**.description = description;

**this**.participants = participants;

}

**public** Event() {

}

**public** String getData() {

**return** **this**.data;

}

**public** **void** setData(String data) {

**this**.data = data;

}

**public** String getStartTime() {

**return** **this**.startTime;

}

**public** **void** setStartTime(String startTime) {

**this**.startTime = startTime;

}

**public** **int** getDuration() {

**return** **this**.duration;

}

**public** **void** setDuration(**int** duration) {

**this**.duration = duration;

}

**public** String getVenue() {

**return** **this**.venue;

}

**public** **void** setVenue(String venue) {

**this**.venue = venue;

}

**public** String getDescription() {

**return** **this**.description;

}

**public** **void** setDescription(String description) {

**this**.description = description;

}

**public** Participant[] getParticipants() {

**return** **this**.participants;

}

**public** **void** setParticipants(Participant[] participants) {

**this**.participants = participants;

}

@Override

**public** String toString() {

StringBuffer buf = **new** StringBuffer();

buf.append("Дата: " + **this**.data + "\n");

buf.append("Время начала: " + **this**.startTime + "\n");

buf.append("Продолжительность: " + **this**.duration + "\n");

buf.append("Место проведения: " + **this**.venue + "\n");

buf.append("Описание: " + **this**.description + "\n");

buf.append("Участники: ");

**for**(**int** i = 0; i < **this**.participants.length; i++) {

buf.append(**this**.participants[i].toString());

}

**return** buf.toString();

}

@Override

**public** **int** compareTo(Event o) {

Event entry = (Event) o;

**int** tmp = o.getData().compareTo(entry.getData());

// this.payment - ((SecondCreate)o).payment;

**return** tmp;

}

}

Participant.java

**package** ua.khpi.oop.tretiakov15;

**import** java.io.Serializable;

**public** **class** Participant **implements** Serializable {

**private** String name;

**private** String surname;

**private** **int** age;

**public** Participant(String name, String surname, **int** age){

**this**.name = name;

**this**.surname = surname;

**this**.age = age;

}

**public** Participant(){

}

**public** String getName() {

**return** **this**.name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getSurname() {

**return** **this**.surname;

}

**public** **void** setSurname(String surname) {

**this**.surname = surname;

}

**public** **int** getAge() {

**return** **this**.age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

@Override

**public** String toString() {

StringBuffer buf = **new** StringBuffer();

buf.append("Имя: " + **this**.name + "\n");

buf.append("Фамилия: " + **this**.surname + "\n");

buf.append("Возраст: " + **this**.age + "\n");

**return** buf.toString();

}

}

**ВАРІАНТИ ВИКОРИСТАННЯ**

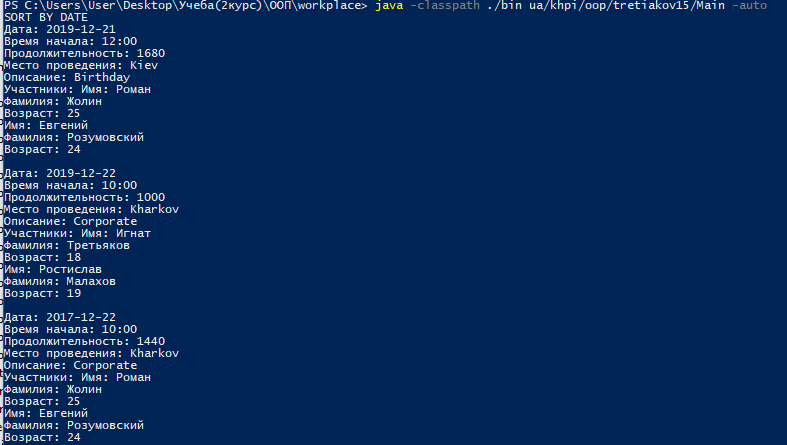


Рис. 15.1 – Результат роботи програми в автоматичному режимі.

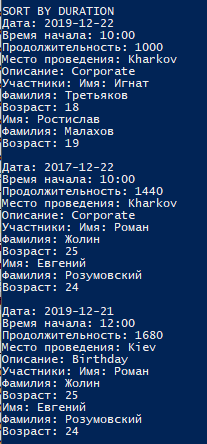


Рис. 15.2 – Результат роботи програми в автоматичному режимі.

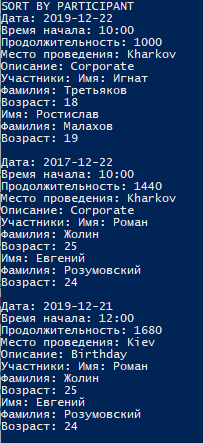


Рис. 15.3 – Результат роботи програми в автоматичному режимі.

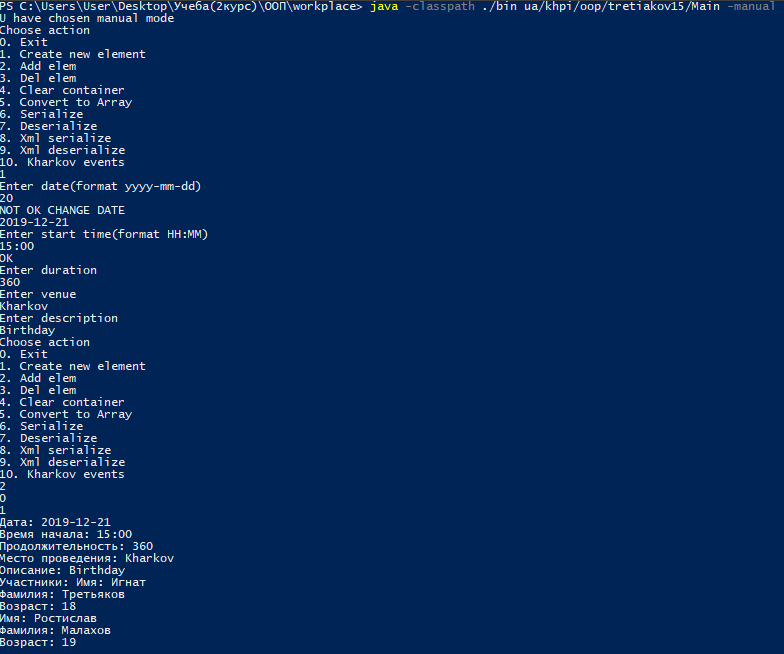


Рис. 15.4 – Результат роботи програми в ручному режимі.

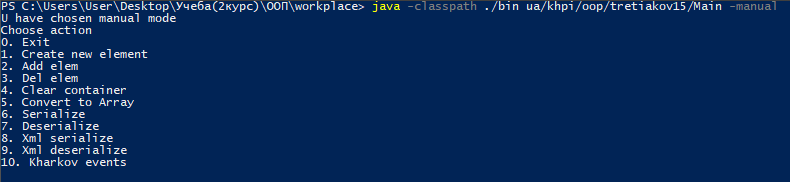


Рис. 15.5 – Результат роботи програми в ручному режимі.

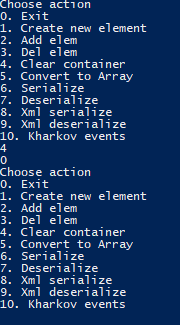


Рис. 15.6 – Результат роботи програми в ручному режимі.

Програма створена для створення, зберігання нових заходів завдяки стандартним колекціям в JAVA в данному випадку була використано колекція ArrayList.

**ВИСНОВКИ**

При виконанні лабораторної роботи набуто практичних навичок щодо розробки параметризованих класів. Завдяки данній програмі можна працювати з параметризованими классами, зберігаючі їх об’єкти у стандартній колекції в JAVA.