Міністерство освіти і науки України

Національний технічний університет

«Харківський політехнічний інститут»

Кафедра «Обчислювальна техніка та програмування»

ЗВІТ

Про виконання лабораторної роботи 15

“Колекції в Java”

Виконавець:

студент гр. КН-920В

Олексієнко Микита

Харків 2022

Лабораторна робота №15

**Колекції в Java**

**Мета**

* Ознайомлення з бібліотекою колекцій *Java SE*.
* Використання колекцій для розміщення об'єктів розроблених класів.

**1.Вимоги**

**1.1 Розробник:**

Олексієнко Микита Віталійович

студент групи КН-920В

1. варіант

**1.2 Загальне завдання**

1. Розробити консольну програму для реалізації завдання обробки даних згідно прикладної області.
2. Для розміщення та обробки даних використовувати контейнери (колекції) і алгоритми з Java Collections Framework.
3. Забезпечити обробку колекції об'єктів: додавання, видалення, пошук, сортування згідно розділу Прикладні задачі л.р. №10.
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.

**1.3 Прикладні задачі**

Автомобіль: марка, рік випуску, технічні характеристики у вигляді "параметр - значення" (серед яких витрата палива в міському і заміському циклі), технічний стан, ціна.

**2. Опис програми**

**2.1 Засоби ООП:**

У лабороторній роботі 14 , я використовую клас Program, який має у собі лише запуск меню, розробленого у лабороторної роботі 8. Ця работа передбачає удосконалення класу Menu, використанням багатопоточного програмування, завдяки якому обробка контейнеру працювати у рази швидше.

Попередньо оголошено клас Auto, який має поля з індивідуального завдання та методи обробки ціх полей.

**2.2 Ієрархія та структура класів:**

* Auto.java
* Container.java
* Iterator.java ( interface )
* LinkedList.java
* Menu.java
* Program.java

**2.3 Важливі фрагменти програми:**

Container.java:

package ua.khpi.oop.alekseenko15;  
  
import java.beans.XMLDecoder;  
import java.beans.XMLEncoder;  
import java.io.\*;  
import java.util.Iterator;  
import java.util.LinkedList;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
  
public class Container implements Iterable<Auto> {  
  
 private static final String ANSI\_RESET = "\u001B[0m";  
 private static final String ANSI\_YELLOW = "\u001B[33m";  
 private static final String ANSI\_GREEN = "\u001B[32m";  
 private static final String ANSI\_BLUE = "\u001B[34m";  
  
 private static final long DIVIDER = 1\_000\_000;  
  
 private LinkedList<Auto> container = new LinkedList<>();  
// private Iterator<Auto> iterator = container.iterator();  
  
 public int size() {  
 return container.size();  
 }  
 // добавление машины в гараж  
 public void add( Auto data ) {  
 container.push(data);  
 }  
  
 // удаление машины из гаража если такая переданная машина имеется в списке  
 public void remove( Auto data ) {  
 container.remove(data);  
 }  
  
 public void remove( int pos ) {  
 container.remove(pos);  
 }  
  
 // очищение списка  
 public void clear() {  
 container.clear();  
 }  
  
 // список в массив  
 public Object[] toArray() {  
 return container.toArray();  
 }  
  
 //из массива в список  
  
 public void fromArray( Object[] o ) {  
 container.clear();  
  
 for (Object value : o) {  
 container.push((Auto) value);  
 }  
 }  
  
 // список в строку  
 public String toString() {  
 return container.toString();  
 }  
  
 // проверка на наличие объекта в списке  
 public boolean contains( Auto i ) {  
 return container.contains(i);  
 }  
  
 public void carList() {  
  
 if ( container.isEmpty() ) {  
 System.out.println("List is empty!");  
 } else {  
 for ( Object o : container.toArray() ) {  
 System.out.println(((Auto) o).toString());  
 }  
 }  
  
 System.out.println();  
 }  
  
 public void xml\_save ( String path ) {  
 System.out.println(ANSI\_GREEN + "XML SAVING...\n" + ANSI\_RESET);  
 try (XMLEncoder xmlEncoder = new XMLEncoder(new FileOutputStream(path))) {  
 xmlEncoder.writeObject( container.toArray() );  
 xmlEncoder.flush();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 System.out.println(ANSI\_GREEN + "XML SAVING SUCCESSFULLY ENDED\n" + ANSI\_RESET);  
 }  
  
 public void xml\_load ( String path ) {  
 System.out.println("LOADING...");  
 try (XMLDecoder xmlDecoder = new XMLDecoder(new FileInputStream(path))) {  
 this.fromArray((Object[]) xmlDecoder.readObject());  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 System.out.println("LOADING SUCCESSFULLY ENDED\n");  
 }  
  
 public void standard\_save(String path ) {  
 System.out.println(ANSI\_GREEN + "TXT SAVING...\n" + ANSI\_RESET);  
 try {  
 FileOutputStream fileOutputStream = new FileOutputStream( path );  
 ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);  
 objectOutputStream.writeObject(container.toArray());  
 objectOutputStream.close();  
 } catch (Exception e) {  
 System.err.println(e);  
 }  
 System.out.println(ANSI\_GREEN + "TXT SAVING SUCCESSFULLY ENDED\n" + ANSI\_RESET);  
 }  
  
 public void standard\_load ( String path ) {  
 try {  
 System.out.println("LOADING...");  
 FileInputStream fileInputStream = new FileInputStream( path );  
 ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);  
 this.fromArray((Object[]) objectInputStream.readObject());  
 } catch (Exception e) {  
 System.err.println(e);  
 }  
 System.out.println("LOADING SUCCESSFULLY ENDED\n");  
 }  
  
 public void sortByReleaseYear () {  
  
 Object[] array = container.toArray();  
 Object temp = null;  
  
 for ( int i = 0; i < array.length; i++ ) {  
 for( int j = i + 1; j < array.length; j++) {  
 if ( ((Auto) array[i]).getReleaseYear().compareTo(((Auto) array[j]).getReleaseYear() ) > 0 )  
 {  
 temp = array[i];  
 array[i] = array[j];  
 array[j] = temp;  
 }  
 }  
 }  
  
 this.fromArray(array);  
 }  
  
 public void sortByFuel () {  
  
 Object[] array = container.toArray();  
 Object temp = null;  
  
 for ( int i = 0; i < array.length; i++ ) {  
 for( int j = i + 1; j < array.length; j++) {  
  
 final int IFuel = (Integer) ((Auto) array[i]).getUrbanFuel() + (Integer) ((Auto) array[i]).getSubUrbanFuel();  
 final int JFuel = (Integer) ((Auto) array[j]).getUrbanFuel() + (Integer) ((Auto) array[j]).getSubUrbanFuel();  
  
 if ( IFuel > JFuel )  
 {  
 temp = array[i];  
 array[i] = array[j];  
 array[j] = temp;  
 }  
 }  
 }  
  
 this.fromArray(array);  
 }  
  
 public void sortByPrice () {  
  
 // P.S.  
 // относительно этого метода сортировки не было точного понятия,  
 // как можно отсортировать массив по заданной цене.  
 // Поэтому сортировка происходит от дешевой машины к дорогой.  
  
 Object[] array = container.toArray();  
 Object temp = null;  
  
 for ( int i = 0; i < array.length; i++ ) {  
 for( int j = i + 1; j < array.length; j++) {  
  
 if ( ((Auto) array[i]).getPrice().compareTo(((Auto) array[j]).getPrice() ) > 0 )  
 {  
 temp = array[i];  
 array[i] = array[j];  
 array[j] = temp;  
 }  
 }  
 }  
  
 this.fromArray(array);  
 }  
  
 public boolean loadFromFile(String path) throws IOException {  
 String result = null;  
  
 DataInputStream reader = new DataInputStream(new FileInputStream(path));  
 int nBytesToRead = reader.available();  
  
 if(nBytesToRead > 0) {  
 byte[] bytes = new byte[nBytesToRead];  
 reader.read(bytes);  
 result = new String(bytes);  
 }  
  
 for ( String line : result.split("\n") ) {  
 if ( !addingWithChecking(line.split(", ")) ) {  
 System.out.println("loadFromFile - FAILED");  
 return false;  
 }  
 }  
  
 System.out.println("loadFromFile - SUCCESS");  
 return true;  
 }  
  
 public boolean addingWithChecking( String[] fields ) {  
  
 Auto ADDED\_CAR = new Auto();  
 int i = 0;  
  
 Pattern MODEL\_NAME\_PATTERN = Pattern.compile("[A-ZА-Я\\s\\d-]+");  
 Matcher MODEL\_NAME\_MATCHER = MODEL\_NAME\_PATTERN.matcher(fields[i]);  
  
 if ( MODEL\_NAME\_MATCHER.matches() ) {  
 ADDED\_CAR.setModel( fields[i++] );  
 } else return ERROR\_MESS();  
  
 Pattern YEAR\_PATTERN = Pattern.compile("\\d\\d\\d\\d");  
 Matcher YEAR\_MATCHER = YEAR\_PATTERN.matcher(fields[i]);  
  
 if ( YEAR\_MATCHER.matches() ) {  
 ADDED\_CAR.setReleaseYear( Integer.parseInt(fields[i++]) );  
 } else return ERROR\_MESS();  
  
 Pattern URBAN\_PATTERN = Pattern.compile("\\d+");  
 Matcher URBAN\_MATCHER = URBAN\_PATTERN.matcher(fields[i]);  
  
 if ( URBAN\_MATCHER.find() ) {  
 ADDED\_CAR.setUrbanFuel( Integer.parseInt(fields[i++]) );  
 } else return ERROR\_MESS();  
  
 Pattern SUBURBAN\_PATTERN = Pattern.compile("\\d+");  
 Matcher SUBURBAN\_MATCHER = SUBURBAN\_PATTERN.matcher(fields[i]);  
  
 if ( SUBURBAN\_MATCHER.find() ) {  
 ADDED\_CAR.setSubUrbanFuel( Integer.parseInt(fields[i++]) );  
 } else return ERROR\_MESS();  
  
 ADDED\_CAR.setTechnicalCondition(Boolean.valueOf(fields[i++]));  
  
 Pattern PRICE\_PATTERN = Pattern.compile("\\d\\d\\d\\d\\d");  
 Matcher PRICE\_MATCHER = PRICE\_PATTERN.matcher(fields[i]);  
  
 if ( PRICE\_MATCHER.find() ) {  
 ADDED\_CAR.setPrice( Integer.parseInt(fields[i++]) );  
 } else return ERROR\_MESS();  
  
 this.add(ADDED\_CAR);  
 return true;  
 }  
  
 private static boolean ERROR\_MESS () {  
 System.err.println("Error. Incorrect input.");  
 return false;  
 }  
  
 private static boolean TimeIsOver (long start\_time\_ms, long max\_time\_ms) {  
 long time\_spent = System.nanoTime() / DIVIDER - start\_time\_ms;  
 System.out.println(ANSI\_GREEN + "Time Spent From Start --> "+ ANSI\_YELLOW + time\_spent + " ms");  
  
 return max\_time\_ms > time\_spent;  
 }  
  
 public boolean findCar() {  
  
 Object[] car\_array = container.toArray();  
  
 for ( Object car : car\_array ) {  
 final int fuel = (Integer) ((Auto) car).getUrbanFuel() + (Integer) ((Auto) car).getSubUrbanFuel();  
 // если топливо не тратиться - значит машина єто електрокар  
  
 if (fuel == 0) {  
  
 Pattern YEAR\_PATTERN = Pattern.compile("2021");  
 Matcher YEAR\_MATCHER = YEAR\_PATTERN.matcher(((Auto) car).getReleaseYear().toString());  
  
 if (YEAR\_MATCHER.find()) {  
 System.out.println("\n" + ANSI\_YELLOW + car.toString() + ANSI\_RESET + "\n");  
 }  
 }  
 }  
  
 System.out.println();  
 return true;  
 }  
  
 public Iterator<Auto> iterator() {  
 return new ListIterator();  
 };  
  
 private class ListIterator implements Iterator<Auto> {  
 private int index;  
 private Object[] object = container.toArray();  
  
 @Override  
 public boolean hasNext() {  
 //System.out.println( index + "| " + container.getSize());  
  
 if ( index < container.size() ) {  
 return true;  
 } else return false;  
 }  
  
 public Auto next() {  
 if ( hasNext() ) {  
 return (Auto) object[index++];  
 }  
 return null;  
 }  
}

Menu.java :

package ua.khpi.oop.alekseenko15;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.IOException;  
import java.util.Scanner;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class Menu {  
  
 private static final String ANSI\_RESET = "\u001B[0m";  
 private static final String ANSI\_YELLOW = "\u001B[33m";  
 private static final String ANSI\_GREEN = "\u001B[32m";  
 private static final String ANSI\_BLUE = "\u001B[34m";  
 private static final String ANSI\_RED = "\u001b[31m";  
  
 private static volatile Container container = new Container();  
  
 public static void create (String[] args) throws IOException, InterruptedException {  
  
 String ConsArg = args.length==0 ? "" : args[0];  
  
 switch (ConsArg) {  
 case "-auto" -> CarContainerInit();  
 case "" -> {  
 System.out.println("\n" + ANSI\_YELLOW + "WARNING:" + ANSI\_RESET);  
 System.out.println("The program was launched without flags.\n");  
 }  
 default -> {  
 System.out.println("\n" + ANSI\_YELLOW + "WARNING:" + ANSI\_RESET);  
 System.out.println("Invalid flags.\n");  
 }  
 }  
  
 while (true) {  
 System.out.println(ANSI\_GREEN + "CHOOSE ANY TASK:" + ANSI\_RESET);  
 System.out.println(ANSI\_YELLOW + "[1] Add Car");  
 System.out.println("[2] Remove Car");  
 System.out.println("[3] Car List");  
 System.out.println("[4] Find new electrocar");  
 System.out.println("[5] Sort by Price");  
 System.out.println("[6] Standard Save");  
 System.out.println("[7] Standard Load");  
 System.out.println("[8] XML Save");  
 System.out.println("[9] XML Load");  
 System.out.println("[0] Close" + ANSI\_RESET + "\n");  
  
 Scanner scan = new Scanner(System.in);  
 int chooseNumber = scan.nextInt();  
  
 Auto car = new Auto();  
  
 switch (chooseNumber) {  
 case 1:  
 if (carAdding()) {  
 System.out.println(ANSI\_GREEN + "ADDED!\n" + ANSI\_RESET );  
 } else {  
 System.err.println("Please try add correctly");  
 }  
 break;  
 case 2: container.remove( inputCarNumber() ); break;  
 case 3: {  
 container.carList();  
 break;  
 }  
 case 4: container.findCar(); break;  
 case 5: container.sortByPrice(); break;  
 case 6: SAVING( true ); break;  
 case 7: LOADING( true ); break;  
 case 8: SAVING( false ); break;  
 case 9: LOADING( false ); break;  
 case 0:  
 return;  
 default: {  
 System.err.println("ERROR");  
 System.out.println("- Please select the correct command!");  
 }  
 }  
 }  
 }  
  
 private static boolean carAdding () {  
  
 int i = 0;  
 String[] result = new String[6];  
  
 Scanner scan = new Scanner(System.in);  
  
 System.out.println("Please type your car model:");  
 result[i++] = scan.nextLine();  
  
 System.out.println("Please type your car release car:");  
 result[i++] = scan.nextLine();  
  
 System.out.println("Please type your car urban fuel:");  
 result[i++] = scan.nextLine();  
  
 System.out.println("Please type your car suburban fuel:");  
 result[i++] = scan.nextLine();  
  
 System.out.println("Please choose your car state:");  
 System.out.println("[1] Good");  
 System.out.println("[2] Bad");  
  
 result[i++] = scan.nextLine();  
  
 System.out.println("Please type your car price:");  
 result[i++] = scan.nextLine();  
  
 if ( !container.addingWithChecking(result) ) {  
 System.out.println("loadFromFile - FAILED\n");  
 return false;  
 }  
  
 return true;  
 }  
  
 private static int inputCarNumber () {  
 Scanner scan = new Scanner(System.in);  
 String number = scan.nextLine();  
  
 if ( number.equals("") ) {  
 return 0;  
 } else return Integer.parseInt(number);  
 }  
  
 private static void SAVING( boolean standard ) {  
 System.out.println("Enter the path to your file:");  
 String PATH = "C:/Users/FORCH/Desktop";  
  
 while (true) {  
 System.out.print(PATH + "/");  
 Scanner scan = new Scanner(System.in);  
 String command = scan.nextLine();  
  
 if ( command.equals("exit") ) {  
 System.out.println("");  
 return;  
 }  
  
 // REGEX -----------------------------------  
  
 Pattern CD\_PATTERN = Pattern.compile("cd\\s.+");  
 Matcher CD\_MATCHER = CD\_PATTERN.matcher(command);  
  
 Pattern TOUCH\_PATTERN = Pattern.compile("touch\\s.+\\..+");  
 Matcher TOUCH\_MATCHER = TOUCH\_PATTERN.matcher(command);  
  
 // -----------------------------------------  
 if ( command.equals("ls") ) {  
 File dir = new File(PATH);  
 for(File item : dir.listFiles()){  
 if(item.isDirectory()){  
 System.out.println( ANSI\_GREEN + "- " + item.getName() + ANSI\_RESET);  
 } else {  
 Pattern LS\_XML\_PATTERN = Pattern.compile(".+\\.xml");  
 Matcher LS\_XML\_MATCHER = LS\_XML\_PATTERN.matcher(item.getName());  
  
 if ( LS\_XML\_MATCHER.find() ) {  
 System.out.println(ANSI\_BLUE + "- " + item.getName() + ANSI\_RESET);  
 } else System.out.println(ANSI\_YELLOW + "- " + item.getName() + ANSI\_RESET);  
 }  
 }  
 }  
 else if (CD\_MATCHER.find()) {  
 String cd\_string = command.substring(CD\_MATCHER.start() + 3, CD\_MATCHER.end());  
 PATH = PATH + "/" + cd\_string;  
 } else if (TOUCH\_MATCHER.find()) {  
 String file\_name = command.substring(TOUCH\_MATCHER.start() + 6, TOUCH\_MATCHER.end());  
  
 File file\_path = new File(PATH + "/" + file\_name);  
  
 if ( file\_path.exists() ) {  
 System.out.println(ANSI\_YELLOW + "WARNING" + ANSI\_RESET);  
 System.out.println("The file already exists at the given path. Recreate?");  
 System.out.println("[1] - yes");  
 System.out.println("[2] - no");  
  
 Scanner recreate = new Scanner(System.in);  
 int returned = recreate.nextInt();  
  
 if ( returned == 1 ) {  
 if ( standard ) {  
 container.standard\_save(PATH + "/" + file\_name);  
 } else {  
 container.xml\_save(PATH + "/" + file\_name);  
 }  
 }  
 } else {  
 if ( standard ) {  
 container.standard\_save(PATH + "/" + file\_name);  
 } else {  
 container.xml\_save(PATH + "/" + file\_name);  
 }  
 }  
  
 }  
  
 // -----------------------------------------  
 }  
 }  
  
 private static void LOADING( boolean standard ) {  
 System.out.println("Enter the path to your file:");  
 String PATH = "C:/Users/FORCH/Desktop";  
  
 while (true) {  
 System.out.print(PATH + "/");  
 Scanner scan = new Scanner(System.in);  
 String command = scan.nextLine();  
  
 if ( command.equals("exit") ) {  
 System.out.println("");  
 return;  
 }  
  
 // REGEX -----------------------------------  
  
 Pattern CD\_PATTERN = Pattern.compile("cd\\s.+");  
 Matcher CD\_MATCHER = CD\_PATTERN.matcher(command);  
  
 Pattern CAT\_PATTERN = Pattern.compile("cat\\s.+\\..+");  
 Matcher CAT\_MATCHER = CAT\_PATTERN.matcher(command);  
  
 // -----------------------------------------  
 if ( command.equals("ls") ) {  
 File dir = new File(PATH);  
 for(File item : dir.listFiles()){  
 if(item.isDirectory()){  
 System.out.println( ANSI\_GREEN + "- " + item.getName() + ANSI\_RESET);  
 } else {  
 Pattern LS\_XML\_PATTERN = Pattern.compile(".+\\.xml");  
 Matcher LS\_XML\_MATCHER = LS\_XML\_PATTERN.matcher(item.getName());  
  
 if ( LS\_XML\_MATCHER.find() ) {  
 System.out.println(ANSI\_BLUE + "- " + item.getName() + ANSI\_RESET);  
 } else System.out.println(ANSI\_YELLOW + "- " + item.getName() + ANSI\_RESET);  
 }  
 }  
 }  
 else if (CD\_MATCHER.find()) {  
 String cd\_string = command.substring(CD\_MATCHER.start() + 3, CD\_MATCHER.end());  
 PATH = PATH + "/" + cd\_string;  
 } else if (CAT\_MATCHER.find()) {  
 String file\_name = command.substring(CAT\_MATCHER.start() + 4, CAT\_MATCHER.end());  
  
 if ( standard ) {  
 container.standard\_load(PATH + "/" + file\_name);  
 } else {  
 container.xml\_load(PATH + "/" + file\_name);  
 }  
 }  
  
 // -----------------------------------------  
 }  
 }  
  
 private static void CarContainerInit() throws IOException {  
  
 System.out.println("Welcome to automatic mode!");  
  
 container.xml\_load("lab15.xml");  
  
 if (carAdding()) {  
 System.out.println(ANSI\_GREEN + "ADDED!\n" + ANSI\_RESET );  
 } else {  
 System.err.println("Please try add correctly");  
 }  
  
 container.carList();  
  
 container.findCar();  
  
 container.sortByPrice();  
  
 container.carList();  
  
 System.exit(0);  
 }  
}

**3. Висновок:**

* Ознайомлено з бібліотекою колекцій *Java SE*.
* Використано колекції для розміщення об'єктів розроблених класів.