МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАТИКИ І ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ

"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

ЛАБОРАТОРНА РОБОТА №8

Робота з потоками в мові програмування Java

Виконав: студент 1 курсу

група: ІО-21

Кузьменко В. З.

Номер залікової книжки № 2114

Прийняв: Невдащенко М.В.

Дата:

Оцінка:

Київ - 2013

Тема: Робота з потоками в мові програмування Java.

Мета: Здобуття навичок у використанні потоків в мові програмування Java.

Опис програми: Клас WorkWithFile складається з шести методів для запису і зчитування з файла. В головному методі ініціалізуються об'єкти, та запускаються відповідні методи.

Код програми:

Головний клас:

***package*** *lab8;*

***public******class*** *Lab8 {*

*/\*\**

*\** ***@param*** *args*

*\** ***@throws*** *MyException*

*\*/*

***public******static******void*** *main(String[] args)* ***throws*** *MyException {*

*MyCollection collection =* ***new*** *MyCollection();*

*Ammunition am =* ***new*** *Ammunition("name0", 100, 200);*

*Ammunition am1 =* ***new*** *Ammunition("name1", 10, 20);*

*Ammunition am2 =* ***new*** *Ammunition("name2", 30, 40);*

*Ammunition am3 =* ***new*** *Ammunition("name2", 50, 60);*

*Ammunition am4 =* ***new*** *Ammunition("name4", 70, 80);*

*collection.add(am);*

*collection.add(am2);*

*collection.add(am3);*

*collection.add(am4);*

*WorkWithFile wwf1 =* ***new*** *WorkWithFile(collection);*

*wwf1.WriteMyCollection();*

***for*** *(Ammunition ammunition : wwf1.ReadMyCollection().toArray()) {*

*System.out.println(ammunition.getInformation());*

*}*

*Ammunition [] mas =* ***new*** *Ammunition[3];*

*mas[0]=am;*

*mas[1]=am1;*

*mas[2]=am2;*

*System.out.println();*

*WorkWithFile wwf2 =* ***new*** *WorkWithFile(mas);*

*wwf2.WriteAmmunitions();*

*wwf2.ReadAmmunition();*

***for*** *(Ammunition ammunition : wwf2.ReadAmmunition()) {*

*System.out.println(ammunition.getInformation());*

*}*

*System.out.println();*

*wwf1.WriteInformation();*

*System.out.println(wwf1.ReadInformation());*

*}*

*}*

Клас WorkWithFile

***package*** *lab8;*

***import*** *java.io.BufferedReader;*

***import*** *java.io.File;*

***import*** *java.io.FileInputStream;*

***import*** *java.io.FileNotFoundException;*

***import*** *java.io.FileOutputStream;*

***import*** *java.io.FileReader;*

***import*** *java.io.FileWriter;*

***import*** *java.io.IOException;*

***import*** *java.io.InputStream;*

***import*** *java.io.ObjectInputStream;*

***import*** *java.io.ObjectOutputStream;*

***import*** *java.io.OutputStream;*

*/\*\**

*\** ***@author*** *Lenovo Клас включає 6 методів, для виконання операції з файлами*

*\*/*

***public******class*** *WorkWithFile {*

*/\*\* Колекція для запису \*/*

***private*** *MyCollection collection;*

*/\*\* послідовність об\*єктів, які обєднані масивом \*/*

***private*** *Ammunition[] ammunitions;*

*/\*\* конструктор для колекції \*/*

***public*** *WorkWithFile(MyCollection col) {*

*collection = col;*

*ammunitions = col.toArray();*

*}*

*/\*\* конструктор для послідовності об\*єктів \*/*

***public*** *WorkWithFile(Ammunition[] am) {*

*ammunitions = am;*

*}*

*/\*\* метод записує у файл колекцію \*/*

***public******void*** *WriteMyCollection() {*

*File myCollectionFile =* ***new*** *File(*

*"C:\\Users\\Lenovo\\Desktop\\MyCollection.txt");*

*OutputStream os =* ***null****;*

***try*** *{*

*os =* ***new*** *FileOutputStream(myCollectionFile);*

*}* ***catch*** *(FileNotFoundException e) {*

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

*}*

*ObjectOutputStream oos =* ***null****;*

***try*** *{*

*oos =* ***new*** *ObjectOutputStream(os);*

*oos.writeObject(collection);*

*}* ***catch*** *(IOException e) {*

*e.printStackTrace();*

*}* ***finally*** *{*

***try*** *{*

*oos.close();*

*os.close();*

*}* ***catch*** *(IOException e) {*

*e.printStackTrace();*

*}*

*}*

*}*

*/\*\* метод зчитує колекцію, та повертає її \*/*

***public*** *MyCollection ReadMyCollection() {*

*File myCollectionFile =* ***new*** *File(*

*"C:\\Users\\Lenovo\\Desktop\\MyCollection.txt");*

*InputStream is =* ***null****;*

*MyCollection col =* ***null****;*

***try*** *{*

*is =* ***new*** *FileInputStream(myCollectionFile);*

*}* ***catch*** *(FileNotFoundException e) {*

*e.printStackTrace();*

*}*

*ObjectInputStream ois =* ***null****;*

***try*** *{*

*ois =* ***new*** *ObjectInputStream(is);*

*col = (MyCollection) (ois.readObject());*

*}* ***catch*** *(IOException e) {*

*e.printStackTrace();*

*}* ***catch*** *(ClassNotFoundException e) {*

*e.printStackTrace();*

*}* ***finally*** *{*

***try*** *{*

*ois.close();*

*is.close();*

*}* ***catch*** *(IOException e) {*

*e.printStackTrace();*

*}*

*}*

***return*** *col;*

*}*

*/\*\* метод записує послідовність об\*єктів \*/*

***public******void*** *WriteAmmunitions() {*

*File ammunitionsFile =* ***new*** *File(*

*"C:\\Users\\Lenovo\\Desktop\\ammunitions.txt");*

*OutputStream os =* ***null****;*

***try*** *{*

*os =* ***new*** *FileOutputStream(ammunitionsFile);*

*}* ***catch*** *(FileNotFoundException e) {*

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

***try*** *{*

*os.close();*

*}* ***catch*** *(IOException e1) {*

*e1.printStackTrace();*

*}*

*}*

*ObjectOutputStream oos =* ***null****;*

***try*** *{*

*oos =* ***new*** *ObjectOutputStream(os);*

*oos.writeObject(ammunitions);*

*}* ***catch*** *(IOException e) {*

*e.printStackTrace();*

*}* ***finally*** *{*

***try*** *{*

*oos.close();*

*os.close();*

*}* ***catch*** *(IOException e) {*

*e.printStackTrace();*

*}*

*}*

*}*

*/\*\* метод зчитує послідовність об\*єктів, у вигляді масиву та повертає його \*/*

***public*** *Ammunition[] ReadAmmunition() {*

*File ammunitionsFile =* ***new*** *File(*

*"C:\\Users\\Lenovo\\Desktop\\ammunitions.txt");*

*InputStream is =* ***null****;*

*Ammunition[] am =* ***null****;*

***try*** *{*

*is =* ***new*** *FileInputStream(ammunitionsFile);*

*}* ***catch*** *(FileNotFoundException e) {*

*e.printStackTrace();*

***try*** *{*

*is.close();*

*}* ***catch*** *(IOException e1) {*

*e1.printStackTrace();*

*}*

*}*

*ObjectInputStream ois =* ***null****;*

***try*** *{*

*ois =* ***new*** *ObjectInputStream(is);*

*am = (Ammunition[]) ois.readObject();*

*}* ***catch*** *(IOException e) {*

*e.printStackTrace();*

*}* ***catch*** *(ClassNotFoundException e) {*

*e.printStackTrace();*

*}* ***finally*** *{*

***try*** *{*

*ois.close();*

*is.close();*

*}* ***catch*** *(IOException e) {*

*e.printStackTrace();*

*}*

*}*

***return*** *am;*

*}*

*/\*\* метод записує на файл інформацію про об\*єкти у вигляді тексту \*/*

***public******void*** *WriteInformation() {*

*File amInformation =* ***new*** *File(*

*"C:\\Users\\Lenovo\\Desktop\\amInformation.txt");*

*FileWriter fw =* ***null****;*

***try*** *{*

*fw =* ***new*** *FileWriter(amInformation);*

*}* ***catch*** *(IOException e) {*

*e.printStackTrace();*

***try*** *{*

*fw.close();*

*}* ***catch*** *(IOException e1) {*

*e1.printStackTrace();*

*}*

*}*

***for*** *(Ammunition ammunition : ammunitions) {*

***try*** *{*

*fw.write(ammunition.getInformation());*

*}* ***catch*** *(IOException e) {*

*e.printStackTrace();*

*}* ***catch*** *(NullPointerException e) {*

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

*}*

*}*

***try*** *{*

*fw.close();*

*}* ***catch*** *(IOException e) {*

*e.printStackTrace();*

*}*

*}*

*/\*\* метод зчитує інформацію про об\*єкти, повертає текст String даних \*/*

***public*** *String ReadInformation() {*

*File amInformation =* ***new*** *File(*

*"C:\\Users\\Lenovo\\Desktop\\amInformation.txt");*

*FileReader fr =* ***null****;*

*String txt =* ***null****;*

***try*** *{*

*fr =* ***new*** *FileReader(amInformation);*

*}* ***catch*** *(FileNotFoundException e) {*

*System.out.println("Нема файла");*

*e.printStackTrace();*

***try*** *{*

*fr.close();*

*}* ***catch*** *(IOException e1) {*

*e1.printStackTrace();*

*}*

*}*

*BufferedReader br =* ***new*** *BufferedReader(fr);*

***try*** *{*

*txt = br.readLine();*

*}* ***catch*** *(IOException e) {*

*e.printStackTrace();*

*}* ***finally*** *{*

***try*** *{*

*br.close();*

*fr.close();*

*}* ***catch*** *(IOException e) {*

*e.printStackTrace();*

*}*

*}*

***return*** *txt;*

*}*

*}*

Клас MyCollection

***package*** *lab8;*

*/\*\**

*\* Клас MyCollection реалізує інтерфейс List*

*\** ***@author*** *Lenovo*

*\*/*

***import*** *java.io.Serializable;*

***import*** *java.util.ArrayList;*

***import*** *java.util.Arrays;*

***import*** *java.util.Collection;*

***import*** *java.util.Iterator;*

***import*** *java.util.List;*

***import*** *java.util.ListIterator;*

***public******class*** *MyCollection* ***implements*** *List<Ammunition>, Serializable {*

*/\*\**

*\* Масив Амуніції*

*\*/*

***private*** *Ammunition[] ammunitions;*

*/\*\**

*\** ***@param*** *кількість*

*\* елементів*

*\*/*

***private******int*** *REALSIZE;*

*/\*\**

*\* Клас MyListIterator реалізовує інтерфейс ListIterator<Ammunition>*

*\**

*\** ***@author*** *Lenovo*

*\**

*\*/*

***public******class*** *MyListIterator* ***implements*** *ListIterator<Ammunition> {*

*/\*\**

*\* next наступний елемент previous попередній елемент item теперішшній*

*\* елемент*

*\*/*

*Ammunition next;*

*Ammunition previous;*

*Ammunition item;*

***int*** *index = 0;*

*/\*\**

*\* Конструктор класу, отримує параметр і*

*\**

*\** ***@param*** *i*

*\* ітератор встановлюється в позицію і*

*\*/*

***public*** *MyListIterator(****int*** *i) {*

*index = i;*

*item = ammunitions[index];*

*index++;*

*next = ammunitions[index];*

***if*** *(i == 0) {*

*previous =* ***null****;*

*}* ***else*** *{*

*previous = ammunitions[i - 1];*

*}*

*}*

*/\*\**

*\* конструктор класу ітератор встановлений в позицію і=0*

*\*/*

***public*** *MyListIterator() {*

*index = 0;*

*item = ammunitions[index];*

*index++;*

*next = ammunitions[index];*

*previous =* ***null****;*

*}*

*/\*\**

*\* метод додає до колекції елемент е*

*\*/*

*@Override*

***public******void*** *add(Ammunition e) {*

*item = e;*

***if*** *(index == ammunitions.length) {*

*next =* ***null****;*

*previous = ammunitions[index];*

*index++;*

*}* ***else*** *{*

*index++;*

*previous = ammunitions[index - 1];*

*}*

*MyCollection.****this****.add(e);*

*}*

*/\*\**

*\* метод перевіряє чи є у колекції наступний елемент*

*\*/*

*@Override*

***public******boolean*** *hasNext() {*

***return*** *ammunitions[index - 1] !=* ***null****;*

*}*

*/\*\**

*\* метод перевіряє чи є у колекції попередній елемент*

*\*/*

*@Override*

***public******boolean*** *hasPrevious() {*

***return*** *previous !=* ***null****;*

*}*

*/\*\**

*\* повертає наступний елемент*

*\*/*

*@Override*

***public*** *Ammunition next() {*

*Ammunition it = item;*

*item = next;*

*previous = it;*

***if*** *(index == ammunitions.length) {*

*next =* ***null****;*

*}* ***else*** *{*

*index++;*

*next = ammunitions[index];*

*previous = ammunitions[index - 1];*

*}*

***return*** *it;*

*}*

*/\*\*повертає позицію наступного елемента\*/*

*@Override*

***public******int*** *nextIndex() {*

***return*** *index;*

*}*

*/\*\* повертає попередній елемент \*/*

*@Override*

***public*** *lab8.Ammunition previous() {*

***return*** *previous;*

*}*

*/\*\* повертає позицію попереднього елемента \*/*

*@Override*

***public******int*** *previousIndex() {*

***return*** *index - 1;*

*}*

*/\*\* видаляє теперішній елемент\*/*

*@Override*

***public******void*** *remove() {*

*MyCollection.****this****.remove(index - 1);*

*}*

*/\*\*встановлює на опзицію елемент е\*/*

*@Override*

***public******void*** *set(Ammunition e) {*

*MyCollection.****this****.set(index - 2, e);*

*}*

*}*

*/\*\**

*\* конструктор класу, створює пусту колекцію*

*\*/*

***public*** *MyCollection() {*

*Ammunition[] am =* ***new*** *Ammunition[15];*

*ammunitions = am;*

*REALSIZE = 0;*

*}*

*/\*\**

*\* конструктор класу створює колекцію з одним елементом е*

*\**

*\** ***@param*** *e*

*\*/*

***public*** *MyCollection(Ammunition e) {*

*Ammunition[] am =* ***new*** *Ammunition[15];*

*REALSIZE = 0;*

*am[REALSIZE] = e;*

*REALSIZE = 1;*

*ammunitions = am;*

*}*

*/\*\**

*\* конструктор класу, створює колекцію з масиву елементів am*

*\**

*\** ***@param*** *am*

*\*/*

***public*** *MyCollection(Ammunition[] am) {*

***int*** *i = 0;*

*Ammunition[] ambuf =* ***new*** *Ammunition[15];*

***for*** *(Ammunition ammunition : am) {*

*ambuf[i] = ammunition;*

*i++;*

*}*

*REALSIZE = i;*

*ammunitions = ambuf;*

*ambuf =* ***null****;*

*}*

*/\*\**

*\* Конструктор класу, створює колекцію з колекції типу ArrayList<Ammunition>*

*\**

*\** ***@param*** *arList*

*\*/*

***public*** *MyCollection(ArrayList<Ammunition> arList) {*

*Ammunition[] ambuf =* ***new*** *Ammunition[15];*

***int*** *k = 0;*

***for*** *(Ammunition ammunition : arList) {*

*ambuf[k] = ammunition;*

*k++;*

*}*

*REALSIZE = k;*

*ammunitions = ambuf;*

*}*

*/\*\**

*\* додає до колекції елемент e*

*\*/*

***public******boolean*** *add(Ammunition e) {*

***if*** *(e !=* ***null****) {*

***if*** *(ammunitions.length - 2 < size()) {*

*ammunitions = Arrays.copyOf(ammunitions, ammunitions.length*

*+ (****int****) (ammunitions.length \* 0.3));*

*}*

***if*** *(REALSIZE >= 0) {*

*ammunitions[REALSIZE] = e;*

*REALSIZE++;*

***return******true****;*

*}* ***else***

***return******false****;*

*}* ***else***

***throw******new*** *NullPointerException();*

*}*

*/\*\**

*\* додає до колекції елемент element на позицію index*

*\*/*

*@Override*

***public******void*** *add(****int*** *index, Ammunition element) {*

***if*** *(element ==* ***null****)*

***throw******new*** *NullPointerException();*

***else******if*** *(index < 0 | index > ammunitions.length) {*

***throw******new*** *ArrayIndexOutOfBoundsException();*

*}* ***else******if*** *(ammunitions.length - 2 < size()) {*

*ammunitions = Arrays.copyOf(ammunitions, ammunitions.length*

*+ (****int****) (ammunitions.length \* 0.3));*

*}*

*Arrays.fill(ammunitions, REALSIZE, REALSIZE + 1, element);*

*REALSIZE++;*

*}*

*/\*\**

*\* додає до колекції колекцію с*

*\*/*

*@Override*

***public******boolean*** *addAll(Collection<?* ***extends*** *Ammunition> c) {*

***if*** *(ammunitions.length - 2 < size()) {*

*ammunitions = Arrays.copyOf(ammunitions, ammunitions.length*

*+ (****int****) (ammunitions.length \* 0.3));*

*}*

***int*** *b = size();*

*Object[] am = c.toArray();*

***for*** *(Object object : am) {*

***try*** *{*

*Arrays.fill(ammunitions, object);*

*REALSIZE++;*

*}* ***catch*** *(NullPointerException e) {*

*}*

*}*

***if*** *(size() - b == 0)*

***return******false****;*

***else***

***return******true****;*

*}*

*/\*\**

*\* додає до колекції колекцію с з позиції i*

*\*/*

*@Override*

***public******boolean*** *addAll(****int*** *index, Collection<?* ***extends*** *Ammunition> c) {*

***if*** *(ammunitions.length - 2 < size()) {*

*ammunitions = Arrays.copyOf(ammunitions, ammunitions.length*

*+ (****int****) (ammunitions.length \* 0.3));*

*}*

***int*** *b = size();*

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

*Object[] am = c.toArray();*

***for*** *(Object object : am) {*

***try*** *{*

*Arrays.fill(ammunitions, index, index + am.length, object);*

*REALSIZE++;*

*}* ***catch*** *(NullPointerException e) {*

*}*

*}*

***if*** *(size() - b == 0)*

***return******false****;*

***else***

***return******true****;*

*}*

*/\*\**

*\* видаляє всі елементи колекції*

*\*/*

*@Override*

***public******void*** *clear() {*

*Ammunition[] am =* ***new*** *Ammunition[15];*

*ammunitions = am;*

*}*

*/\*\**

*\* перевіряє чи об'єкт e належить колекції*

*\*/*

*@Override*

***public******boolean*** *contains(Object e) {*

***if*** *(e ==* ***null****)*

***throw******new*** *NullPointerException();*

***else*** *{*

***boolean*** *q =* ***false****;*

***for*** *(Ammunition ammunition : ammunitions) {*

***if*** *(ammunition.equals(e)) {*

*q =* ***true****;*

*}*

*}*

***return*** *q;*

*}*

*}*

*/\*\**

*\* перевіряє чи колекція с належить колекції*

*\*/*

*@Override*

***public******boolean*** *containsAll(Collection<?> c) {*

***boolean*** *q =* ***true****;*

***int*** *i = 0;*

*Object[] amm = c.toArray();*

***for*** *(****int*** *j = 0; j < amm.length; j++) {*

*i = 0;*

*q =* ***true****;*

***while*** *((ammunitions[i] !=* ***null****) & (q)) {*

***if*** *(ammunitions[i].equals(amm[j])) {*

*q =* ***false****;*

*i++;*

*amm[j] =* ***null****;*

*}*

*}*

*}*

*q =* ***true****;*

***for*** *(Object ammunition : amm) {*

***if*** *(ammunition ==* ***null****) {*

*q =* ***true****;*

*}* ***else***

*q =* ***false****;*

*}*

***return*** *q;*

*}*

*/\*\**

*\* повертає елемент колекції за позицією index*

*\*/*

*@Override*

***public*** *Ammunition get(****int*** *index) {*

***if*** *(index < 0 | index > ammunitions.length)*

***throw******new*** *IndexOutOfBoundsException();*

***return*** *ammunitions[index];*

*}*

*/\*\**

*\* повертає позицію першого об'єкта е який зустрівся в колекції*

*\*/*

*@Override*

***public******int*** *indexOf(Object e) {*

***boolean*** *q =* ***true****;*

***int*** *positon = 0;*

*ListIterator<Ammunition> itr = listIterator();*

***if*** *(q)*

***while*** *((itr.hasNext())) {*

*Ammunition ammunition = itr.next();*

***if*** *(e.equals(ammunition)) {*

*q =* ***false****;*

*positon = itr.nextIndex() - 2;*

*}*

*}*

***if*** *(q) {*

***return*** *-1;*

*}* ***else***

***return*** *positon;*

*}*

*/\*\**

*\* преевіряє чи колекція пуста*

*\*/*

*@Override*

***public******boolean*** *isEmpty() {*

***boolean*** *q =* ***true****;*

***for*** *(Ammunition ammunition : ammunitions) {*

***if*** *(!(ammunition ==* ***null****)) {*

*q =* ***false****;*

*;*

*}*

*}*

***return*** *q;*

*}*

*/\*\**

*\* повертає ітератор*

*\*/*

*@Override*

***public*** *Iterator<Ammunition> iterator() {*

*MyListIterator itr =* ***new*** *MyListIterator();*

***return*** *itr;*

*}*

*/\*\**

*\* повертає останнє входження об'єкта е в колекцію*

*\*/*

*@Override*

***public******int*** *lastIndexOf(Object e) {*

***boolean*** *q =* ***true****;*

***int*** *i = REALSIZE;*

***while*** *((q) & (ammunitions[i] !=* ***null****)) {*

***if*** *(ammunitions[i].equals(e)) {*

*q =* ***false****;*

*}* ***else***

*i--;*

*}*

***if*** *(i == 0)*

***return*** *-1;*

***else***

***return*** *i - 1;*

*}*

*/\*\**

*\* повертає ListIterator для колекції*

*\*/*

*@Override*

***public*** *ListIterator<Ammunition> listIterator() {*

*MyListIterator itr =* ***new*** *MyListIterator();*

***return*** *itr;*

*}*

*/\*\**

*\* повертає ListIterator для колекції з позиції index*

*\*/*

*@Override*

***public*** *ListIterator<Ammunition> listIterator(****int*** *index) {*

*MyListIterator itr =* ***new*** *MyListIterator(index);*

***return*** *itr;*

*}*

*/\*\**

*\* видаляє обєкт e з колекції*

*\*/*

*@Override*

***public******boolean*** *remove(Object e) {*

***if*** *(e ==* ***null****)*

***throw******new*** *NullPointerException();*

***else*** *{*

***int*** *i = 0;*

***if*** *(indexOf(e) != -1) {*

*Ammunition[] am =* ***new*** *Ammunition[ammunitions.length];*

*ListIterator<Ammunition> itr = listIterator();*

***while*** *(itr.hasNext()) {*

*Ammunition ammunition = itr.next();*

***if*** *(!ammunition.equals(e)) {*

*am[i] = ammunition;*

*i++;*

*}*

*}*

*ammunitions =* ***null****;*

*ammunitions = am;*

*REALSIZE--;*

***return******true****;*

*}* ***else***

***return******false****;*

*}*

*}*

*/\*\**

*\* видаляє об'єкт за позицією index*

*\*/*

*@Override*

***public*** *Ammunition remove(****int*** *index) {*

***if*** *(index < 0 | index > ammunitions.length)*

***throw******new*** *IndexOutOfBoundsException();*

***else*** *{*

*Ammunition[] newam =* ***new*** *Ammunition[ammunitions.length];*

***int*** *k = 0;*

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

***if*** *(i != index) {*

*newam[k] = ammunitions[i];*

*k++;*

*}*

*}*

*Ammunition buf = ammunitions[index];*

*ammunitions =* ***null****;*

*ammunitions = newam;*

*REALSIZE--;*

***return*** *buf;*

*}*

*}*

*/\*\**

*\* видаляє з колекції всі обєкти колекції с*

*\*/*

*@Override*

***public******boolean*** *removeAll(Collection<?> c) {*

*Object[] am = c.toArray();*

*ListIterator<Ammunition> itr1 = listIterator();*

***while*** *(itr1.hasNext()) {*

*Ammunition ammunition = itr1.next();*

***for*** *(Object object : am) {*

***if*** *(ammunition.equals(object)) {*

*remove(ammunition);*

*}*

*}*

*}*

***return******true****;*

*}*

*/\*\**

*\* видаляє з колекції всі об'єкти що не належить до колекції c*

*\*/*

*@Override*

***public******boolean*** *retainAll(Collection<?> c) {*

*Object[] am = c.toArray();*

*Ammunition[] newam =* ***new*** *Ammunition[ammunitions.length];*

***int*** *k = 0;*

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

***if*** *(indexOf(am[i]) != -1) {*

*newam[k] = (Ammunition) am[i];*

*k++;*

*}*

*}*

***if*** *(newam[0] !=* ***null****) {*

*ammunitions =* ***null****;*

*ammunitions = newam;*

***return******true****;*

*}* ***else***

***return******false****;*

*}*

*/\*\**

*\* встановлює елемент element на позицію index*

*\*/*

*@Override*

***public*** *Ammunition set(****int*** *index, Ammunition element) {*

***if*** *((index > 0) & (index < ammunitions.length))*

*ammunitions[index] = element;*

***return*** *element;*

*}*

*/\*\**

*\* повертає кількість елементів колекції*

*\*/*

*@Override*

***public******int*** *size() {*

***return*** *REALSIZE;*

*}*

*/\*\**

*\* повертає список ArrayList колекції з позиції fromIndex до позиції toIndex*

*\*/*

*@Override*

***public*** *ArrayList<Ammunition> subList(****int*** *fromIndex,* ***int*** *toIndex) {*

*ArrayList<Ammunition> am =* ***new*** *ArrayList<Ammunition>();*

***for*** *(****int*** *i = fromIndex; i < toIndex; i++) {*

*am.add(ammunitions[i]);*

*}*

***return*** *am;*

*}*

*/\*\**

*\* повертає масив з елементів колекції*

*\*/*

*@Override*

***public*** *Ammunition[] toArray() {*

***int*** *i = 0;*

*Ammunition[] am =* ***new*** *Ammunition[size()];*

*ListIterator<Ammunition> itr = listIterator();*

***while*** *(itr.hasNext()) {*

*Ammunition ammunition = itr.next();*

*am[i] = ammunition;*

*i++;*

*}*

***return*** *am;*

*}*

*/\*\**

*\* повертає масив з елементів колекції з елементами клас яких співпав з*

*\* елементами масиву array*

*\*/*

*@SuppressWarnings({ "hiding", "unchecked", "null" })*

*@Override*

***public*** *<Ammunition> Ammunition[] toArray(Ammunition[] array) {*

***int*** *reconciliation\_counter = 0;*

***for*** *(lab8.Ammunition ammunition : ammunitions) {*

***for*** *(Ammunition ams : array) {*

***if*** *(ammunition.getClass().equals(ams.getClass())) {*

*reconciliation\_counter++;*

*}*

*}*

*}*

***if*** *(reconciliation\_counter == array.length) {*

***return*** *array;*

*}* ***else*** *{*

*Ammunition[] newam =* ***null****;*

***int*** *k = 0;*

***for*** *(lab8.Ammunition ammunition : ammunitions) {*

***for*** *(Ammunition ams : array) {*

***if*** *(ammunition.getClass().equals(ams.getClass())) {*

*newam[k] = (Ammunition) ammunition;*

*}*

*}*

*}*

***return*** *newam;*

*}*

*}*

*/\*\**

*\* перевіряє чи підходить даний набір амуніції для одного лицаря*

*\*/*

***public******boolean*** *isCorrectAmmunition() {*

***boolean*** *bul =* ***true****;*

***int*** *i = 0;*

*ListIterator<Ammunition> itr = listIterator();*

***if*** *(size() <= 4) {*

***while*** *(itr.hasNext()) {*

*Ammunition ammunition = itr.next();*

*ListIterator<Ammunition> itr1 = listIterator();*

*i = 0;*

***while*** *(itr1.hasNext()) {*

*Ammunition ammunition2 = itr1.next();*

***if*** *((ammunition.getClass()).equals(ammunition2.getClass())) {*

*i++;*

*}*

*}*

***if*** *(i >= 2) {*

*bul =* ***false****;*

*}*

*}*

***return*** *bul;*

*}* ***else***

***return******false****;*

*}*

*}*

Клас TestWorkWithFile

***package*** *lab8;*

***import******static*** *org.junit.Assert.\*;*

***import*** *org.junit.Test;*

*/\*\**

*\* Тестовий клас для класу WorkWithFile*

*\** ***@author*** *Lenovo*

*\**

*\*/*

***public******class*** *TestWorkWithFile {*

*/\*\**

*\* Тест для методу WriteMyCollection()*

*\* Тест для методу ReadMyCollection()*

*\*/*

*@Test*

***public******void*** *testReadWriteMyCollection() {*

*Ammunition am=****null****;*

***try*** *{*

*am =* ***new*** *Ammunition("name", 10, 20);*

*}* ***catch*** *(MyException e) {*

*e.printStackTrace();*

*}*

*MyCollection col =* ***new*** *MyCollection(am);*

*WorkWithFile w =* ***new*** *WorkWithFile(col);*

*w.WriteMyCollection();*

*assertTrue(w.ReadMyCollection().toArray()[0].getInformation().equals(am.getInformation()));*

*}*

*/\*\**

*\* Тест для методу WriteAmmunitions()*

*\* Тест для методу ReadAmmunition()*

*\*/*

*@Test*

***public******void*** *testReadWriteAmmunitions() {*

*Ammunition am=****null****;*

*Ammunition am1 =* ***null****;*

***try*** *{*

*am =* ***new*** *Ammunition("name", 10, 20);*

*am1 =* ***new*** *Ammunition("name2", 40, 50);*

*}* ***catch*** *(MyException e) {*

*e.printStackTrace();*

*}*

*Ammunition [] mas =* ***new*** *Ammunition[2];*

*mas[0]=am;*

*mas[1]=am1;*

*WorkWithFile w =* ***new*** *WorkWithFile(mas);*

*w.WriteAmmunitions();*

***int*** *i=0;*

***for*** *(Ammunition ammunition : w.ReadAmmunition()) {*

*assertTrue(ammunition.getInformation().equals(mas[i].getInformation()));*

*i++;*

*}*

*}*

*/\*\**

*\* Тест для методу WriteInformation()*

*\* Тест для методу ReadInformation()*

*\*/*

*@Test*

***public******void*** *testWriteInformation() {*

*Ammunition am =* ***null****;*

***try*** *{*

*am =* ***new*** *Ammunition("name", 10, 20);*

*}* ***catch*** *(MyException e) {*

*e.printStackTrace();*

*}*

*MyCollection col =* ***new*** *MyCollection(am);*

*WorkWithFile w =* ***new*** *WorkWithFile(col);*

*w.WriteInformation();*

*assertTrue(w.ReadInformation().equals(am.getInformation()));*

*}*

*}*

Висновки: Я навчився використовувати потоки в мові програмування Java як засіб роботи з файлами.