# **Національнийтехнічний університет України**

# **«Київський політехнічний інститут»**

# **Факультет інформатики та обчислювальної техніки**

# **Кафедра обчислювальноїтехніки**

# **Лабораторна робота №7**

### **З програмування**

**виконав студент I курсу**

**група ІО-01**

**Шийка Владислав Володимирович**

|  |  |  |
| --- | --- | --- |
| **Тема:** | | Робота з колекціями в мові програмування Java. |
| **Мета:** | Здобуття навичок у створенні власних та використанні стандартних колекцій в мові програмування Java. | | |

Номер залікової : 129 mod 2 =1

Список

**Завдання:**Створитиклас, що описує типізовану колекцію (типом колекції є клас з лабораторноїроботи №5) із заданною внутрішньою структурою (п.2), що складається не менше ніж з 3 конструкторів (1 – порожній, 2 – в який передається об’єкт, 3 – в який передається стандартна колекція об’єктів, наприклад, ArrayList), а також методів додавання, видалення та заміщення об’єкту(ів) в колекції. При додаванні об’єкту до колекції виконувати перевірку існування його в колекції. Забороняється використання, в якості внутрішньої структури, стандартних колекцій. Всі початкові дані задаються у виконавчомуметоді.

**Лістинг класу колекції**

**package** lab5;

**import** java.util.ArrayList;

**public** **class** PlaneList {

**private** Component current,next,header;

**private** **int** size=0;

PlaneList(){

**this**.header=**new** Component();

}

PlaneList(Component c){

header=c;

}

PlaneList(Plane pl){

**this**();

addToEnd(pl,getSize());

}

PlaneList(ArrayList<Plane> list){

**this**();

addToEnd(list);

}

**public** **void** addPlane(Plane pl){

addToEnd(pl,getSize());

}

**public** **void** addToEnd(Plane pl,**int** index){

**if**(index<0){

System.*out*.println("Bad Index");

}**else**{

current=next=header;

moveTo(index,current,next);

add(pl,current,next);

}

}

**public** **void** addToEnd(ArrayList<Plane> list){

Plane[] array=**new** Plane[list.size()];

list.toArray(array);

**for**(Plane p:list){

addToEnd(p,getSize());

}

}

**public** **void** addToEnd(Plane[] array){

**for**(Plane p:array){

addToEnd(p,getSize());

}

}

**public** **void** add(Plane pl,Component previous,Component next){

Component c=**new** Component();

c.setPlane(pl);

previous.setNext(c);

c.setNext(next);

size++;

}

**public** **boolean** move(Plane p,**int** index){

**if**(0<index){

System.*out*.println("Bad Index");

**return** **false**;

}**else**{

current=next=header;

moveTo(index,current,next);

Component c=**new** Component();

c.setNext(next.getNext());

current.setNext(c);

**return** **true**;

}

}

**public** **void** move(Plane[] array,**int** index){

**if**(0<index){

System.*out*.println("Bad Index");

}**else**{

**for**(Plane p:array){

**if**(move(p,index)==**true**){

move(p,index);

index++;

}

}

}

}

**public** **void** remove(Component current,Component next){

current=next.getNext();

size--;

}

**public** **void** remove(Component current,Component next,**int** number){

remove(current,next);

size-=number;

}

**public** **boolean** hasNext(Component c){

**return** c.getNext()!=**null**;

}

**public** Component getCurrent(){

**return** current;

}

**public** **boolean** next(){

**if**(hasNext(current)==**false**){

**return** **false**;

}**else**{

current=next;

next=next.getNext();

**return** **true**;

}

}

**public** Component search(**int** index){

**if** (index<0){

System.*out*.println("Bad Index");

**return** **null**;

}**else**{

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

next();

}

**return** getCurrent();

}

}

**public** **boolean** moveTo(**int** index,Component previous,Component next){

**if**(0<index){

System.*out*.println("Bad Index");

**return** **false**;

}**else**{

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

next();

}

**return** **true**;

}

}

**public** **boolean** check(Plane pl){

**for**(**int** i=0;i<getSize();i++){

current=next=header;

moveTo(i,current,next);

**if**(pl.equals((Plane)getCurrent().getPlane())==**true**){

**return** **false**;

}

}

**return** **true**;

}

**public** **void** setSize(**int** size) {

**this**.size = size;

}

**public** **int** getSize() {

**return** size;

}

**public** **void** print(Component c){

c.printPlane();

}

**public** **void** printAll(){

**for**(**int** i=0;i<getSize();i++){

next();

print(getCurrent());

}

}

}

**Лістинг классу елементу списку**

**package** lab5;

**public** **class** Component {

**private** Plane pl;

**private** Component next;

Component(){

**this**.pl=**null**;

**this**.next=**null**;

}

Component(Plane pl){

setPlane(pl);

}

**public** **void** setNext(Component c){

**this**.next=c;

}

**public** Component getNext(){

**return** next;

}

**public** **void** setPlane(Plane pl){

**this**.pl=pl;

}

**public** Plane getPlane(){

**return** pl;

}

**public** **void** printPlane(){

pl.printAll();

}

}

**Виконавчий клас**

**package** lab5;

**import** java.util.ArrayList;

**public** **class** ExeLab5 {

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

Plane p1 = **new** Plane();

Plane p2 = **new** Plane(123,0,0,0,0,"","");

ArrayList<Plane> array = **new** ArrayList<Plane>();

array.add(p1);

array.add(p2);

PlaneList list = **new** PlaneList(array);

System.*out*.println(list.getSize());

list.printAll();

}

}

**Опис:** Клас Component має поля типу Plane та типу Component ,pl та next відповідно методи доступу до цих полів. Component використовується як ссилка на наступний елемент списку. Клас List містить методи створення колекції, методи додавання, заміни, видалення елементів з колекції, методи перевірки наявності елемента в колекції та наявності наступного елемента після заданого, також містить допоміжні методи такі як search,next,getSize,print,check,moveTo.Клас ExecLab5 має виконавчий метод у якому задаються початкові дані.

**Висновки:** під час виконання роботи я вдосконалив свої навички у роботі зі створення колекцій (створення методів додавання, видалення , заміни елементів колекції) і використанні стандартних колекцій java.