

|  |  |
| --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  федеральное государственное бюджетное образовательное учреждение высшего образования  **«МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ ГОСУДАРСТВЕННЫЙ**  **ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ)»** |  |

**Кафедра «Высшая математика»**

**Лабораторная работа № 6**

по дисциплине

«Структуры и алгоритмы обработки данных»

на тему:

**«**Работа со списками**»**

**Выполнил:**

Учебная группа: 1бПМ   
ФИО: Кузнецов А.-С. О.

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Руководитель лабораторной работы:**

Должность: старший преподаватель

Звание: б/з

ФИО: Кутейников И. А.

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Цель: 1 Реализовать в Java класс в соответствии со своим вариантом.

Предусмотреть не менее 3 параметров, одним из которых является

объект другого класса, 2 методов и 2 конструкторов (включая

конструктор по умолчанию). Предусмотреть счетчик экземпляров

классов.

Предусмотреть для классов, являющихся полями не менее 2

параметров, 2 методов и 2 конструкторов (включая конструктор по

умолчанию).

2 Реализовать хранение объектов классов в виде списка.

Предусмотреть следующие операции над списком:

1) Добавления элемента (доп. нескольких элементов) в конец или в

определенное место списка;

2) Удаление элемента (доп. нескольких элементов) в конце или в

определенном месте списка;

3) Изменение определенного элемента списка;

4) Поиск и вывод номера и информации введенного элемента по

разным полям;

5) Вывод на экран всех объектов списка или определенного

элемента списка;

6) Вывод числа элементов в списке.

Класс дерево.

Код:

class Tree{  
 String name;  
 int age;  
 Leaves leaves;  
 int treeCounter;  
 public Tree(){  
 treeCounter++;  
 }  
 public Tree(String name, int age, Leaves leaves){  
 this.name=name;  
 this.age=age;  
 this.leaves=leaves;  
 treeCounter++;  
 }  
 public void treeinfo(){  
 *out*.println("Name tree: "+name+' '+"Age tree: "+age);  
 *out*.println("Inf Leaves: "+leaves.getInfoLeavs());  
 }  
 public String getName(){  
 return name;  
 }  
 public int getAge(){  
 return age;  
 }  
 public String getColor(){  
 return leaves.color;  
 }  
 public String getShape(){  
 return leaves.shape;  
 }  
}  
class Leaves{  
 String color;  
 String shape;  
 public Leaves(){  
 color="green";  
 shape="romb";  
 }  
 public Leaves(String color, String shape){  
 this.color=color;  
 this.shape=shape;  
 }  
 public String getInfoLeavs(){  
 return "color:"+color+' '+"shape:"+shape;  
 }  
}  
public class Main{  
 static ArrayList<Tree>*TreeList*=new ArrayList<>();  
 public static void main(String []args){  
 Scanner in= new Scanner(System.*in*);  
 boolean menu=true;  
 while(menu) {  
 *out*.println("Menu");  
 *out*.println("1.add tree");  
 *out*.println("2.remove tree");  
 *out*.println("3.change tree");  
 *out*.println("4.find tree");  
 *out*.println("5.all tree");  
 *out*.println("6.all tree num");  
 *out*.println("7.exit");  
 *out*.print("enter ans num: ");  
 int ans;  
 ans = in.nextInt();  
 switch (ans){  
 case 1:  
 *addTree*();  
 break;  
 case 2:  
 *removeTree*();  
 break;  
 case 3:  
 *modTree*();  
 break;  
 case 4:  
 *findTree*();  
 break;  
 case 5:  
 *disinfo*();  
 break;  
 case 6:  
 *numerTree*();  
 break;  
 case 7:  
 menu=false;  
 break;  
 }  
 }  
 }  
 public static void addTree(){  
 Scanner in= new Scanner(System.*in*);  
 *out*.println("Enter name tree: ");  
 String name=in.next();  
 *out*.println("Enter age tree: ");  
 int age= in.nextInt();  
 *out*.println("Do you want take tree inf?");  
 *out*.println("1-Yes/2-No");  
 int leavm= in.nextInt();  
 if(leavm==1){  
 *out*.println("enter color: ");  
 String color =in.next();  
 *out*.println("enter shape:");  
 String shape=in.next();  
 Tree newTree=new Tree(name, age,new Leaves(color,shape));  
 *TreeList*.add(newTree);  
 *out*.println("Tree in list");  
 }  
 else {  
 Tree newTree=new Tree(name,age, new Leaves());  
 *TreeList*.add(newTree);  
 *out*.println("Tree in list");  
 }  
 }  
 public static void removeTree(){  
 Scanner in = new Scanner(System.*in*);  
 *out*.println("удалить одно или несколько деревьев"+'\n'+"1-Одно 2-Несколько");  
 int numM2= in.nextInt();  
 if (numM2==1) {  
 *out*.println("enter num:");  
 int index = in.nextInt();  
 if (index >= 0 && index < *TreeList*.size()) {  
 *TreeList*.remove(index);  
 *out*.println("Tree are delete");  
 } else {  
 *out*.println("think again");  
 }  
 }  
 else {  
 *out*.println("Введите диапазон");  
 *out*.println("от:");  
 int diap1= in.nextInt();  
 *out*.println("до:");  
 int diap2= in.nextInt();  
 for (int i12 = diap1;i12<=diap2;i12++){  
 *TreeList*.remove(i12);  
 }  
 *out*.println("trees are deleted");  
 }  
 }  
 public static void modTree(){  
 Scanner in= new Scanner(System.*in*);  
 *out*.println("enter the index tree: ");  
 int index=in.nextInt();  
 *out*.println("vse der or no"+'\n'+"1-vse 2-chast");  
 int menush=in.nextInt();  
 if(menush==1) {  
 if (index >= 0 && index < *TreeList*.size()) {  
 *out*.println("enter new name: ");  
 String name = in.next();  
 *out*.println("enter new age: ");  
 int age = in.nextInt();  
 *out*.println("Do you want mod leaves inf?");  
 *out*.println("1-Yes/2-No");  
 int leavm = in.nextInt();  
 if (leavm == 1) {  
 *out*.println("enter color: ");  
 String color = in.next();  
 *out*.println("enter shape:");  
 String shape = in.next();  
 Tree Tree1 = new Tree(name, age, new Leaves(color, shape));  
 *TreeList*.set(index, Tree1);  
 *out*.println("new Tree in list");  
 } else {  
 Tree Tree1 = new Tree(name, age, new Leaves());  
 *TreeList*.set(index, Tree1);  
 *out*.println("new Tree in list");  
 }  
 }  
 }  
 else{  
 boolean R1=true;  
 while (R1){  
 *out*.println("type");  
 *out*.println("age");  
 *out*.println("color");  
 *out*.println("shape");  
 *out*.println("exit");  
 *out*.println("take men p");  
 int localans=in.nextInt();  
 switch (localans){  
 case 1:  
 *out*.println("take new name");  
 String newname=in.next();  
 *TreeList*.get(index).name=newname;  
 break;  
 case 2:  
 *out*.println("take new age");  
 int newage=in.nextInt();  
 *TreeList*.get(index).age=newage;  
 break;  
 case 3:  
 *out*.println("take new color");  
 String newcolor=in.next();  
 *TreeList*.get(index).leaves.color=newcolor;  
 break;  
 case 4:  
 *out*.println("take new shape");  
 String newshape=in.next();  
 *TreeList*.get(index).leaves.shape=newshape;  
 break;  
 case 5:  
 R1=false;  
 break;  
 }  
 }  
 }  
 }  
 public static void findTree(){  
 Scanner in= new Scanner(System.*in*);  
 boolean R=true;  
 while (R) {  
 *out*.println("type");  
 *out*.println("age");  
 *out*.println("color");  
 *out*.println("shape");  
 *out*.println("exit");  
 *out*.println("take num menu ");  
 int localans= in.nextInt();  
 switch (localans) {  
 case 1:  
 *out*.println("enter the name of tree what you want to find");  
 String nameFS = in.next();  
 int numTree = -1;  
 for (Tree tree : *TreeList*) {  
 numTree++;  
 if (tree.getName().equalsIgnoreCase(nameFS)) {  
 *out*.println("Num of tree: " + numTree + "; Name tree: " + tree.name + "; Age tree: " + tree.age + "; tree leaves: " + tree.leaves.getInfoLeavs());  
 return;  
 }  
 }  
 break;  
 case 2:  
 *out*.println("enter the age of tree what you want to find");  
 int ageFS = in.nextInt();  
 int numTree1 = -1;  
 for (Tree tree : *TreeList*) {  
 numTree1++;  
 if (tree.age==ageFS) {  
 *out*.println("Num of tree: " + numTree1 + "; Name tree: " + tree.name + "; Age tree: " + tree.age + "; tree leaves: " + tree.leaves.getInfoLeavs());  
 return;  
 }  
 }  
 break;  
 case 3:  
 *out*.println("enter the color of tree what you want to find");  
 String colorFS = in.next();  
 int numTree2 = -1;  
 for (Tree tree : *TreeList*) {  
 numTree2++;  
 if (tree.getColor().equalsIgnoreCase(colorFS)) {  
 *out*.println("Num of tree: " + numTree2 + "; Name tree: " + tree.name + "; Age tree: " + tree.age + "; tree leaves: " + tree.leaves.getInfoLeavs());  
 return;  
 }  
 }  
 break;  
 case 4:  
 *out*.println("enter the shape of tree what you want to find");  
 String shapeFS = in.next();  
 int numTree3 = -1;  
 for (Tree tree : *TreeList*) {  
 numTree3++;  
 if (tree.getShape().equalsIgnoreCase(shapeFS)) {  
 *out*.println("Num of tree: " + numTree3 + "; Name tree: " + tree.name + "; Age tree: " + tree.age + "; tree leaves: " + tree.leaves.getInfoLeavs());  
 return;  
 }  
 }  
 break;  
 case 5:  
 R=false;  
 break;  
 }  
  
 }  
 }  
 public static void disinfo(){  
 *out*.println("All trees:");  
 for (Tree tree: *TreeList*){  
 tree.treeinfo();  
 }  
 }  
 public static void numerTree(){  
 *out*.println("num tree in list: "+*TreeList*.size());  
 }  
}

Результат:

Menu

1.add tree

2.remove tree

3.change tree

4.find tree

5.all tree

6.all tree num

7.exit

enter ans num: 1

Enter name tree:

oak

Enter age tree:

23

Do you want take tree inf?

1-Yes/2-No

1

enter color:

red

enter shape:

tyb

Tree in list

enter ans num: 1

Enter name tree:

lipa

Enter age tree:

43

Do you want take tree inf?

1-Yes/2-No

2

Tree in list

enter ans num: 3

enter the index tree:

1

vse der or no

1-vse 2-chast

1

enter new name:

pipa

enter new age:

23

Do you want mod leaves inf?

1-Yes/2-No

2

new Tree in list

enter ans num: 4

type

age

color

shape

exit

take num menu

1

enter the name of tree what you want to find

oak

Num of tree: 0; Name tree: oak; Age tree: 23; tree leaves: color:red shape:tyb

enter ans num: 5

All trees:

Name tree: oak Age tree: 23

Inf Leaves: color:red shape:tyb

Name tree: pipa Age tree: 23

Inf Leaves: color:green shape:romb

enter ans num: 6

num tree in list: 2

Вывод:

В ходе лабораторной работы было получено понимание того, как работать со списками, а именно

1. Как создавать отдельные классы
2. Как создавать подклассы
3. Как работать со списками типа List

Также было получено понимание того, как оптимальнее создавать код.