Институт информационных технологий и управления в технических системах

Кафедра информационных технологий и компьютерных систем

ОТЧЕТ

По лабораторной работе №2

«Манипулирование объектами разных классов в Java-программе. Программирование взаимодействия простых объектов»

По дисциплине «Программирование. Языки высокого уровня»

Выполнил:

студент группы ИВТ/б-13о

Мосенков В.А

Проверил:

ст. преподаватель Владимирова Е.С

Севастополь

2019

**Цель работы:**

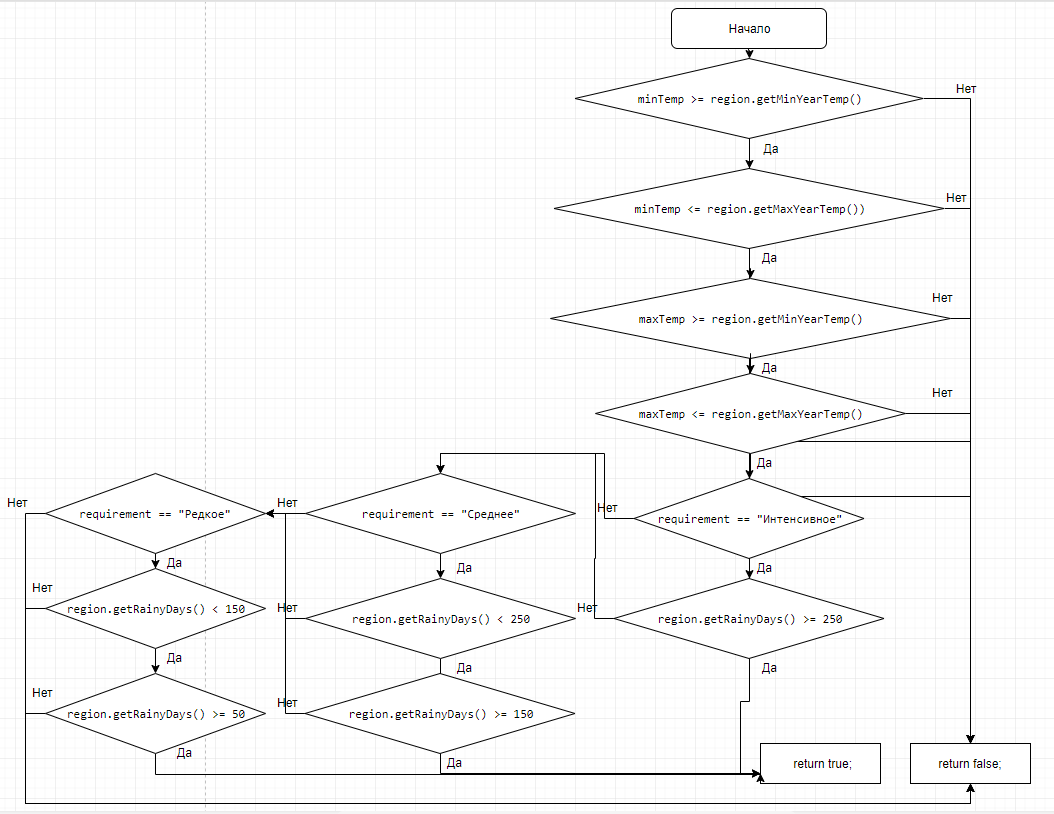
Целью данной работы является закрепление навыков разработки классов-шаблонов для создания объектов в соответствии с принципом инкапсуляции данных и методов (подпрограмм) для работы с этими данными, а также создания и использования простых объектов в программе, организующей взаимодействие объектов разных классов.

**Постановка задачи:**

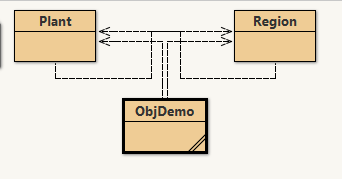
Вариант 14

Разработать программу, реализующую взаимодействие двух объектов, заданных вариантом задания. Для создания объектов разработать соответствующие классы. Методы объектов отладить, используя возможности панели объектов BlueJ.

**Алгоритм accord():**



**Структура проекта:**



**Текст классов:**

Plant:

public class Plant implements Comparable <Plant>

{

private final static String PLANT\_FORMAT\_STRING = "Растение: %10s,%10s.Требования к увлажнению - %10s. Минимальная температура воздуха - %5d. Максимальная температура воздуха - %5d";

private String name, type, requirement;

private int minTemp,maxTemp;

/\* Конструкторы \*/

public Plant(){ //конструктор без параметров

this.name = "";

this.type = "";

this.requirement = "";

this.minTemp = 0;

this.maxTemp = 0;

}

public Plant(String name, String type, String requirement, int minTemp, int maxTemp){ // конструктор с параметрами

this.name = name;

this.type = type;

this.requirement = requirement;

this.minTemp = minTemp;

this.maxTemp = maxTemp;

}

public Plant(Plant plant){ //клон объекта

this.name = plant.getName();

this.type = plant.getType();

this.requirement = plant.getRequirement();

this.minTemp = plant.getMinTemp();

this.maxTemp = plant.getMaxTemp();

}

/\* Конец конструкторов \*/

/\* Методы \*/

public double averageTemp() { // средняя температура

double result = (this.minTemp + this.maxTemp)/2;

return result;

}

public boolean accord(Region region){

boolean tempResult = (minTemp >= region.getMinYearTemp() && minTemp <= region.getMaxYearTemp()) &&

(maxTemp >= region.getMinYearTemp() && maxTemp <= region.getMaxYearTemp());

boolean rainResult = (requirement == "Интенсивное" && region.getRainyDays() >= 250) ||

(requirement == "Среднее" && region.getRainyDays() < 250 && region.getRainyDays() >= 150) ||

(requirement == "Редкое" && region.getRainyDays() < 150 && region.getRainyDays() >= 50);

return (tempResult && rainResult);

}

@Override

public boolean equals(Object ob){

if(ob == this) return true;

if(ob == null) return false;

if(getClass()!=ob.getClass()) return false;

Plant plant = (Plant) ob;

return name.equals(plant.getName()) &&

type.equals(plant.getType()) &&

requirement.equals(plant.getRequirement()) &&

(minTemp == plant.getMinTemp()) &&

(maxTemp == plant.getMaxTemp());

}

public int hashCode(){

int result = 17;

result = 37 \* result + (this.name == null ? 0 : this.name.hashCode());

result = 37 \* result + (this.type == null ? 0 : this.type.hashCode());

result = 37 \* result + (this.requirement == null ? 0 : this.requirement.hashCode());

result = 37 \* result + (int)this.minTemp;

result = 37 \* result + (int)this.maxTemp;

return result;

}

public String toString(){

return String.format(PLANT\_FORMAT\_STRING,name,type,requirement,minTemp,maxTemp);

}

public int compareTo(Plant plant){

if(this.averageTemp() < plant.averageTemp()) return -1;

if((this.averageTemp() == plant.averageTemp()) && (name.compareTo(plant.getName()) < 0)) return -1;

if((this.averageTemp() == plant.averageTemp()) && (name.compareTo(plant.getName()) == 0) && (type.compareTo(plant.getType()) < 0)) return -1;

if((this.averageTemp() == plant.averageTemp()) && (name.compareTo(plant.getName()) == 0) && (type.compareTo(plant.getType()) == 0) && (requirement.compareTo(plant.getRequirement()) < 0)) return -1;

if((this.averageTemp() == plant.averageTemp()) && (name.compareTo(plant.getName()) == 0) && (type.compareTo(plant.getType()) == 0) && (requirement.compareTo(plant.getRequirement()) == 0)) return 0;

else return 1;

}

/\* Конец методов \*/

/\* Геттеры \*/

public String getName(){

return this.name;

}

public String getType(){

return this.type;

}

public String getRequirement(){

return this.requirement;

}

public int getMinTemp(){

return this.minTemp;

}

public int getMaxTemp(){

return this.maxTemp;

}

/\* Конец геттеров \*/

/\* Сеттеры \*/

public void setName(String name){

this.name = name;

}

public void setRequirement(String requirement){

this.requirement = requirement;

}

public void setType(String type){

this.type = type;

}

public void setMinTemp(int temp){

this.minTemp = temp;

}

public void setMaxTemp(int temp){

this.maxTemp = temp;

}

/\* Конец сеттеров \*/

}

Region:

public class Region implements Comparable <Region>

{

private final static String REGION\_FORMAT\_STRING = "Регион: %10s . Минимальная температура за год %3d . Максимальная температура за год %3d . Количество дождливых дней в году %3d";

private String name;

private int minYearTemp,maxYearTemp,rainyDays;

public Region(){

this.name = "";

this.minYearTemp = 0;

this.maxYearTemp = 0;

this.rainyDays = 0;

}

public Region(String name, int minTemp, int maxTemp, int rainDays){

this.name = name;

this.minYearTemp = minTemp;

this.maxYearTemp = maxTemp;

this.rainyDays = rainDays;

}

public Region(Region r){

this.name = r.getName();

this.minYearTemp = r.getMinYearTemp();

this.maxYearTemp = r.getMaxYearTemp();

this.rainyDays = r.getRainyDays();

}

@Override

public boolean equals(Object ob){

if(ob==this) return true;

if(ob==null) return false;

if(getClass()!=ob.getClass()) return false;

Region r = (Region) ob;

return (name.equals(r.getName())) &&

(minYearTemp == r.getMinYearTemp()) &&

(maxYearTemp == r.getMaxYearTemp()) &&

(rainyDays == r.getRainyDays());

}

public int hashCode(){

int result = 17;

result = 37 \* result + (this.name == null ? 0 : this.name.hashCode());

result = 37 \* result + (int) this.minYearTemp;

result = 37 \* result + (int) this.maxYearTemp;

result = 37 \* result + (int) this.rainyDays;

return result;

}

public boolean accord(Plant plant){

boolean tempResult = (plant.getMinTemp() >= minYearTemp && plant.getMinTemp() <= maxYearTemp) &&

(plant.getMaxTemp() >= minYearTemp && plant.getMaxTemp() <= maxYearTemp);

boolean rainResult = (plant.getRequirement() == "Интенсивное" && rainyDays >= 250) ||

(plant.getRequirement() == "Среднее" && rainyDays < 250 && rainyDays >= 150) ||

(plant.getRequirement() == "Редкое" && rainyDays < 150 && rainyDays >= 50);

return (tempResult && rainResult);

}

public String toString(){

return String.format(REGION\_FORMAT\_STRING,name,minYearTemp,maxYearTemp,rainyDays);

}

public int compareTo(Region r){

if(name.compareTo(r.getName()) < 0) return -1;

if((name.compareTo(r.getName()) == 0) && (minYearTemp < r.getMinYearTemp())) return -1;

if((name.compareTo(r.getName()) == 0) && (minYearTemp == r.getMinYearTemp()) && (maxYearTemp < r.getMaxYearTemp())) return -1;

if((name.compareTo(r.getName()) == 0) && (minYearTemp == r.getMinYearTemp()) && (maxYearTemp == r.getMaxYearTemp()) && (rainyDays < r.getRainyDays())) return -1;

if((name.compareTo(r.getName()) == 0) && (minYearTemp == r.getMinYearTemp()) && (maxYearTemp == r.getMaxYearTemp()) && (rainyDays == r.getRainyDays())) return 0;

else return 1;

}

public void setName(String name){

this.name = name;

}

public void setMinYearTemp(int temp){

this.minYearTemp = temp;

}

public void setMaxYearTemp(int temp){

this.maxYearTemp = temp;

}

public void setRainyDays(int days){

this.rainyDays = days;

}

public String getName(){

return this.name;

}

public int getMinYearTemp(){

return this.minYearTemp;

}

public int getMaxYearTemp(){

return this.maxYearTemp;

}

public int getRainyDays(){

return this.rainyDays;

}

}

ObjDemo:

public class ObjDemo {

public static void bubbleSort(Comparable[] arr){

boolean flag;

for(int m = arr.length - 1; m>0; m--){

flag = true;

for(int j = 0; j < m; j++){

if(arr[j].compareTo(arr[j+1]) > 0){

Comparable b = arr[j];

arr[j] = arr[j+1];

arr[j+1] = b;

flag = false;

}

if(flag) break;

}

}

}

public static void putArr(Plant[] arr){

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

System.out.printf("%s, хэшкод: %15d\n",arr[i],arr[i].hashCode());

}

}

public static void putArr(Region[] arr){

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

System.out.printf("%s, хэшкод: %15d\n",arr[i],arr[i].hashCode());

}

}

public static void main(String[] arg){

System.out.print("1) Демонстрируем работу методов toString(), equals()");

System.out.println("и hashCode() для объектов класса Region");

Region region1 = new Region("Алтайский",-5,30,80);

Region region2 = region1;

Region region3 = new Region(region1);

Region region4 = new Region("Крымский",-10,50,30);

Region region5 = null;

System.out.printf("region1: %s, хэшкод: %d\n",region1,region1.hashCode());

System.out.printf("region2: %s, хэшкод: %d\n",region2,region2.hashCode());

System.out.printf("region3: %s, хэшкод: %d\n",region3,region3.hashCode());

System.out.printf("region4: %s, хэшкод: %d\n",region4,region4.hashCode());

System.out.printf("region5: %s",region5);

System.out.println("Результаты сравнения на равенство");

System.out.printf("region1 == region2: %s\n",region1.equals(region2));

System.out.printf("region1 == region3: %s\n",region1.equals(region3));

System.out.printf("region1 == region4: %s\n",region1.equals(region4));

System.out.printf("region1 == region5: %s\n",region1.equals(region5));

System.out.println();

System.out.print("2)Демонстрпируем сортировку элементов массивов");

System.out.println("arr1 и arr2 одним и тем же методом bubbleSort().");

Plant[] arr = new Plant[7];

arr[0] = new Plant("Борец","Травянистое","Интенсивное",1,40);

arr[1] = new Plant("Борец","Травянистое","Интенсивное",1,30);

arr[2] = new Plant("Борец","Травянистое","Интенсивное",2,40);

arr[3] = new Plant("Борец","Травянистое","Среднее",1,40);

arr[4] = new Plant("Борец","Дерево","Интенсивное",1,40);

arr[5] = new Plant("Живучка","Травянистое","Интенсивное",1,40);

arr[6] = new Plant("Борец","Травянистое","Интенсивное",1,40);

System.out.println("Массив Plant растений до сортировки:");

putArr(arr);

bubbleSort(arr);

System.out.println("Массив Plant растений после сортировки:");

putArr(arr);

Region[] arr1 = new Region[6];

arr1[0] = new Region("Сибирский",-20,40,140);

arr1[1] = new Region("Крым",-20,40,145);

arr1[2] = new Region("Сибирский",-20,45,140);

arr1[3] = new Region("Сибирский",30,40,140);

arr1[4] = new Region("Алтай",-20,40,180);

arr1[5] = new Region("Кавказ",-20,40,140);

System.out.println("Массик регионов Region до сортировки:");

putArr(arr1);

bubbleSort(arr1);

System.out.println("Массик регионов Region после сортировки:");

putArr(arr1);

System.out.println();

System.out.println("3) Демонстрируем взаимодействие объектов при определении соответствия растений и региона.");

Region region = new Region("Южный",-5,50,270);

Plant plant1 = new Plant("Яблоня","Дерево","Среднее",-10,40);

Plant plant2 = new Plant("Груша","Дерево","Интенсивное",0,30);

Plant plant3 = new Plant("Дудник","Травянистое","Редкое",5,40);

Plant plant4 = new Plant("Папоротник","Кустарное","Среднее",10,50);

Plant plant5 = new Plant("Ель","Дерево","Редкое",-20,40);

Plant plant6 = new Plant("Звездчатка","Травянистое","Интенсивное",0,35);

System.out.printf("Region: %s\n",region);

System.out.printf("plant1: %s\n",plant1);

System.out.printf("plant2: %s\n",plant2);

System.out.printf("plant3: %s\n",plant3);

System.out.printf("plant4: %s\n",plant4);

System.out.printf("plant5: %s\n",plant5);

System.out.printf("plant6: %s\n",plant6);

System.out.printf("plant1.accord(region): %s\n",plant1.accord(region));

System.out.printf("region.accord(plant1): %s\n",region.accord(plant1));

System.out.printf("plant2.accord(region): %s\n",plant2.accord(region));

System.out.printf("region.accord(plant2): %s\n",region.accord(plant2));

System.out.printf("plant3.accord(region): %s\n",plant3.accord(region));

System.out.printf("region.accord(plant3): %s\n",region.accord(plant3));

System.out.printf("plant4.accord(region): %s\n",plant4.accord(region));

System.out.printf("region.accord(plant4): %s\n",region.accord(plant4));

System.out.printf("plant5.accord(region): %s\n",plant5.accord(region));

System.out.printf("region.accord(plant5): %s\n",region.accord(plant5));

System.out.printf("plant6.accord(region): %s\n",plant6.accord(region));

System.out.printf("region.accord(plant6): %s\n",region.accord(plant6));

region.setName("Камчатка");

region.setMaxYearTemp(25);

region.setMinYearTemp(-40);

region.setRainyDays(100);

plant6.setName("Мох");

plant6.setType("Кустарник");

plant6.setRequirement("Редкое");

plant6.setMaxTemp(23);

plant6.setMinTemp(-15);

if(plant6.accord(region)){

System.out.printf("Объект %s\n",plant6);

System.out.println("соответствует объекту");

System.out.println(region);

} else {

System.out.printf("Объект %s\n",plant6);

System.out.println("не соответствует объекту");

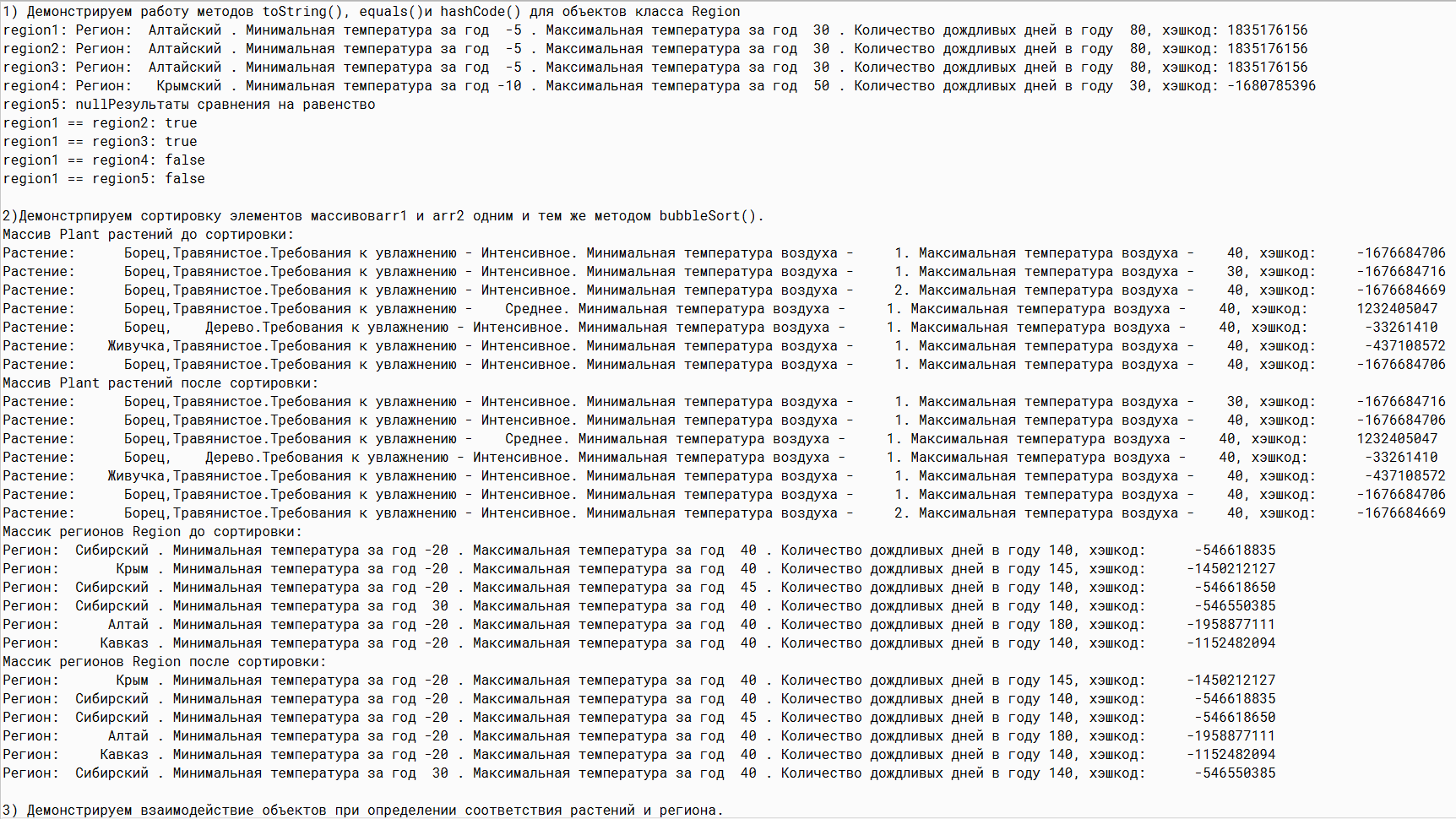
System.out.println(region);

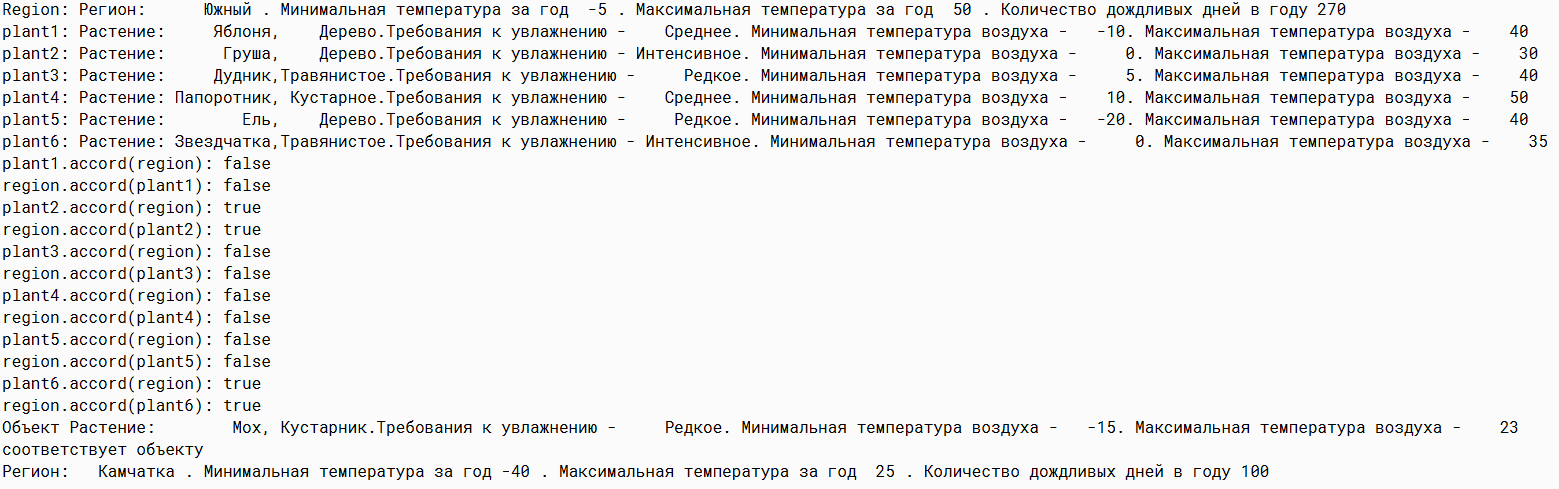
}

}

}

**Результаты выполнения программы:**





**Вывод:**

Проделав лабораторную работу я закрепил навыки разработки классов-шаблонов для создания объектов в соответствии с принципом инкапсуляции данных и методов для работы с этими данными.