Институт информационных технологий и управления в технических системах

Кафедра информационных технологий и компьютерных систем

ОТЧЕТ

По лабораторной работе №1

«Разработка подклассов. Переопределение методов в подклассах. Использование библиотечного класса Comparable»

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 ознакомление с реализацией принципа наследования в Java, получение навыков в проектировании классов-шаблонов, позволяющих осуществлять сравнение соответствующих объектов на равенство и больше/меньше/равно.

**Постановка задачи:**

Вариант 14

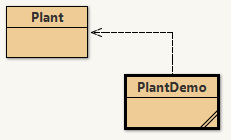
Проект1: Разработать в соответствии с вариантом задания класс (Растение), позволяющий создавать объекты, которые можно сравнивать на равенство (использовать переопределение методов equals(), hashCode() и toString() суперкласса Object) и на больше/меньше/равно. Разработать класс (РастениеDemo), демонстрирующий сравнение и упорядочивание (сортировку) объектов класса Растение.

Проект 2: Разработать подкласс РедкоеРастение класса Растение, позволяющий создавать объекты, которые можно сравнивать на равенство (использовать переопределение методов equals(), hashCode() и toString() суперкласса Растение) и на больше/меньше/равно. Разработать класс (РедкоеРастениеDemo), демонстрирующий сравнение и упорядочивание (сортировку) объектов класса РедкоеРастение. Продемонстрировать также возможность обращения к объектам класса РедкоеРастение с помощью ссылочной переменной суперкласса (Растение).

Проект 3: Скопировать проект Проект2 и включить в классы Растение и РедкоеРастение библиотечный интерфейс Comparable. Разработать класс (ComparableDemo), демонстрирующий использование интерфейсной ссылки для унификации действий при сортировке любых объектов (в данном случае объектов классов Растение и РедкоеРастение), включающих интерфейс Comparable.

Вариант 14.

**Структура первого проекта:**



**Текст классов:**

Plant:

public class 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;

}

@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 \*/

}

PlantDemo:

public class PlantDemo

{

public static void bubbleSort(Plant[] 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){

Plant b = arr[j];

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

arr[j+1] = b;

flag = false;

}

if(flag) break;

}

}

}

public static void putPlantArr(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 main(String arg[]){

Plant plant0 = new Plant();

plant0.setName("Мухоловка");

plant0.setType("Травянистое");

plant0.setRequirement("Среднее");

plant0.setMinTemp(5);

plant0.setMaxTemp(40);

System.out.printf("Plant0: %s, средняя температура равна %.2f\n",plant0.getName(),plant0.averageTemp());

Plant plant1 = new Plant("Тимьян","Травянистое","Интенсивное",3,25);

Plant plant2 = plant1;

Plant plant3 = new Plant(plant1);

Plant plant4 = new Plant("Яблоня","Дерево","Интенсивное",-5,50);

Plant plant5 = null;

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

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

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

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

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

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

System.out.printf("plant1==plant2: %s\n",plant1.equals(plant2));

System.out.printf("plant1==plant3: %s\n",plant1.equals(plant3));

System.out.printf("plant1==plant4: %s\n",plant1.equals(plant4));

System.out.printf("plant1==plant5: %s\n",plant1.equals(plant5));

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 растений до сортировки:");

putPlantArr(arr);

bubbleSort(arr);

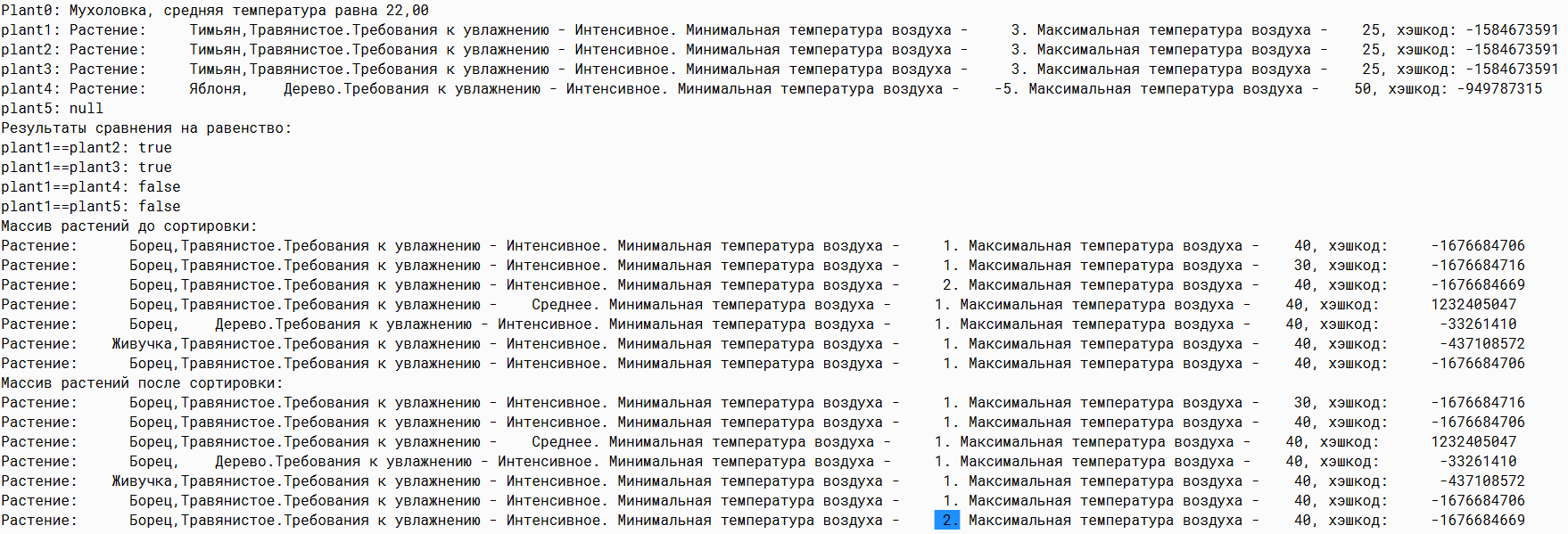
System.out.println("Массив растений после сортировки:");

putPlantArr(arr);

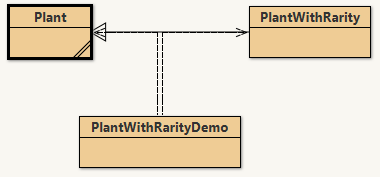
}

}

**Результат выполнения программы:**



**Структура второго проекта:**



**Текст классов:**

Plant:

public class 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;

}

@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 \*/

}

PlantWithRarity:

public class PlantWithRarity extends Plant

{

private final static String PlantWithRarity\_FORMAT\_STRING = "%s, редкое: %s";

private boolean isRare;

public PlantWithRarity(){

super();

isRare = false;

}

public PlantWithRarity(String name, String type, String requirement, int minTemp, int maxTemp, boolean isRare){

super(name,type,requirement,minTemp,maxTemp);

this.isRare = isRare;

}

public PlantWithRarity(PlantWithRarity ob){

super(ob);

this.isRare = ob.getRare();

}

public boolean equals(Object ob){

if(!super.equals(ob)) return false;

PlantWithRarity pwr = (PlantWithRarity) ob;

return (isRare == (pwr.getRare()));

}

public int hashCode(){

int result = super.hashCode();

result = 37 \* result + (this.isRare ? 1 : 0);

return result;

}

public int compareTo(Plant plant){

int comp = super.compareTo(plant);

if(comp < 0) return -1;

PlantWithRarity pwh = (PlantWithRarity) plant;

if((comp == 0) && (Boolean.toString(isRare).compareTo(Boolean.toString(pwh.getRare())) < 0)) return -1;

if((comp == 0) && (Boolean.toString(isRare).compareTo(Boolean.toString(pwh.getRare())) == 0)) return 0;

else return 1;

}

public String toString(){

return String.format(PlantWithRarity\_FORMAT\_STRING,super.toString(),isRare);

}

public boolean getRare(){

return this.isRare;

}

public void setRare(boolean rarity){

this.isRare = rarity;

}

}

PlantWithRarityDemo:

public class PlantWithRarityDemo

{

public static void bubbleSort(PlantWithRarity[] 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){

PlantWithRarity b = arr[j];

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

arr[j+1] = b;

flag = false;

}

if(flag) break;

}

}

}

public static void putPwrArr(PlantWithRarity[] 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[]){

PlantWithRarity pwr0 = new PlantWithRarity();

pwr0.setName("Мухоловка");

pwr0.setType("Травянистое");

pwr0.setRequirement("Среднее");

pwr0.setMinTemp(5);

pwr0.setMaxTemp(40);

pwr0.setRare(true);

System.out.printf("Plant0: %s, средняя температура равна %.2f , редкое: %s\n",pwr0.getName(),pwr0.averageTemp(), pwr0.getRare());

PlantWithRarity pwr1 = new PlantWithRarity("Тимьян","Травянистое","Интенсивное",3,25,false);

PlantWithRarity pwr2 = pwr1;

PlantWithRarity pwr3 = new PlantWithRarity(pwr1);

PlantWithRarity pwr4 = new PlantWithRarity("Яблоня","Дерево","Интенсивное",-5,50,false);

PlantWithRarity pwr5 = null;

System.out.printf("plant1: %s, хэшкод: %d\n",pwr1,pwr1.hashCode());

System.out.printf("plant2: %s, хэшкод: %d\n",pwr2,pwr2.hashCode());

System.out.printf("plant3: %s, хэшкод: %d\n",pwr3,pwr3.hashCode());

System.out.printf("plant4: %s, хэшкод: %d\n",pwr4,pwr4.hashCode());

System.out.printf("plant5: %s\n",pwr5);

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

System.out.printf("plant1==plant2: %s\n",pwr1.equals(pwr2));

System.out.printf("plant1==plant3: %s\n",pwr1.equals(pwr3));

System.out.printf("plant1==plant4: %s\n",pwr1.equals(pwr4));

System.out.printf("plant1==plant5: %s\n",pwr1.equals(pwr5));

Plant plant = pwr1;

System.out.println("Выполнен оператор Plant plant = plant1;");

System.out.printf("plant == pwr3: %s\n",plant.equals(pwr3)); //true

PlantWithRarity[] arr = new PlantWithRarity[8];

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

arr[1] = new PlantWithRarity("Борец","Травянистое","Интенсивное",1,40,false);

arr[2] = new PlantWithRarity("Борец","Травянистое","Интенсивное",1,30,true);

arr[3] = new PlantWithRarity("Борец","Травянистое","Интенсивное",2,40,true);

arr[4] = new PlantWithRarity("Борец","Травянистое","Среднее",1,40,true);

arr[5] = new PlantWithRarity("Борец","Дерево","Интенсивное",1,40,true);

arr[6] = new PlantWithRarity("Живучка","Травянистое","Интенсивное",1,40,true);

arr[7] = new PlantWithRarity("Борец","Травянистое","Интенсивное",1,40,true);

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

putPwrArr(arr);

bubbleSort(arr);

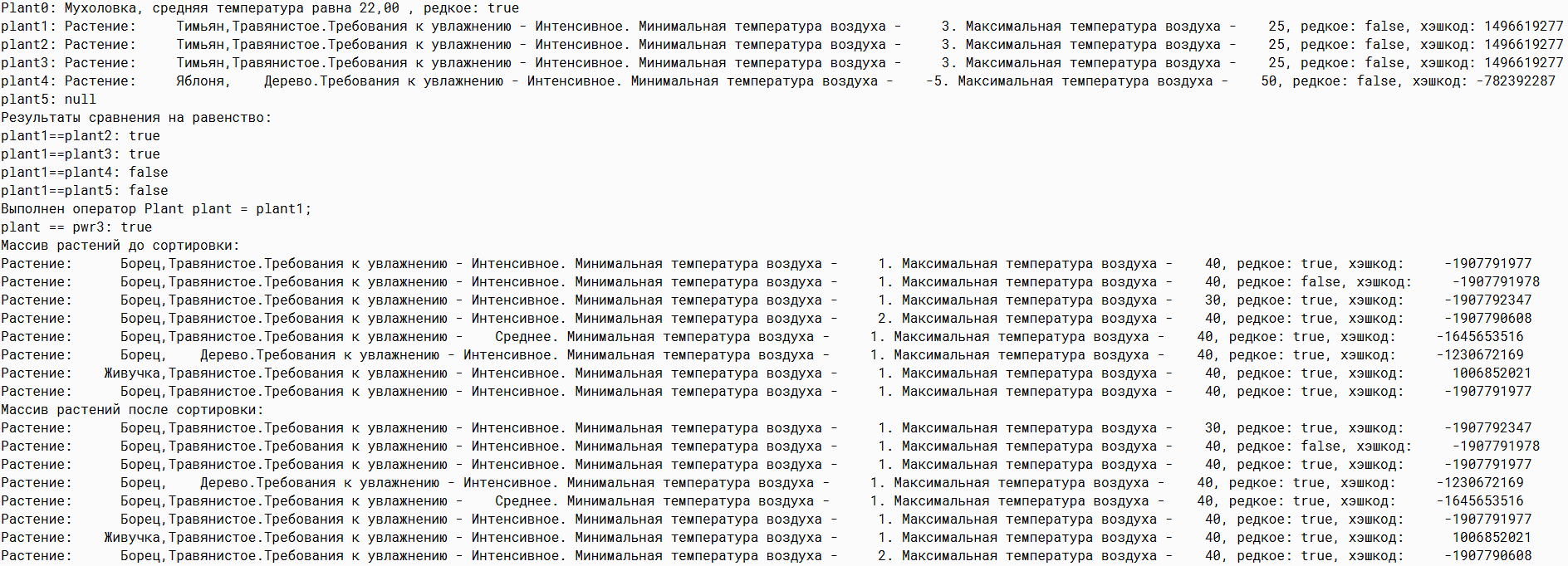
System.out.println("Массив растений после сортировки:");

putPwrArr(arr);

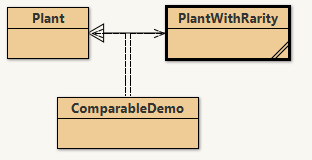
}

}

**Результат выполнения программы:**



**Структура третьего 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;

}

@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 \*/

}

PlantWithRarity:

public class PlantWithRarity extends Plant

{

private final static String PlantWithRarity\_FORMAT\_STRING = "%s, редкое: %s";

private boolean isRare;

public PlantWithRarity(){

super();

isRare = false;

}

public PlantWithRarity(String name, String type, String requirement, int minTemp, int maxTemp, boolean isRare){

super(name,type,requirement,minTemp,maxTemp);

this.isRare = isRare;

}

public PlantWithRarity(PlantWithRarity ob){

super(ob);

this.isRare = ob.getRare();

}

public boolean equals(Object ob){

if(!super.equals(ob)) return false;

PlantWithRarity pwr = (PlantWithRarity) ob;

return (isRare == (pwr.getRare()));

}

public int hashCode(){

int result = super.hashCode();

result = 37 \* result + (this.isRare ? 1 : 0);

return result;

}

public int compareTo(Plant plant){

int comp = super.compareTo(plant);

if(comp < 0) return -1;

PlantWithRarity pwh = (PlantWithRarity) plant;

if((comp == 0) && (Boolean.toString(isRare).compareTo(Boolean.toString(pwh.getRare())) < 0)) return -1;

if((comp == 0) && (Boolean.toString(isRare).compareTo(Boolean.toString(pwh.getRare())) == 0)) return 0;

else return 1;

}

public String toString(){

return String.format(PlantWithRarity\_FORMAT\_STRING,super.toString(),isRare);

}

public boolean getRare(){

return this.isRare;

}

public void setRare(boolean rarity){

this.isRare = rarity;

}

}

ComparableDemo:

public class ComparableDemo

{

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(PlantWithRarity[] 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[] args){

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);

PlantWithRarity[] arr1 = new PlantWithRarity[8];

arr1[0] = new PlantWithRarity("Борец","Травянистое","Интенсивное",1,40,true);

arr1[1] = new PlantWithRarity("Борец","Травянистое","Интенсивное",1,40,false);

arr1[2] = new PlantWithRarity("Борец","Травянистое","Интенсивное",1,30,true);

arr1[3] = new PlantWithRarity("Борец","Травянистое","Интенсивное",2,40,true);

arr1[4] = new PlantWithRarity("Борец","Травянистое","Среднее",1,40,true);

arr1[5] = new PlantWithRarity("Борец","Дерево","Интенсивное",1,40,true);

arr1[6] = new PlantWithRarity("Живучка","Травянистое","Интенсивное",1,40,true);

arr1[7] = new PlantWithRarity("Борец","Травянистое","Интенсивное",1,40,true);

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

putArr(arr1);

bubbleSort(arr1);

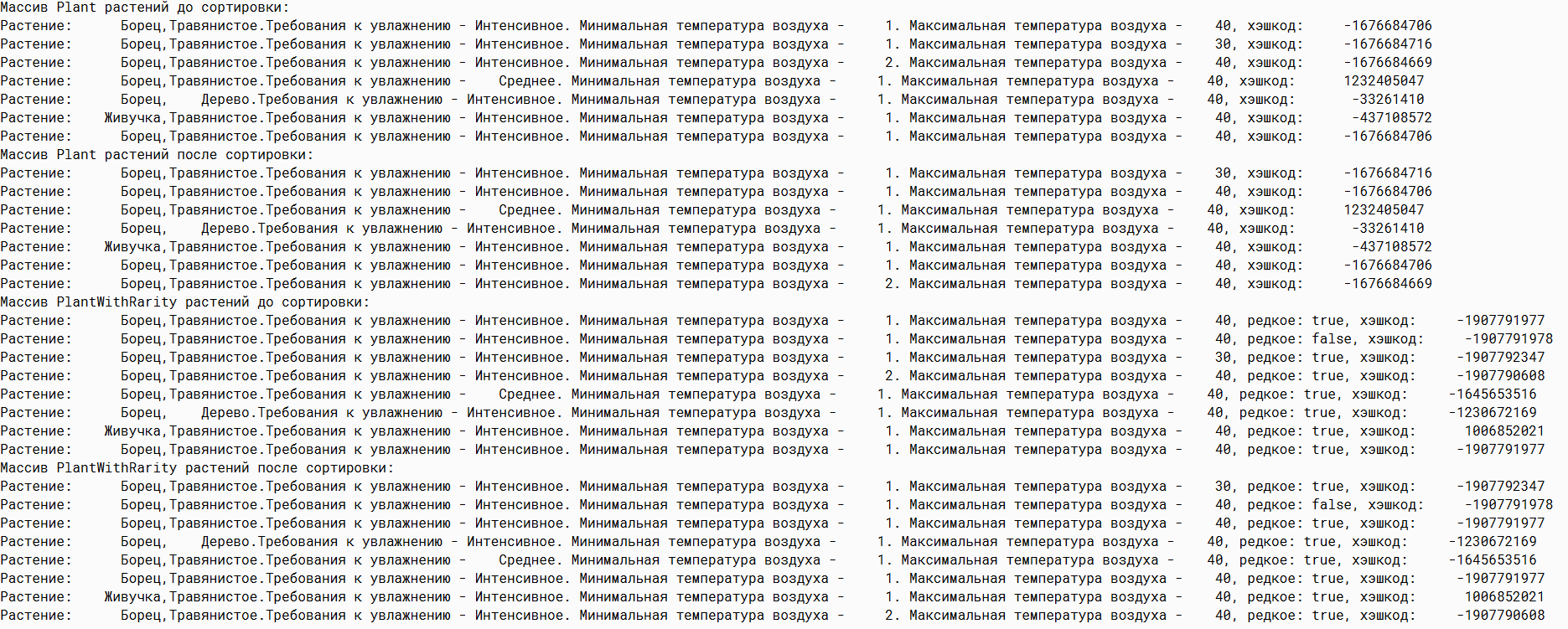
System.out.println("Массив PlantWithRarity растений после сортировки:");

putArr(arr1);

}

}

**Результат выполнения программы:**



**Выводы:**

Проделав лабораторную работу я ознакомился с реализацией принципа наследования в Java и получил навыки проектирования классов-шаблонов.