Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра «ЭВМиС»

Отчёт по лабораторной работе №3

“Интерфейсы”

Выполнил:

Студент 3-го курса

Группы МС-7

Мороз Е. В.

Проверил:

Николаюк-Ртищева М. В.

Брест, 2023

**Цель работы:** ознакомиться с основными способами создания и работы с интерфейсами.

**Ход работы**

**1. Растение.** Возможности: задать и редактировать информацию о растении; получить информацию об имеющихся растениях в оранжерее/растениях определенного вида/месте происхождения; закупить новый вид растений; произвести полив; задать температуру; установить освещение; выкопать определенный вид растений (убрать из оранжереи). Добавить специализированные методы для Кустовое растение, Цветковое растение, Комнатное растение.

Исходный код программы:

**Plant.java:**

public interface Plant {  
 String getName();  
 String getDescription();  
 void setTemperature(float temperature);  
 void waterPlant();  
  
 void setPlantLightning(boolean isLight);  
 void editName(String newName, String newDescription);  
 void getPlant();  
}

**Greenhouse.java:**

import java.util.ArrayList;  
import java.util.List;  
  
public class Greenhouse {  
 private List<Plant> plantList = new ArrayList<>();  
  
 Greenhouse() { }  
  
 public void getPlants() {  
 System.*out*.println("Plants in greenhouse: ");  
 for (Plant plant : plantList) {  
 System.*out*.println(plant.getName() + " : " + plant.getDescription());  
 }  
 }  
  
 public void buyPlant(Plant plant) {  
 this.plantList.add(plant);  
 System.*out*.println("Plant " + plant.getName() + " was purchased");  
 }  
  
 public void removePlant(Plant plant) {  
  
 String plantName = plant.getName();  
  
 for (Plant p : plantList) {  
 if (plant == p) {  
 plantList.remove(p);  
 System.*out*.println("Removed plant " + plantName);  
 return;  
 }  
 }  
 System.*out*.println("Plant not found");  
 }  
}

**FloweringPlant.java:**

public class FloweringPlant implements Plant {  
  
 private boolean isWatered = false;  
 private float temperature = 18;  
 private boolean isLight = false;  
 private String name;  
 private String description;  
  
 FloweringPlant(String name, String description) {  
 this.name = name;  
 this.description = description;  
 }  
  
 @Override  
 public String getName() {  
 return this.name;  
 }  
  
 @Override  
 public String getDescription() {  
 return this.description;  
 }  
  
 @Override  
 public void setTemperature(float temperature) {  
 this.temperature = temperature;  
 System.*out*.println(this.getName() + " set temperature to " + this.temperature);  
 }  
  
 @Override  
 public void waterPlant() {  
 this.isWatered = true;  
 System.*out*.println(this.getName() + " is watered");  
 }  
  
 @Override  
 public void setPlantLightning(boolean isLight) {  
 this.isLight = isLight;  
 System.*out*.println("For " + this.getName() + " set light to " + this.isLight);  
 }  
  
 @Override  
 public void editName(String newName, String newDescription) {  
 this.name = newName;  
 this.description = newDescription;  
 }  
  
 @Override  
 public void getPlant() {  
 System.*out*.println("Plant name: " + this.name);  
 System.*out*.println("Plant description" + this.description);  
 }  
}

**BushPlant.java:**

public class BushPlant implements Plant {  
  
 private boolean isWatered = false;  
 private float temperature = 18;  
 private boolean isLight = false;  
 private String name;  
 private String description;  
  
 BushPlant(String name, String description) {  
 this.name = name;  
 this.description = description;  
 }  
  
 @Override  
 public String getName() {  
 return this.name;  
 }  
  
 @Override  
 public String getDescription() {  
 return this.description;  
 }  
  
 @Override  
 public void setTemperature(float temperature) {  
 this.temperature = temperature;  
 System.*out*.println(this.getName() + " set temperature to " + this.temperature);  
 }  
  
 @Override  
 public void waterPlant() {  
 this.isWatered = true;  
 System.*out*.println(this.getName() + " is watered");  
 }  
  
 @Override  
 public void setPlantLightning(boolean isLight) {  
 this.isLight = isLight;  
 System.*out*.println("For " + this.getName() + " set light to " + this.isLight);  
 }  
  
 @Override  
 public void editName(String newName, String newDescription) {  
 this.name = newName;  
 this.description = newDescription;  
 }  
  
 @Override  
 public void getPlant() {  
 System.*out*.println("Plant name: " + this.name);  
 System.*out*.println("Plant description" + this.description);  
 }  
}

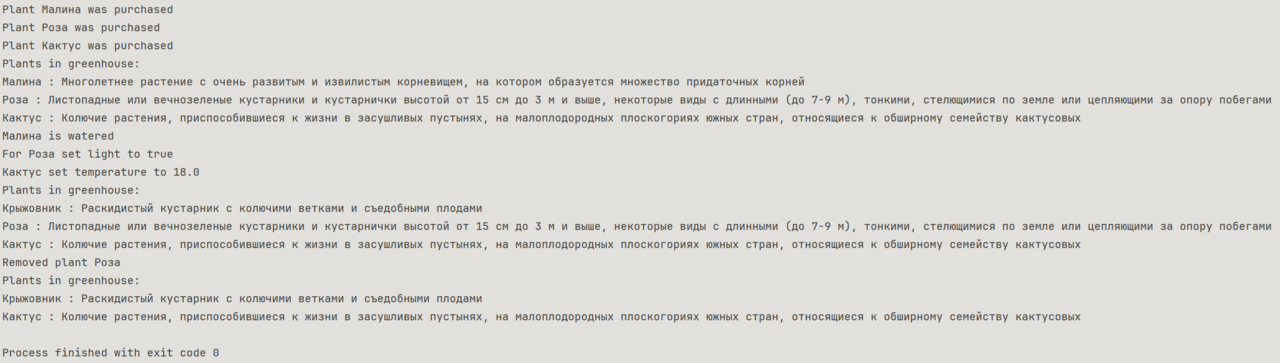
**IndoorPlant.java:**

public class IndoorPlant implements Plant{  
  
 private boolean isWatered = false;  
 private float temperature = 18;  
 private boolean isLight = false;  
 private String name;  
 private String description;  
  
 IndoorPlant(String name, String description) {  
 this.name = name;  
 this.description = description;  
 }  
  
 @Override  
 public String getName() {  
 return this.name;  
 }  
  
 @Override  
 public String getDescription() {  
 return this.description;  
 }  
  
 @Override  
 public void setTemperature(float temperature) {  
 this.temperature = temperature;  
 System.*out*.println(this.getName() + " set temperature to " + this.temperature);  
 }  
  
 @Override  
 public void waterPlant() {  
 this.isWatered = true;  
 System.*out*.println(this.getName() + " is watered");  
 }  
  
 @Override  
 public void setPlantLightning(boolean isLight) {  
 this.isLight = isLight;  
 System.*out*.println("For " + this.getName() + " set light to " + this.isLight);  
 }  
  
 @Override  
 public void editName(String newName, String newDescription) {  
 this.name = newName;  
 this.description = newDescription;  
 }  
  
 @Override  
 public void getPlant() {  
 System.*out*.println("Plant name: " + this.name);  
 System.*out*.println("Plant description" + this.description);  
 }  
}

**Main.java:**

public class Main {  
 public static void main(String[] args) {  
  
 Greenhouse winterGarden = new Greenhouse();  
  
 Plant bushPlant = new BushPlant("Малина", "Многолетнее растение с очень развитым и извилистым корневищем, на котором образуется множество придаточных корней");  
 Plant floweringPlant = new FloweringPlant("Роза", "Листопадные или вечнозеленые кустарники и кустарнички высотой от 15 см до 3 м и выше, некоторые виды с длинными (до 7-9 м), тонкими, стелющимися по земле или цепляющими за опору побегами");  
 Plant indoorPlant = new IndoorPlant("Кактус", "Колючие растения, приспособившиеся к жизни в засушливых пустынях, на малоплодородных плоскогориях южных стран, относящиеся к обширному семейству кактусовых");  
  
 winterGarden.buyPlant(bushPlant);  
 winterGarden.buyPlant(floweringPlant);  
 winterGarden.buyPlant(indoorPlant);  
  
 winterGarden.getPlants();  
  
 bushPlant.waterPlant();  
 floweringPlant.setPlantLightning(true);  
 indoorPlant.setTemperature(18);  
  
 bushPlant.editName("Крыжовник", "Раскидистый кустарник с колючими ветками и съедобными плодами");  
  
 winterGarden.getPlants();  
  
 winterGarden.removePlant(floweringPlant);  
  
 winterGarden.getPlants();  
 }  
}

Результат выполнения программы:



2. **Шеф-повар.** Определить иерархию овощей. Сделать салат. Подсчитать калорийность. Провести сортировку овощей для салата на основе одного из параметров. Найти овощи в салате, соответствующие заданному диапазону калорийности.

Исходный код программы:

**Vegetable.java:**

public abstract class Vegetable {  
 protected final float calorie;  
 protected final float weight;  
 protected final int count;  
  
 public Vegetable(float weight, int count) {  
 this.calorie = (int) Math.*floor*( ((weight / 100) \* 23) \* count );  
 this.weight = weight;  
 this.count = count;  
 }  
  
 public abstract float getCalorie();  
 public abstract int getCount();  
 public abstract float getWeight();  
 public abstract String getVegetableType();  
}

**Tomato.java:**

public class Tomato extends Vegetable {  
  
 Tomato(float weight, int count) {  
 super(weight, count);  
 }  
  
 @Override  
 public String getVegetableType() {  
 return "Tomato";  
 }  
  
 @Override  
 public float getCalorie() {  
 return super.calorie;  
 }  
  
 @Override  
 public int getCount() {  
 return super.count;  
 }  
  
 @Override  
 public float getWeight() {  
 return super.weight;  
 }  
}

**Onion.java:**

public class Onion extends Vegetable {  
  
  
 Onion(float weight, int count) {  
 super(weight, count);  
 }  
  
 @Override  
 public String getVegetableType() {  
 return "Onion";  
 }  
  
 @Override  
 public float getCalorie() {  
 return super.calorie;  
 }  
  
 @Override  
 public int getCount() {  
 return super.count;  
 }  
  
 @Override  
 public float getWeight() {  
 return super.weight;  
 }  
}

**Cucumber.java:**

public class Cucumber extends Vegetable {  
  
 Cucumber(float weight, int count) {  
 super(weight, count);  
 }  
  
 @Override  
 public float getCalorie() {  
 return super.calorie;  
 }  
  
 @Override  
 public int getCount() {  
 return super.count;  
 }  
  
 @Override  
 public float getWeight() {  
 return super.weight;  
 }  
  
 @Override  
 public String getVegetableType() {  
 return "Cucumber";  
 }  
}

**Salad.java:**

import java.util.TreeSet;  
  
public class Salad {  
  
 VegetableComparator vegetableComparator = new VegetableComparator();  
 TreeSet<Vegetable> salad = new TreeSet<Vegetable>(vegetableComparator);  
 Salad() { }  
  
 public void addIngredient(Vegetable ingredient) {  
 this.salad.add(ingredient);  
 }  
  
 public void getSaladIngredients() {  
 for (Vegetable v : salad) {  
 System.*out*.println(v.getVegetableType() + " " + v.count + " " + v.weight + " " + v.calorie);  
 }  
 }  
  
 private void printIngredientInfo(Vegetable v) {  
 System.*out*.println(v.getVegetableType() + " " + v.count + " " + v.weight + " " + v.calorie);  
 }  
  
 void getCalorieFromRange(float cr1, float cr2) {  
 for (Vegetable v : salad) {  
 if (v.getCalorie() >= cr1 && v.getCalorie() <= cr2) {  
 this.printIngredientInfo(v);  
 }  
 }  
 }  
}

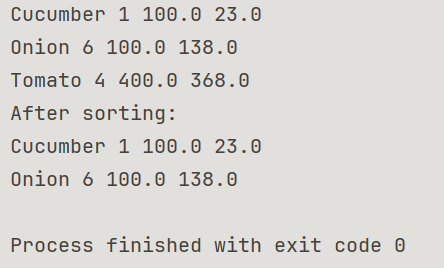
**VegetableComparator.java:**

import java.util.Comparator;  
  
public class VegetableComparator implements Comparator<Vegetable> {  
  
 @Override  
 public int compare(Vegetable o1, Vegetable o2) {  
 if (o1.calorie == o2.calorie) return 0;  
  
 return (o1.calorie > o2.calorie) ? 1 : -1;  
 }  
}

**Main.java:**

public class Main {  
 public static void main(String[] args) {  
  
 Salad salad = new Salad();  
  
 Vegetable tomato = new Tomato(400, 4);  
 Vegetable onion = new Onion(100, 6);  
 Vegetable cucumber = new Cucumber(100, 1);  
  
 salad.addIngredient(tomato);  
 salad.addIngredient(onion);  
 salad.addIngredient(cucumber);  
  
 salad.getSaladIngredients();  
  
 System.*out*.println("After filtering: ");  
 salad.getCalorieFromRange(20, 150);  
 }  
}

Результат работы программы:



**Вывод:** в результате выполнения данной лабораторной работы, я ознакомиться с основными способами создания интерфейсов и работы с ними.