МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №6

з дисципліни «Об’єктно-орієнтоване програмування»

на тему: *«Наслідування та поліморфізм.»*

Виконав:

студент гр.ІО-31

Долинний О.В.

Перевірив:

Невдащенко М.В.

Київ 2014

**Лабораторна робота №6**

**Тема:** Наслідування та поліморфізм.

**Мета:** Ознайомлення з механізмом наслідування та принципом поліморфізму. Використання механізму наслідування та принципу поліморфізму в мові Java. Здобуття навичок у використанні механізму наслідування та принципу поліморфізму.

1. ***Варіант завдання***

Номер залікової книжки  *3110*

Варіант: *C3 = 3110 mod 13 = 3*

Завдання: Визначити ієрархію овочів. Зробити салат. Порахувати калорійність салату. Провести сортування салату на основі одного з параметрів. Знайти овоч у салаті, , що відповідає заданому діапазону калорійності.

Створити узагальнений клас та не менше 3 класів-нащадків, що описують задану варіантом область знань. Створити клас, що складається з масиву об’єктів, з яким можна виконати вказані варіантом дії. Необхідно обробити всі виключні ситуації, що можуть виникнути під час виконання програмного коду. Всі змінні повинні бути описані та значення їх задані у виконавчому методі. Код повинен відповідати стандартам JCC та бути детально задокументований.

1. ***Опис алгоритму програми***

Клас *Main* містить виконавчий метод, що створює салат, ираховує його калорійність, проводить сортування інгредієнтів салату на основі маси овочів за спаданням, знаходить овоч, що відповідає заданому діапазону калорійності.

У класі *Main* створюється екземпляр класу *Salad*, екземпляр класу *Salad* складається з масиву екземплярів класу *Vegetables.*

*У* класі *Salad* знаходяться методи для знаходження калорійності салату, сортування інгредієнтів салату на основі маси овочів за спаданням, знаходження овочів, що відповідають заданому діапазону калорійності.

1. ***Програмний код***

Class Main

/\*

\* @(#)Main.java 1.0 30/04/14

\*

\* Copyright (c) 2014 Alex Dolinniy

\*/

**package** lab\_6;

/\*\* Class Main was created to do 6 lab on Programming.

\* The task consists in determing hierarchy of vegetables,

\* making salad, counting caloricity of salad, sorting salad

\* for its parameter and search vegetables in salad,

\* that have caloricity in specified range.

\*

\* **@version** 1.0 30 April 2014

\* **@author** Alex Dolinniy

\* **@since** 1.0

\*/

**public** **class** Main {

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

/\* Input data about vegetables \*/

**final** **int** listLength = 7;

Vegetables[] listVeg = **new** Vegetables [listLength];

listVeg[0] = **new** Bulbs("onion", 10, 41.2);

listVeg[1] = **new** Seeds("bean", 31.2, 73.4);

listVeg[2] = **new** Tubers("potato", 51, 80.6);

listVeg[3] = **new** Leaves("lettuce", 30, 17.8);

listVeg[4] = **new** Leaves("cabbage", 71, 27.778);

listVeg[5] = **new** Roots("carrot", 50.12, 37.5);

listVeg[6] = **new** Roots("radish", 48.23, 21.38);

/\* Checking input \*/

**boolean** flag = **true**;

**for** (**int** i = 0; (i < listVeg.length)&&(flag); i++)

**for** (**int** j = i+1; j < listVeg.length; j++)

**if** ((listVeg[i] == **null**)||(listVeg[j] == **null**)

||(listVeg[i].getName() == listVeg[j].getName()))

flag = **false**;

/\* Doing task if input data is true \*/

**if** (flag == **true**) {

Salad salad = **new** Salad(listVeg);

salad.setSalad(listVeg);

System.*out*.println(salad.getSalad());

/\* Calculation caloricity of salad\*/

**double** saladCaloricity = salad.getCaloricitySalad();

System.*out*.println("\nCaloricity of salad:" + saladCaloricity);

/\* Sorting salad by weight of vegetables by increase\*/

System.*out*.println("\nResults of sorting" );

salad.sortSaladWeight();

System.*out*.println(salad.getSalad());

/\*Input range of caloricity and searching elements\*/

**int** minCaloricity = 10;

**int** maxCaloricity = 71;

System.*out*.println("\nVegetables from diapason " +

minCaloricity + " : " + maxCaloricity);

String[] foundVeg =

salad.searchElement(minCaloricity,maxCaloricity);

**for** (**int** i = 0; i < foundVeg.length; i++)

**if** (foundVeg[i] != **null**)

System.*out*.println(foundVeg[i]);

}

**else**

System.*out*.println("Error in input of data");

}

}

Class Salad

/\*

\* @(#)Salad.java 1.0 30/04/14

\*

\* Copyright (c) 2014 Alex Dolinniy

\*/

**package** lab\_6;

/\*\* Class Salad creates array of vegetables.

\* Class has getter, setter, constructor and

\* method getCaloricitySalad that finds caloricity of salad,

\* method sortSaladWeight that sorts salad by weight of parts,

\* method searchElement that finds vegetables in salad,

\* that have caloricity in specified range.

\*

\* **@version** 1.0 30 April 2014

\* **@author** Alex Dolinniy

\* **@since** 1.0

\*/

**public** **class** Salad {

/\*\*

\* **@param** salad

\* array of vegetables

\*/

**private** Vegetables[] salad;

/\*\*

\* Constructor of Salad

\*/

**public** Salad(Vegetables[] veg) {

salad = **new** Vegetables[veg.length];

setSalad(veg);

}

/\*\*

\* Setter

\*/

**public** Vegetables[] setSalad(Vegetables[] veg) {

**for** (**int** i = 0; i < veg.length; i++)

salad[i] = veg[i];

**return** salad;

}

/\*\*

\* **@return** caloricity of salad

\*/

**public** **double** getCaloricitySalad() {

**double** caloricitySalad = 0;

**for** (**int** i = 0; i < salad.length; i++)

caloricitySalad += salad[i].getWeight() \*

salad[i].getCaloricity();

**return** caloricitySalad;

}

/\*\*

\* methode sorts salad by weight of ingridient by increase

\*/

**public** **void** sortSaladWeight() {

**for**(**int** i = salad.length - 1; i >= 0; i--)

**for**(**int** j = 0 ; j < i ; j++)

**if**( salad[i].getWeight() > salad[j].getWeight()) {

Vegetables veg = salad[j];

salad[j] = salad[i];

salad[i] = veg;

}

}

/\*\*

\* **@return** array of elements that has weight in required range

\*/

**public** String[] searchElement(**double** limit1, **double** limit2) {

String list[] = **new** String[salad.length];

**int** counter = 0;

**for**(**int** i = 0 ; i < salad.length ; i++)

**if**( (salad[i].getCaloricity() >= limit1)

&&(salad[i].getCaloricity() <= limit2))

list[counter++] = salad[i].getName();

**return** list;

}

/\*\*

\* Getter

\*/

**public** String getSalad() {

String result = "";

**for** (**int** i = 0; i < salad.length; i++)

result = result + '\n' + salad[i].getInformation();

**return** result;

}

}

Class Vegetables

/\*

\* @(#)Vegetables.java 1.0 30/04/14

\*

\* Copyright (c) 2014 Alex Dolinniy

\*/

**package** lab\_6;

/\*\* Class Vegetables creates object vegetable.

\* Class has getter, setter, constructor and

\* method getCaloricitySalad that finds caloricity of salad,

\* method sortSaladWeight that sorts salad by weight of parts,

\* method searchElement that finds vegetables in salad,

\* that have caloricity in specified range.

\*

\* **@version** 1.0 30 April 2014

\* **@author** Alex Dolinniy

\* **@since** 1.0

\*/

**public** **class** Vegetables {

/\*\*

\* **@param** name

\* name of vegetable

\* **@param** weight

\* weight of vegetables

\* **@param** caloricity

\* caloricity of vegetables

\*/

**private** String name;

**private** **double** weight;

**private** **double** caloricity;

/\*\*

\* Constructor of Vegetable

\*/

**public** Vegetables (String inName, **double** inWeight, **double** inCaloricity) {

setName(inName);

setWeight(inWeight);

setCaloricity(inCaloricity);

}

/\*\*

\* **@return** name

\*/

**public** String getName() {

**return** name;

}

/\*\*

\* **@return** weight

\*/

**public** **double** getWeight() {

**return** weight;

}

/\*\*

\* **@return** caloricity

\*/

**public** **double** getCaloricity() {

**return** caloricity;

}

/\*\*

\* **@param** name

\*/

**public** **void** setName (String inName) {

name = inName;

}

/\*\*

\* **@param** weight

\*/

**public** **void** setWeight (**double** inWeight) {

weight = inWeight;

}

/\*\*

\* **@param** caloricity

\*/

**public** **void** setCaloricity (**double** inCaloricity) {

caloricity = inCaloricity;

}

/\*\*

\* **@return** information about vegetable in salad

\*/

**public** String getInformation() {

**return** "Name:" + name + " Weight:" + String.*valueOf*(weight) +

" Caloricity:" + String.*valueOf*(caloricity);

}

}

Class Bulbs

/\*

\* @(#)Bulbs.java 1.0 30/04/14

\*

\* Copyright (c) 2014 Alex Dolinniy

\*/

**package** lab\_6;

/\*\* Class Bulbs creates object of Bulbs.

\* Class Bulbs extends class Vegetables.

\* Class has constructor and overriden method getInformation.

\*

\* **@version** 1.0 30 April 2014

\* **@author** Alex Dolinniy

\* **@since** 1.0

\*/

**public** **class** Bulbs **extends** Vegetables {

/\*\*

\* **@param** type

\* type of vegetable - Bulbs

\*/

**private** String type = "Bulbs";

/\*\*

\* Constructor of Bulbs

\*/

**public** Bulbs(String inName, **double** inWeight, **double** inCaloricity) {

**super**(inName, inWeight, inCaloricity);

}

/\*\*

\* **@return** information

\*/

@Override

**public** String getInformation() {

**return** "Type:" + type + " " + **super**.getInformation();

}

}

Class Leaves

/\*

\* @(#)Leaves.java 1.0 30/04/14

\*

\* Copyright (c) 2014 Alex Dolinniy

\*/

**package** lab\_6;

/\*\* Class Leaves creates object of Leaves.

\* Class Leaves extends class Vegetables.

\* Class has constructor and overriden method getInformation.

\*

\* **@version** 1.0 30 April 2014

\* **@author** Alex Dolinniy

\* **@since** 1.0

\*/

**public** **class** Leaves **extends** Vegetables {

/\*\*

\* **@param** type

\* type of vegetable - Leaves

\*/

**private** String type = "Leaves";

/\*\*

\* Constructor of Bulbs

\*/

**public** Leaves(String inName, **double** inWeight, **double** inCaloricity) {

**super**(inName, inWeight, inCaloricity);

}

/\*\*

\* **@return** information

\*/

**public** String getInformation() {

**return** "Type:" + type + " " + **super**.getInformation();

}

}

Class Roots

/\*

\* @(#)Roots.java 1.0 30/04/14

\*

\* Copyright (c) 2014 Alex Dolinniy

\*/

**package** lab\_6;

/\*\* Class Roots creates object of Roots.

\* Class Roots extends class Vegetables.

\* Class has constructor and overriden method getInformation.

\*

\* **@version** 1.0 30 April 2014

\* **@author** Alex Dolinniy

\* **@since** 1.0

\*/

**public** **class** Roots **extends** Vegetables {

/\*\*

\* **@param** type

\* type of vegetable - Roots

\*/

**private** String type = "Roots";

/\*\*

\* Constructor of Roots

\*/

**public** Roots(String inName, **double** inWeight, **double** inCaloricity) {

**super**(inName, inWeight, inCaloricity);

}

/\*\*

\* **@return** information

\*/

**public** String getInformation() {

**return** "Type:" + type + " " + **super**.getInformation();

}

}

Class Seeds

/\*

\* @(#)Seeds.java 1.0 30/04/14

\*

\* Copyright (c) 2014 Alex Dolinniy

\*/

**package** lab\_6;

/\*\* Class Seeds creates object of Seeds.

\* Class Seeds extends class Vegetables.

\* Class has constructor and overriden method getInformation.

\*

\* **@version** 1.0 30 April 2014

\* **@author** Alex Dolinniy

\* **@since** 1.0

\*/

**public** **class** Seeds **extends** Vegetables {

/\*\*

\* **@param** type

\* type of vegetable - Seeds

\*/

**private** String type = "Seeds";

/\*\*

\* Constructor of Seeds

\*/

**public** Seeds(String inName, **double** inWeight, **double** inCaloricity) {

**super**(inName, inWeight, inCaloricity);

}

/\*\*

\* **@return** information

\*/

**public** String getInformation() {

**return** "Type:" + type + " " + **super**.getInformation();

}

}

Class Tubers

/\*

\* @(#)Tubers.java 1.0 30/04/14

\*

\* Copyright (c) 2014 Alex Dolinniy

\*/

**package** lab\_6;

/\*\* Class Tubers creates object of Tubers.

\* Class Tubers extends class Vegetables.

\* Class has constructor and overriden method getInformation.

\*

\* **@version** 1.0 30 April 2014

\* **@author** Alex Dolinniy

\* **@since** 1.0

\*/

**public** **class** Tubers **extends** Vegetables {

/\*\*

\* **@param** type

\* type of vegetable - Tubers

\*/

**private** String type = "Tubers";

/\*\*

\* Constructor of Seeds

\*/

**public** Tubers(String inName, **double** inWeight, **double** inCaloricity) {

**super**(inName, inWeight, inCaloricity);

}

/\*\*

\* **@return** information

\*/

**public** String getInformation() {

**return** "Type:" + type + " " + **super**.getInformation();

}

}

***Висновок***

Під час лабораторної роботи я використав механізм наслідування при створенні ієрархії овочів. Використав принцип поліморфізму при виведенні інформації про овоч. Здобув навички у використанні механізму наслідування та принципу поліморфізму.

Код був оформлений і задокументований відповідно до стандартів JCC.