МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАТИКИ І ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ

"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

ЛАБОРАТОРНА РОБОТА **№6**

З КУРСУ "**ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ**"

Виконав: **студент 1 курсу**

**гр. ІО-35**

**Головенько В.Д**

Прийняв: **Невдащенко М.В.**

Київ - 2014

**Тема:** Наслідування та поліморфізм.

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

Механізму наслідування та принципу поліморфізму.

**Завдання:**

**Номер залікової книжки** – *3504*

*Визначити ієрархію квітів. Створити кілька об’єктів-квітів. Зібрати букет (використовуючи аксесуари) з визначенням його вартості. Провести сортування квітів у букеті за рівнем свіжості. Знайти квітку в букеті, що відповідає заданому діапазону довжин.*

Створити узагальнений клас та не менше 3 класів-нащадків, що описують задану варіантом (п.2) область знань. 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.

**Код програми:**

**Основний клас з виконавчим методом:**

/\*\*Класс, выполняющий задание шестой лабораторной работы.

\* **@author** Slava Golowenko

\*/

**public** **class** Bouquet {

/\*\*

\* Метод, возвращающий результаты работы программы.

\*/

**public** **static** String printFlowerParametrs(Flower f){

**return** "Flower \"" + f.getNameOfFlower() + "\" cost " + f.getPrice() + " UAH" + ", her length is " + f.getLength() + " cm" + ", her freshness is " + f.getFreshness() + " days.";

}

/\*\*

\* Метод, выводящий результы работы программы на консоль.

\*/

**public** **static** **void** outputArray(Flower[] bouquet){

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

System.*out*.println(*printFlowerParametrs*(bouquet[i]));

}

}

/\*\*

\* Метод поиска цветка по длинне.

\*/

**public** **static** Flower searchByFlowerLength(Flower [] bouquet, **double** length1,**double** length2){

Flower result = bouquet[0];

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

**if** (bouquet[i].getLength() >= length1 && bouquet[i].getLength() <= length2)

result = bouquet[i];

}

**return** result;

}

/\*\*

\* Метод, который считает сумму за букет цветов.

\*/

**public** **static** **int** totalBouquetPrice(Flower [] bouquet) {

**int** totalPrice = 0;

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

totalPrice += bouquet[i].getPrice();

}

System.*out*.println("Total bouquet price: " + totalPrice + " UAH");

**return** totalPrice;

}

/\*\*

\* Метод сортировки цветов по свежести.

\*/

**public** **static** **void** sortByFreshness(Flower[] bouquet, **int** k) {

System.*out*.println("Bouquet is sorted by decrision of freshness:");

Flower buf;

**for** (**int** i = 0; i <= k - 1; i++) {

**for** (**int** j = 0; j <= k - 1; j++) {

**if** (bouquet[i].getFreshness() > bouquet[j].getFreshness()) {

buf = bouquet[i];

bouquet[i] = bouquet[j];

bouquet[j] = buf;

}

}

}

}

/\*\*

\* Основной класс.

\*/

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

**final** **int** count = 7;

/\*\*

\* Массив цветов.

\*/

Flower bouquet[] = **new** Flower[count];

bouquet[0] = **new** Rose(3,50.5,80);

bouquet[1] = **new** Rose(4,49.6,80);

bouquet[2] = **new** Rose(3,48.0,80);

bouquet[3] = **new** Orchid(2,28.3,120);

bouquet[4] = **new** Orchid(2,27.9,110);

bouquet[5] = **new** Lily(1,15.4,50);

bouquet[6] = **new** Rose();

/\*\*

\* Вывод массива цветов на консоль.

\*/

System.*out*.println("Bouqeot consist of:");

*outputArray*(bouquet);

/\*\*

\* Вывод отсортированного по свежести массива цветов.

\*/

System.*out*.println();

*sortByFreshness*(bouquet, count);

*outputArray*(bouquet);

/\*\*

\* Вывод на консоль результата из поиска цветка по длинне.

\*/

System.*out*.println();

System.*out*.println("Serched Flower by length:");

System.*out*.println(*printFlowerParametrs*(*searchByFlowerLength*(bouquet, 27.0, 28.0)));

System.*out*.println();

*totalBouquetPrice*(bouquet);

}

}

**Клас квітка:**

/\*\*Класс является родительским классом цветов.

\* **@author** Slava Golowenko

\*/

**public** **class** Flower {

/\*\*

\* Свежесть цветка, указывается в количестве дней хранения.

\*/

**private** **int** freshness;

/\*\*

\* Длинна цветка, указывается в сантиметрах.

\*/

**private** **double** length;

/\*\*

\* Цена цветка, указывается в гривнах.

\*/

**private** **int** price;

/\*\*

\* Название цветка.

\*/

**private** String nameOfFlower;

/\*\*

\* Максимальный срок хранения (свежесть) цветка.

\*/

**private** **int** MAX\_FRESHNESS = 5;

/\*\*

\* Максимальная длинна цветка.

\*/

**private** **double** MAX\_LENGTH = 100;

/\*\*

\* Максимальная цена цветка.

\*/

**private** **int** MAX\_PRICE = 120;

/\*\*

\* Гэтер возвращает свежесть (срок хранения) цветка.

\* **@return** сколько дней хранится цветок.

\*/

**public** **int** getFreshness(){

**return** freshness;

}

/\*\*

\* Гэтер возвращает длинну цветка.

\* **@return** длинна цветка.

\*/

**public** **double** getLength(){

**return** length;

}

/\*\*

\* Гэтер возвращает цену цветка.

\* **@return** цена цветка.

\*/

**public** **int** getPrice(){

**return** price;

}

/\*\*

\* Гэтер возвращает название цветка.

\* **@return** название цветка.

\*/

**public** String getNameOfFlower(){

**return** nameOfFlower;

}

/\*\*

\* Сэтер устанавливает свежесть цветка.

\* **@param** f устанавливает свежесть цветка.

\*/

**public** **void** setFreshness(**int** f){

**if** (f <= MAX\_FRESHNESS){

freshness = f;

}

}

/\*\*

\* Сэтер устанавливает длинну цветка.

\* **@param** l устанавливает длинну цветка.

\*/

**public** **void** setLength(**double** l){

**if** (l <= MAX\_LENGTH){

length = l;

}

}

/\*\*

\* Сэтер устанавливает цену цветка.

\* **@param** p устанавливает цену цветка.

\*/

**public** **void** setPrice(**int** p){

**if** (p <= MAX\_PRICE){

price = p;

}

}

/\*\*

\* Сэтер устанавливает название цветка.

\* **@param** fl название цветка.

\*/

**public** **void** setNameOfFlower(String fl){

nameOfFlower = fl;

}

/\*\*

\* Конструктор без параметров, устаналивает значения полей класса, ввёденных в конструкторе.

\*/

**public** Flower(){

freshness = 5;

length = 25.5;

price = 100;

nameOfFlower = "Hrizantema";

}

/\*\*

\* Конструктор с параметрами, устанавливает значения полей класса, ввёденных как параметры конструктора при его вызове.

\* **@param** f Свежесть цветка.

\* **@param** l Длинна цветка.

\* **@param** p Цена цветка.

\*/

**public** Flower(**int** f, **double** l, **int** p){

**if** (f <= MAX\_FRESHNESS) freshness = f;

**if** (l <= MAX\_LENGTH) length = l;

**if** (p <= MAX\_PRICE) price = p;

nameOfFlower = "Hrizantema";

}

/\*\*

\* Метод возвращает цену букета.

\* **@return** возвращает цену букета.

\*/

**public** **int** bouquetTotalPrice(){

**return** 3;

}

}

**Клас лілія:**

/\*\*Дочерный класс по отношению к классу Flower.

\* **@author** Slava Golowenko

\*/

**public** **class** Lily **extends** Flower{

/\*\*

\* Унаследованные конструкторы от класса Flower.

\*/

**public** Lily(){

**super**();

setNameOfFlower("Lily");

}

**public** Lily(**int** f, **double** l, **int** p){

**super**(f, l, p);

setNameOfFlower("Lily");

}

/\*\*

\* Переопределённый метод класса Flower, который возвращает цену цветка.

\*/

@Override

**public** **int** bouquetTotalPrice(){

**int** totalPrice = getPrice();

**return** totalPrice;

}

}

**Клас орхідея:**

/\*\*Дочерный класс по отношению к классу Bouquet.

\* **@author** Slava Golowenko

\*/

**public** **class** Orchid **extends** Flower{

/\*\*

\* Унаследованные конструкторы от класса Flower.

\*/

**public** Orchid(){

**super**();

setNameOfFlower("Orchid");

}

**public** Orchid(**int** f, **double** l, **int** p){

**super**(f, l, p);

setNameOfFlower("Orchid");

}

/\*\*

\* Переопределённый метод класса Flower, который возвращает цену цветка.

\*/

@Override

**public** **int** bouquetTotalPrice(){

**int** totalPrice = getPrice();

**return** totalPrice;

}

}

**Клас троянда:**

/\*\*Дочерный класс по отношению к классу Bouquet.

\* **@author** Slava Golowenko

\*/

**public** **class** Rose **extends** Flower{

/\*\*

\* Унаследованные конструкторы от класса Flower.

\*/

**public** Rose(){

**super**();

setNameOfFlower("Rose");

}

**public** Rose(**int** f, **double** l, **int** p){

**super**(f, l, p);

setNameOfFlower("Rose");

}

/\*\*

\* Переопределённый метод класса Flower, который возвращает цену цветка.

\*/

@Override

**public** **int** bouquetTotalPrice(){

**int** totalPrice = getPrice();

**return** totalPrice;

}

}

**Опис роботи програми:**

Дана програма складається з 5 класів. Клас Flower має 3 дочірні класи – Lily (лілія), Orchid (орхідея), Rose (троянда). В класі Bouquet знаходиться масив об’єктів класу квітка, в який ініціалізуються різні види квітів. В цьому ж класі реалізовано пошук елемента по заданому діапазону довжин та сортування елементів за рівнем свіжості. В основному класі Bouquet знаходиться виконавчий метод, в якому підраховується ціна букета і проводиться пошук та сортування.

**Висновок:**

В даній лабораторній роботі ми ознайомились з принципами наслідування та поліморфізму в мові програмування Java. Виконавши дану роботу я вивчив механізм наслідування та принципи поліморфізму.