# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет “Львівська політехніка”**



**Інститут післядипломної освіти**

**ЗВІТ**

**Про виконання лабораторної роботи №3**

**«Наслідування. Створення та використання ієрархії класів»**

**з дисципліни «Об’єктно-орієнтоване програмування»**

Виконав:

слухач групи ПЗС-11

Гринчук Тарас

Прийняла:

доц. Кортєєва Т.О.

« »\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 р.

∑ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

ЛЬВІВ – 2014

**Тема роботи**: Наслідування. Створення та використання ієрархії класів.

## 1. Завдання

Розробити ієрархію класів згідно отриманого завдання (таблиця 1) та описати їх алгоритмічною мовою С++. Використати абстрактні класи і віртуальні функції. Написати фрагмент програми з використанням об’єктів створених класових типів.

**Варіант 5.** Побутові прилади: пральна машина, пилосос, телевізор, магнітофон, радіоприймач, музичний центр.

## 2. Структурна схема ієрархії класів



Рис. 2.1. Структурна схема ієрархії класів

**3. Текст програми на мові програмування JAVA**

/\*\*

\* Л/р № 3 (варіант 5)

\* Наслідування. Створення та використання ієрархії класів.

\* @author Taras

\*/

package lab3\_oop;

import java.util.\*;

/\* --------------------------------------------------------------------------

\* клас Лічильник електроенергії - підраховує к-сть використаних кВт\*год

\* при роботі електропристроїв, в залежності від часу роботи

\* та їх потужності.

\* Клас має лише один екземпляр і створюється "на вимогу", реалізовано на базі

\* патерна Одинак (Singleton)

\*/

class ElectricityCounter {

public static ElectricityCounter value() {

return ec;

}

//збільження лічильника

public void add(double kilowatts, double hours) {

kilowattHours += kilowatts \* hours;

}

//показати поточний стан

public double indicate() {

return kilowattHours;

}

private double kilowattHours = 0;

//приватний конструктор, екземпяр створюється тільки через статичний виклик

private ElectricityCounter() {}

private static ElectricityCounter ec = new ElectricityCounter();

}

/\* --------------------------------------------------------------------------

\* клас Мотор - входить у склад кожного Побутового пристрою

\*/

class Engine {

//потужність

public double power;

//конструктор

public Engine(double power) {

this.power = power;

}

//запуск

public void start() {

System.out.println("Start engine...");

}

//припинення роботи

public void stop() {

System.out.print("Stop engine.");

}

}

/\* --------------------------------------------------------------------------

\* абстрактний клас Побутових пристроїв - батьківський для всіх інших

\*/

abstract class Appliance {

//поля-характеристики

protected String type; //тип

private String manufacturer; //виробник

private String model; //модель

private Engine eng; //мотор

//конструктор

public Appliance(double power, String type, String manufacturer, String model) {

eng = new Engine(power);

this.type = type;

this.manufacturer = manufacturer;

this.model = model;

}

//перевизначення стандартного методу Object.toString() - використовуємо

//для виведення на екран назви об'єкту з його характеристиками

@Override

public String toString() {

return type + ": " + manufacturer + " " + model + " / " + eng.power + " kWt";

}

//запуск пристрою

public void start() {

eng.start();

}

//вимкнення пристрою

public void stop(double hours) {

eng.stop();

System.out.println(" Time of work: " + hours + " hours.");

ElectricityCounter.value().add(eng.power, hours);

}

//абстрактний (віртуальний) метод. Реалізований для кожного виду

//побутового пристрою по-різному

abstract void work();

}

/\* -------------------------------------------------------------------------

\* клас Сервісні прилади - дочірній до Побутових пристроїв

\* використовується для розширення інтерфейсу батьківського класу

\*/

abstract class ServiceEquipment extends Appliance {

//конструктор

public ServiceEquipment(double power, String type, String manufacturer, String model) {

//виклик конструктора батьківського класу

super(power, type, manufacturer, model);

}

//"Очистка приладу після роботи"

//абстрактний (віртуальний) метод, для кожного конкретного виду

//сервісної техніки, передбачено своя реалізація цього методу

abstract void CleanItself();

}

/\* --------------------------------------------------------------------------

\* клас Інформаційна(розважальна) техніка - дочірній до Побутових пристроїв

\* використовується для розширення інтерфейсу батьківського класу

\*/

abstract class InformationalTechnics extends Appliance {

//рівень звуку

private int volumeLevel;

//конструктор

public InformationalTechnics(double power, String type, String manufacturer, String model) {

//виклик конструктора батьківського класу

super(power, type, manufacturer, model);

//початкове значення рівня звуку

volumeLevel = 10;

}

//встановлення рівня звуку

public void setVolumeLevel(int volumeLevel) {

this.volumeLevel = volumeLevel;

}

}

/\* --------------------------------------------------------------------------

\* клас Пральна машина - дочірній до Сервісних приладів

\*/

class WashingMachine extends ServiceEquipment {

//місткість баку з водою

private int waterTankCapacity;

//конструктор

public WashingMachine(double power, String manufacturer, String model, int waterTankCapacity) {

//виклик конструктора батьківського класу

super(power, "Washing machine", manufacturer, model);

this.waterTankCapacity = waterTankCapacity;

}

//до батьківського представлення назви додамо ще місткість баку

@Override

public String toString() {

return super.toString() + " / " + waterTankCapacity + " Liters";

}

//прання

public void wash() {

System.out.println("Washing...");

}

//викручування

public void squeeze() {

System.out.println("Squeezing...");

}

//реалізація віртуального методу класу Побутових пристроїв

@Override

public void work() {

wash();

squeeze();

}

//реалізація віртуального методу класу Сервісних приладів

@Override

public void CleanItself() {

//очистка баку з водою

System.out.println("Cleaning water from the tank.");

}

}

/\* --------------------------------------------------------------------------

\* клас Пилосос - дочірній до Сервісних приладів

\*/

class VacuumCleaner extends ServiceEquipment{

//довжина шлангу

private int cordLength;

//конструктор

public VacuumCleaner(double power, String manufacturer, String model, int cordLength) {

//виклик конструктора батьківського класу

super(power, "Vacuum cleaner", manufacturer, model);

this.cordLength = cordLength;

}

//до батьківського представлення назви додамо ще довжину шлангу

@Override

public String toString() {

return super.toString() + " / " + cordLength + " meters";

}

//метод "пилососити"

public void clean() {

System.out.println("Cleaning...");

}

//реалізація віртуального методу класу Побутових пристроїв

@Override

public void work() {

clean();

}

//реалізація віртуального методу класу Сервісних приладів

@Override

public void CleanItself() {

//очистити мішок з сміттям

System.out.println("Cleaning garbage collector.");

}

}

/\* --------------------------------------------------------------------------

\* клас Радіо - дочірній до Інформаційної техніки

\*/

class Radio extends InformationalTechnics {

//діапазон хвиль

private String frequencyRange;

//конструктор

public Radio(double power, String manufacturer, String model, String frequencyRange) {

//виклик конструктора батьківського класу

super(power, "Radio", manufacturer, model);

this.frequencyRange = frequencyRange;

}

//до батьківського представлення назви додамо ще діапазон хвиль

@Override

public String toString() {

return super.toString() + " / " + frequencyRange;

}

//програвати радіохвилю

public void playRadio() {

System.out.println("Playing radio...");

}

//реалізація віртуального методу класу Побутових пристроїв

@Override

public void work() {

playRadio();

}

}

/\* --------------------------------------------------------------------------

\* клас Телевізор - дочірній до Інформаційної техніки

\*/

class TV extends InformationalTechnics {

//довжина діагоналі в дюймах

private int diagonal;

//конструктор

public TV(double power, String manufacturer, String model, int diagonal) {

//виклик конструктора батьківського класу

super(power, "TV-set", manufacturer, model);

this.diagonal = diagonal;

}

//до батьківського представлення назви додамо ще довжину діагоналі

@Override

public String toString() {

return super.toString() + " / " + diagonal + " inch";

}

//показати телеканал

public void playTV() {

System.out.println("Playing TV channel...");

}

//телетекст

public void displayTeleText() {

System.out.println("Displaying teletext...");

}

//реалізація віртуального методу класу Побутових пристроїв

@Override

public void work() {

playTV();

displayTeleText();

}

}

/\* --------------------------------------------------------------------------

\* клас Магнітофон - дочірній до Радіо

\*/

class TapeRecorder extends Radio {

//кількість касетників

private int cassetteNumber;

//конструктор

public TapeRecorder(double power, String manufacturer, String model,String frequencyRange, int cassetteNumber) {

//виклик конструктора батьківського класу

super(power, manufacturer, model, frequencyRange);

//переприсвоїмо тип на потрібний

type = "Tape recorder";

this.cassetteNumber = cassetteNumber;

}

//до батьківського представлення назви додамо ще кількість касетників

@Override

public String toString() {

return super.toString() + " / " + cassetteNumber + " tape(s)";

}

//програвати запис на касеті

public void playTapes() {

System.out.println("Playing tapes...");

}

//реалізація віртуального методу класу Побутових пристроїв

@Override

public void work() {

//спочатку запустимо батьківський функціонал

super.work();

playTapes();

}

}

/\* --------------------------------------------------------------------------

\* клас Музичний центр - дочірній до Магнітофон

\*/

class StereoSystem extends TapeRecorder {

//кількість слотів для дисків

private int cdNumber;

//конструктор

public StereoSystem(double power, String manufacturer, String model, String frequencyRange, int cassetteNumber,

int cdNumber) {

//виклик конструктора батьківського класу

super(power, manufacturer, model, frequencyRange, cassetteNumber);

//переприсвоїмо тип на потрібний

type = "Stereo system";

this.cdNumber = cdNumber;

}

//до батьківського представлення назви додамо ще кількість СD-слотів

@Override

public String toString() {

return super.toString() + " / " + cdNumber + " cd's";

}

//програвати CD-диск

public void playCD() {

System.out.println("Playing cd's...");

}

//реалізація віртуального методу класу Побутових пристроїв

@Override

public void work() {

//спочатку запустимо батьківський функціонал

super.work();

playCD();

}

}

/\* ==========================================================================

\* головний клас пакету

\*/

public class Lab3\_OOP {

public static void main(String[] args) {

//Створення масиву Побутових пристроїв

Appliance[] app = {

new WashingMachine(0.95, "Bosch", "WAB 20071 CE", 42),

new VacuumCleaner(0.25, "Philips", "FC 9174", 9),

new Radio(0.01, "LG", "SP820", "AM+FM"),

new TV(0.1, "Sony", "KDL-42W807", 42),

new TapeRecorder(0.05, "Pioneer", "BDP-150", "FM", 2),

new StereoSystem(0.12, "Sharp", "GX-M10HOR", "AM+FM", 1, 4)

};

//отримання випадкових значень для вказання часу роботи пристроїв

Calendar c = Calendar.getInstance();

Random rand = new Random(c.getTimeInMillis());

//обхід масиву пристроїв

for(Appliance a : app) {

System.out.println("=========================================");

System.out.println(a);

//включимо пристрій

a.start();

//протестуємо весь функціонал пристрою

a.work();

//зупиняємо

//після проходження випадкової к-сті годин в рамках доби

a.stop(rand.nextInt(24) + 1);

//якщо пристрій належить до гілки Сервісних,

//очистимо його після роботи

if(a instanceof ServiceEquipment)

((ServiceEquipment) a).CleanItself();

//виведемо поточний стан індикатора лічильника електроенергії

System.out.println("----------------------------------------");

System.out.println("Electricity counter: "

+ ElectricityCounter.value().indicate() + " kWt\*hrs");

}

}

}

**4. Результат виконання програми**

Запустимо програму на виконання (рис. 4.1).

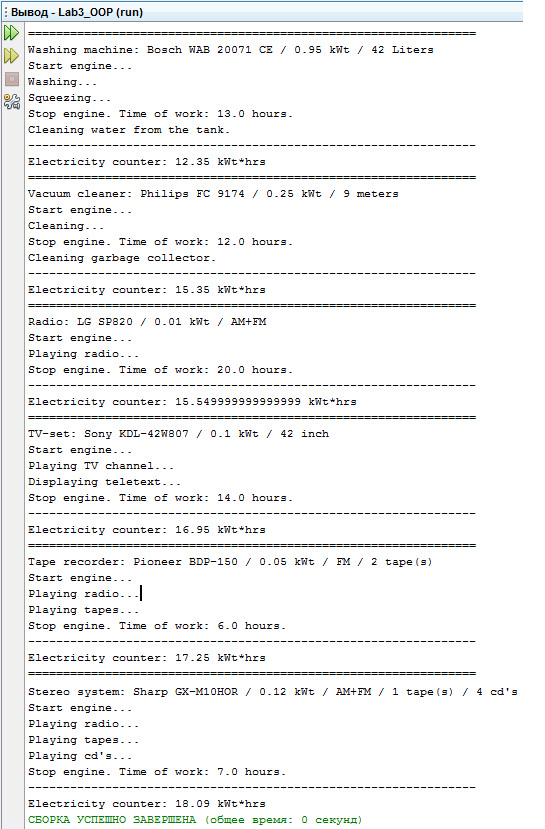


Рис. 4.1. Результат виконання програми

В результаті бачимо протокол роботи різних електричних пристроїв і збільшення показника електролічилька відсортовані, внаслідок цього.

## ВИСНОВКИ

На даній лабораторній роботі я навчився створювати ієрархічні структури класів, використовувати наслідування для написання ефективних програм на JAVA. Також вивчив особливості роботи з абстрактними класами та функціями, застосовуванню поліморфізму у прикладних заадчачах.