МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

**Лабораторна робота № 3**

з дисципліни « ООП на Java »

*назва дисципліни*

на тему: «ВИКОРИСТАННЯ ІНТЕРФЕЙСІВ У МОВІ JAVА»

Виконав: студент 2 курсу групи № 622п

освітньої програми

121 інженерія програмного забезпечення

(шифр і назва ОП)

Зайченко Ярослав Ігорович

(прізвище й ініціали студента)

Прийняв: доцент

Симбірський Генадій Дмитрович

(посада, науковий ступінь, прізвище й ініціали)

Кількість балів:

Харків – 2023

**ЗМІСТ**

[Постановка завдання 3](#_Toc158672321)

[Зміст звіту 3](#_Toc158672322)

[Порядок виконання роботи 3](#_Toc158672323)

[Виконання роботи 5](#_Toc158672324)

[Висновок 10](#_Toc158672325)

[Контрольні питання 11](#_Toc158672326)

Мета роботи: отримання практичних навичок із використання інтерфейсів при проектуванні об'єктно-орієнтованих програмних додатків мовою Java.

Постановка завдання

Розробити та реалізувати мовою Java програмний додаток для моделювання властивостей, поведінки та взаємодії об'єктів. Предметною областю для цього додатка є об'єкти, наведені у п.3 цієї роботи. Для заданої предметної області створити ієрархію класів.

У додатку продемонструвати використання механізмів створення та реалізації інтерфейсів для додання нової поведінки до об'єктів класів, успадкування інтерфейсів, а також використання інтерфейсів в якості посилального (об'єктного) типу та поліморфізму.

Порядок виконання роботи

1. Вивчити теоретичний матеріал, пов'язаний з темою даної лабораторної роботи, використовуючи конспект лекцій, літературу, що рекомендується, і документацію, що входить до складу пакету Java SE JDK.
2. Розробити концептуальну модель для заданої категорії об'єктів у вигляді словесного опису їх основних властивостей та поведінки.
3. Спроектувати абстрактний клас, у якому буде визначено найбільш загальні характерні властивості заданої категорії об'єктів.
4. Визначити найбільш характерні риси поведінки заданої категорії об'єктів і подати їх у вигляді набору методів, визначених в абстрактному класі.
5. Розробити підкласи для об'єктів, які успадковують клас узагальненого об'єкта (абстрактний) та реалізують поведінку (методи). Ці методи мають бути задекларовані у заздалегідь розроблених абстрактних класах або суперкласах.
6. Спроектувати два інтерфейси, в методах яких визначити на абстрактному рівні додаткову поведінку для деяких підкласів.
7. У розроблених підкласах реалізувати додаткову поведінку за допомогою спроектованих інтерфейсів
8. Відобразити структурні відносини між спроектованими класами та інтерфейсами за допомогою діаграми класів UML.
9. Реалізувати програмний код класів та інтерфейсів на мові Java. При реалізації програмного коду програми обов'язково використовувати коментарі та дескриптори документування для опису класів, конструкторів, методів та полів.
10. Розробити об'єкти для підкласів, для яких викликатимуться методи, що змінюють властивості та поведінку цих об'єктів. Продемонструвати роботу методів інтерфейсів, реалізованних у підкласах та їх поліморфного використання.
11. Запустити розроблену програму на виконання за допомогою середовища Intellij IDEA.
12. Протестувати розроблений додаток.
13. Отримати документацію на розроблений додаток за допомогою утиліти javadoc.exe.
14. Зробити висновки щодо роботи та відповісти на контрольні запитання.
15. Оформити звіт з лабораторної роботи відповідно до вимог, що пред'являються.
16. Захистити роботу.

Завдання для варіанту 10:

Принтери (матричні, струменеві, лазерні, світлодіодні). Властивості: тип моделі, формат паперу, вага, ціна, швидкість друку, режим роботи, тип інтерфейсу, стан. Поведінка: визначення та зміна властивостей.

Виконання роботи

Загальні відомості

Мова програмування – Java

Операційна система – Windows 11 Pro

Процесор – Apple Silicon M1 Pro with 6 CPU cores

Тип компілятора – IntelliJ IDEA 2024

Концептуальна модель об'єктів

Властивості:

* Тип моделі: тип транспортного засобу (наприклад, "автобус", "трамвай", "тролейбус").
* Фірма-виробник: компанія, яка виготовила принтер.
* Номер: унікальний ідентифікатор принтера.
* Маршрут: маршрут, за яким рухається принтер.
* Місткість: кількість пасажирів, які можуть перевозитися принтером.
* Закріплений оператор: водій або оператор, який керує принтером.

Поведінка:

* Визначення та зміна властивостей: методи для встановлення та зміни властивостей принтера, таких як тип моделі, фірма-виробник, номер, маршрут, місткість та закріплений оператор.
* Визначення стану: методи для визначення поточного стану принтера, наприклад, чи працює він, скільки паперу залишилося, чи є з'єднання з комп'ютером, і так далі.

Абстрактний клас, у якому визначено найбільш загальні характерні властивості заданої категорії об'єктів

Створений абстрактний клас з загальними характерними властивостями та поведінкою поданий в Додатку А.

Розроблені підкласи з об'єктами, які успадковують абстрактний клас та реалізують поведінку.

Створені класи транспорту автобус, тролейбус та трамвай подані в Додатках Б, В, Г.

При створенні підкласів об’єктів були перевизначені методи класу Object, а саме: equals(), hashCode() та toString(). Їх реалізацію можна переглянути у відповідних класах об’єктів.

Структурні відносини між спроектованими класами за допомогою діаграми класів UML.

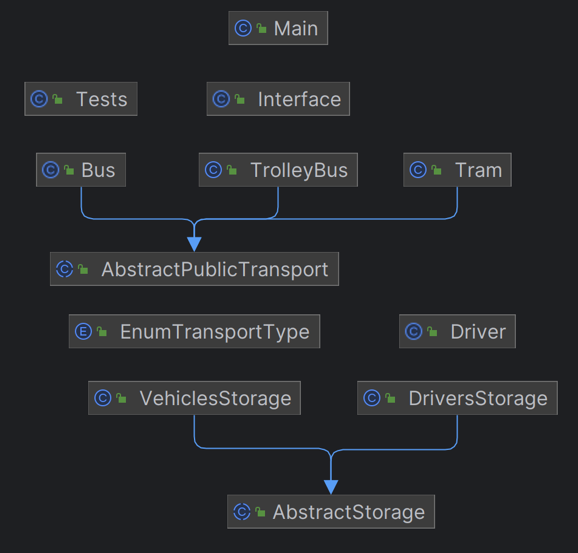


Рисунок 1 – згенерована діаграма у середовищі розробки

Реалізувати програмний код класів на мові Java.

При написанні коду були виконані умови: абстрактний клас з характерними властивостями та рисами поведінки. Підкласи об’єктів, які успадковують абстрактний клас та реалізовують поведінку об’єкту, перевизначені методи класу Object в кожному класі окремо. Фрагменти фунціоналу коду описані коментарями. Програмний код класів можна переглянути в додатках А-І.

Тестування програми

Тестувати будемо правильними значеннями. Всі тестування були виконані за допомогою ручного введення даних. Результат на рисунку 2-9.

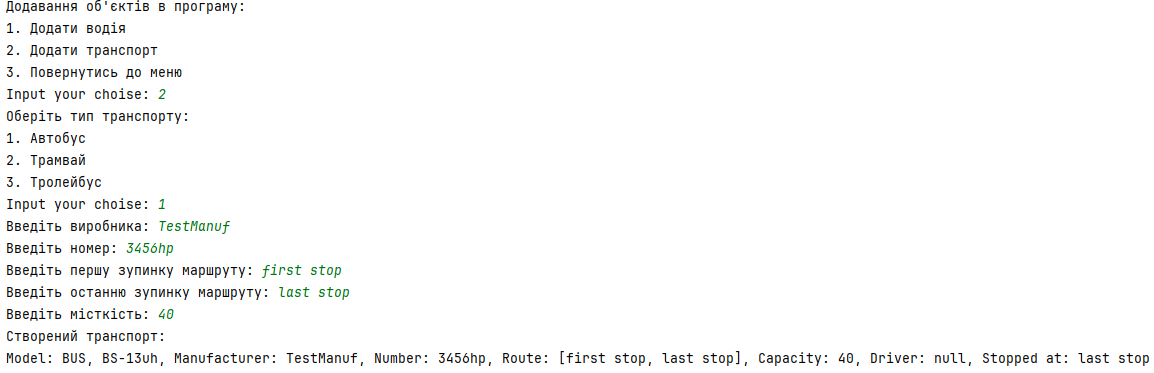


Рисунок 2 – створення об’єкту транспорту

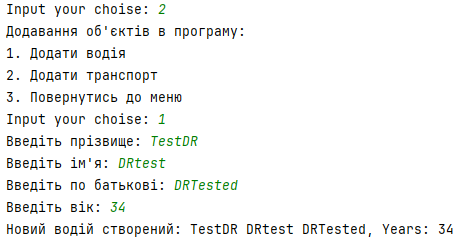


Рисунок 3 – створення водія

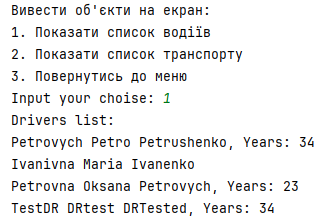


Рисунок 1 – перегляд водіїв

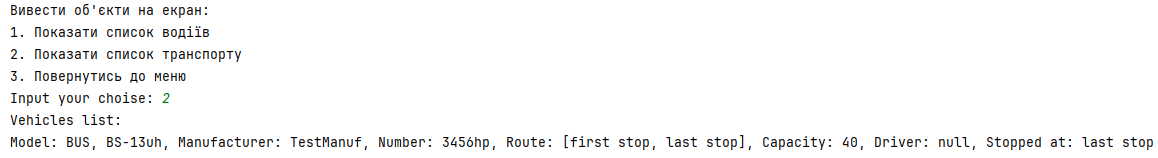


Рисунок 1 – перегляд транспорту

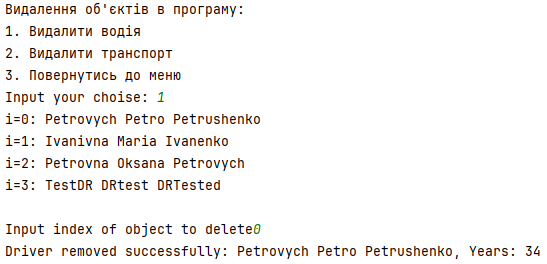


Рисунок 1 – видалення водія

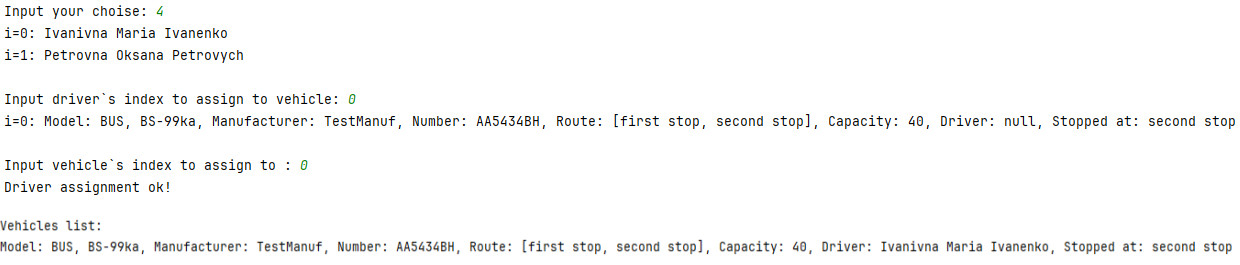


Рисунок 1 – додавання водія до транспорту

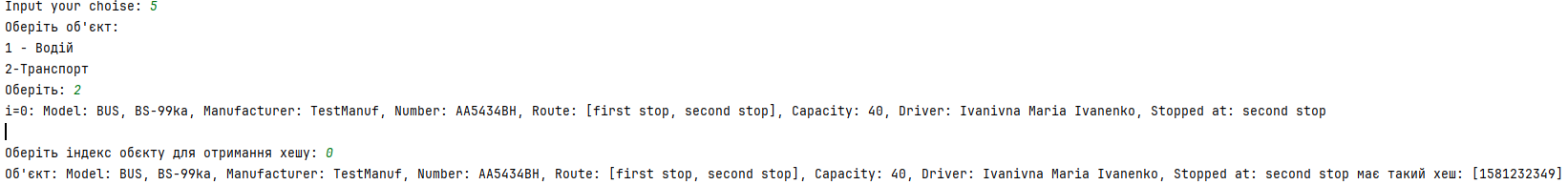


Рисунок 1 – отримання хешу об’єкта

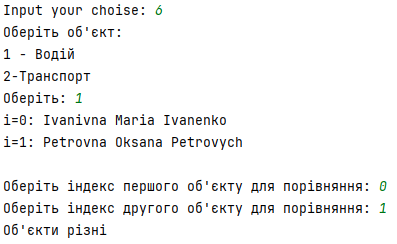


Рисунок 1 – порівняння об’єктів

Генерація документації на розроблений додаток

Для цього необхідно скористатись командою **javadoc**. В даному випадку вона буде написана таким чином: ***javadoc -d docs Main.java***. Результат виконання команди для створення документації зображено на рисунку 5.

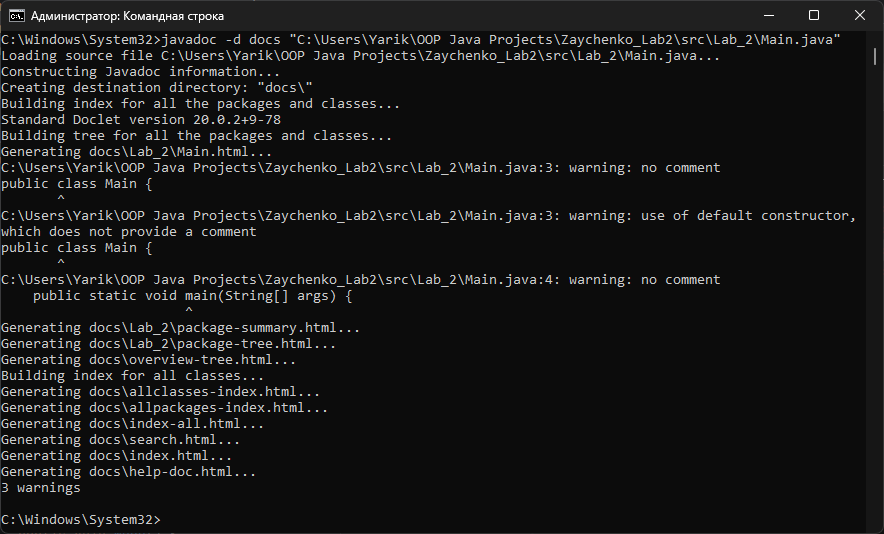


Рисунок 5 – створення документації додатку стандартними командами JRE

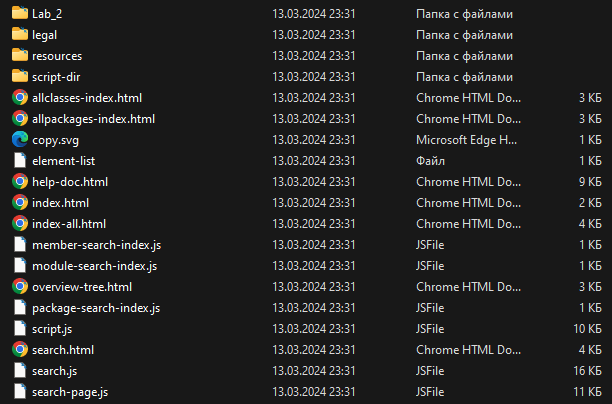


Рисунок 6 – отримані файли документації на додаток

Висновок

В ході даної лабораторної роботи було проведено дослідження теоретичного матеріалу, пов'язаного з програмуванням на мові Java та використанням його засобів для моделювання об'єктів та їх взаємодії. Згідно з вимогами завдання, спочатку була розроблена концептуальна модель для заданої предметної області, описуючи основні властивості та поведінку об'єктів. На основі цієї моделі було спроектовано абстрактний клас, що містив у собі найбільш загальні характеристики цих об'єктів.

Визначено основні риси поведінки об'єктів та представлено їх у вигляді набору методів, визначених у абстрактному класі. Після цього були реалізовані підкласи, які успадковували абстрактний клас та реалізували відповідну поведінку через перевизначення абстрактних методів.

У розроблених підкласах були перевизначені методи equals(), hashCode() та toString(), що дозволило правильно порівнювати об'єкти за їх вмістом, визначати унікальний ідентифікатор об'єкта та отримувати рядкове представлення об'єкту в зручному для виведення форматі.

Для відображення структурних відносин між класами та інтерфейсами була створена діаграма класів UML, що спрощує розуміння структури програми та взаємозв'язків між її складовими.

Після реалізації програмного коду була проведена компіляція та запуск програми для перевірки її працездатності. Розроблений додаток було протестовано, що дозволило переконатися в правильності його функціонування.

Завершальним етапом було отримання документації на розроблений додаток за допомогою утиліти **javadoc.exe**, що дозволяє автоматично генерувати документацію на основі коментарів у програмному коді. В результаті вдалої роботи над проектом було успішно досягнуто поставлених цілей та вимог завдання.

Контрольні питання

1. З якою метою застосовуються інтерфейси у мові Java?
2. Який синтаксис має оголошення інтерфейсу у мові Java?
3. Які модифікатори можливо використовувати для оголошення полів інтерфейсу?
4. Які модифікатори можливо використовувати для оголошення методів інтерфейсу?
5. Яку кількість інтерфейсів може успадковувати інтерфейс?
6. Яку кількість інтерфейсів може реалізувати клас?
7. З яким модифікатором повинен оголошуватися клас у якому визначені методи, які не мають реалізації?
8. Що може зберігатися в змінній, яка має тип інтерфейсу

*Додаток А*Машинний лістинг файлу **Main.java**

package Lab\_2;

import Lab\_2.Classes.Storages.DriversStorage;

import Lab\_2.Classes.Transport.Driver;

public class Main {

public static void main(String[] args) {

Interface ui = new Interface();

ui.Menu();

//Tests.TestActions();

//Lab\_2.Tests.testInterfaceMethods();

}

}

*Додаток Б*Машинний лістинг файлу **AbstractPublicTransport.java**

package Lab\_2.Classes.Transport;

import java.util.Arrays;

public abstract class AbstractPublicTransport {

// Властивості

protected EnumTransportType type;

protected String modelType; // модель

protected String manufacturer; // иробник

protected String number; // номер маршруту

protected String[] route = new String[2]; // маршрут

protected int capacity; // місткість

protected Driver driver; // закріплений водій

protected String stopped; // зупинка

// Конструктор

public AbstractPublicTransport(EnumTransportType tt, String modelType, String manufacturer, String number, String route\_start, String route\_end, int capacity) {

this.type = tt;

setModelType(modelType);

setManufacturer(manufacturer);

setNumber(number);

setRoute(route\_start, route\_end);

setCapacity(capacity);

this.stopped = route\_end;

}

// Методи доступу до властивостей

public String getModelType() {

return modelType;

}

public void setModelType(String modelType) {

this.modelType = modelType;

}

public String getManufacturer() {

return manufacturer;

}

public void setManufacturer(String manufacturer) {

this.manufacturer = manufacturer;

}

public String getNumber() {

return number;

}

public void setNumber(String number) {

this.number = number;

}

public String[] getRoute() {

return route;

}

public void setRoute(String route1, String route2) {

this.route[0] = route1;

this.route[1] = route2;

}

public int getCapacity() {

return capacity;

}

public void setCapacity(int capacity) {

this.capacity = capacity;

}

public Driver getAssignedDriver() {

return driver;

}

public void setAssignedDriver(Driver assignedDriver) {

this.driver = assignedDriver;

}

// Методи поведінки

public void Movement() // метод руху

{

if (this.stopped == this.route[0]) {

this.stopped = this.route[1];

} else if (this.stopped == this.route[1]) {

this.stopped = this.route[0];

}

}

public abstract void assignDriver(Driver driver);

public abstract void assignModel();

@Override

public String toString() {

return "Model: " + this.type.toString() + ", " + this.modelType + ", Manufacturer: " + this.manufacturer + ", Number: " + this.number + ", Route: " + Arrays.toString(this.route) + ", Capacity: " + this.capacity + ", Driver: " + this.driver + ", Stopped at: " + this.stopped;

}

}

*Додаток В*Машинний лістинг файлу **Bus.java**

package Lab\_2.Classes.Transport;

import Lab\_2.Classes.Storages.AbstractStorage;

import java.util.Arrays;

import java.util.Objects;

import java.util.Random;

import java.util.concurrent.ThreadLocalRandom;

public class Bus extends AbstractPublicTransport {

public Bus(String manufacturer, String number, String route\_start, String route\_end, int capacity, AbstractStorage storage) {

super(EnumTransportType.BUS, "", manufacturer, number, route\_start, route\_end, capacity);

assignModel();

storage.add(this);

}

@Override

public void assignDriver(Driver driver) {

this.driver=driver;

}

@Override

public void assignModel() {

// Створюємо рандомну цифру в діапазоні від 11 до 99

int randomNumber = ThreadLocalRandom.current().nextInt(11, 100);

// Створюємо рандомний рядок з латинських символів (a-z) довжиною 2 символи

StringBuilder randomLetters = new StringBuilder();

Random random = new Random();

for (int i = 0; i < 2; i++) {

char letter = (char) ('a' + random.nextInt(26));

randomLetters.append(letter);

}

String model = "BS-" + randomNumber + randomLetters.toString();

this.modelType=model;

}

// Перевизначення методу equals()

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Bus bus = (Bus) o;

return Objects.equals(this.type, bus.type) &&

Objects.equals(this.modelType, bus.modelType) &&

Objects.equals(this.manufacturer, bus.manufacturer) &&

Objects.equals(this.number, bus.number) &&

Arrays.equals(this.route, bus.route) &&

this.capacity == bus.capacity &&

Objects.equals(this.driver, bus.driver) &&

Objects.equals(this.stopped, bus.stopped);

}

// Перевизначення методу hashCode()

@Override

public int hashCode() {

return Objects.hash(this.type, this.modelType, this.manufacturer, this.number, Arrays.hashCode(this.route), this.capacity, this.driver, this.stopped);

}

}

*Додаток Г*Машинний лістинг файлу **Driver.java**

package Lab\_2.Classes.Transport;

import Lab\_2.Classes.Storages.AbstractStorage;

public class Driver {

protected int years;

private final String[] fullName = new String[3];

public Driver(String name, String secondName, String middleName, AbstractStorage storage) {

this(name, secondName, middleName, 0, storage);

}

// Конструктор з чотири параметри

public Driver(String name, String secondName, String middleName, int age, AbstractStorage storage) {

this.fullName[0] = secondName;

this.fullName[1] = name;

this.fullName[2] = middleName;

try {

setAge(age);

} catch (Exception e) {

throw new RuntimeException(e);

}

storage.add(this);

}

private void setAge(int age) throws Exception {

if (age == 0 )

this.years = age;

else if (age < 20 || age > 100)

throw new Exception("Invalid age parameters");

else this.years = age;

}

public int getAge() throws Exception {

if (this.years != 0)

return this.years;

else

throw new Exception("Age haven`t been set");

}

public String getFullInfo() {

if (this.years == 0) {

return this.toString();

} else return this.toString() + ", Years: " + this.years;

}

@Override

public String toString() {

return String.join(" ", this.fullName);

}

}

*Додаток Ґ*Машинний лістинг файлу **EnumTransportType.java**

package Lab\_2.Classes.Transport;

public enum EnumTransportType {

BUS,

TRAM,

TROLLEYBUS,

DEFAULT

}

*Додаток Д*Машинний лістинг файлу **Tram.java**

package Classes;

import Classes.Puppy;

import javax.swing.\*;

public class Testing {

public static void StartTest(String puppyName){

// Створення об'єкта класу Цуценя

Puppy TestMYpuppy = new Puppy(puppyName);

Actions(TestMYpuppy);

Diff\_Constructors();

}

private static void Actions(Puppy object)

{

// Виклик методів для демонстрації

object.askName();

object.makeSound();

object.jump();

object.run();

object.bite();

}

public static void Diff\_Constructors(){

// Створення об'єкта собаки з використанням різних значень статі

Puppy p1 = new Puppy("Бобік", 3, "brown", Gender.MALE);

Puppy p2 = new Puppy("Ляля", 2);

System.out.println(p1.getInfo());

System.out.println(p2.getInfo());

}

public static void AllTest() {

//перевірка імені

try {

System.out.println("Правильне ім'я конструктора");

Puppy a = new Puppy("ПравильнийЦуцик");

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Мало літер ім'я конструктора");

Puppy a = new Puppy("По");

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Багато літер ім'я конструктора - 24шт");

Puppy a = new Puppy("ПоПоПоПоПоПоПоПоПоПоПоПо");

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Не літери ім'я конструктора");

Puppy a = new Puppy("12\_34");

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

//перевірка віку

try {

System.out.println("Правильний вік");

Puppy a = new Puppy("Мінік",2);

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Вік нуль");

Puppy a = new Puppy("Мінік",0);

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

try {

System.out.println("Вік меньше нуля");

Puppy a = new Puppy("Мінік",-2);

System.out.println(a.getInfo());

System.out.println();

} catch (IllegalArgumentException e) {

System.out.println("Помилка: " + e.getMessage());

System.out.println();

}

}

}

*Додаток Е*Машинний лістинг файлу **TrolleyBus.java**

package Lab\_2.Classes.Transport;

import Lab\_2.Classes.Storages.AbstractStorage;

import java.util.Arrays;

import java.util.Objects;

import java.util.Random;

import java.util.concurrent.ThreadLocalRandom;

public class TrolleyBus extends AbstractPublicTransport {

public TrolleyBus(String manufacturer, String number, String route\_start, String route\_end, int capacity, AbstractStorage storage) {

super(EnumTransportType.TROLLEYBUS, "", manufacturer, number, route\_start, route\_end, capacity);

assignModel();

storage.add(this);

}

@Override

public void assignDriver(Driver driver) {

setAssignedDriver(driver);

}

@Override

public void assignModel() {

int randomNumber = ThreadLocalRandom.current().nextInt(500, 900);

StringBuilder randomLetters = new StringBuilder();

Random random = new Random();

for (int i = 0; i < 2; i++) {

char letter = (char) ('a' + random.nextInt(26));

randomLetters.append(letter);

}

String model = "TB-" + randomNumber + randomLetters.toString();

this.modelType = model;

}

// Перевизначення методу equals()

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

TrolleyBus bus = (TrolleyBus) o;

return Objects.equals(this.type, bus.type) &&

Objects.equals(this.modelType, bus.modelType) &&

Objects.equals(this.manufacturer, bus.manufacturer) &&

Objects.equals(this.number, bus.number) &&

Arrays.equals(this.route, bus.route) &&

this.capacity == bus.capacity &&

Objects.equals(this.driver, bus.driver) &&

Objects.equals(this.stopped, bus.stopped);

}

// Перевизначення методу hashCode()

@Override

public int hashCode() {

return Objects.hash(this.type, this.modelType, this.manufacturer, this.number, Arrays.hashCode(this.route), this.capacity, this.driver, this.stopped);

}

}

*Додаток Є*Машинний лістинг файлу **AbstractStorage.java**

package Lab\_2.Classes.Storages;

import java.util.ArrayList;

public abstract class AbstractStorage {

protected ArrayList<Object> list = new ArrayList<>();

// Метод для додавання об'єкта в список

public void add(Object item) {

list.add(item);

}

// Метод для видалення об'єкта зі списку

public void remove(int index) {

list.remove(index);

}

// Метод для отримання об'єкта за індексом

public Object get(int index) {

return list.get(index);

}

public int getCount() {

int counter = 0;

for (Object o : list) {

counter++;

}

return counter;

}

// Абстрактний метод, який потрібно реалізувати в підкласах

public abstract String getFullInformation();

}

*Додаток Ж*Машинний лістинг файлу **DriverStorage.java**

package Lab\_2.Classes.Storages;

import Lab\_2.Classes.Transport.Driver;

public class DriversStorage extends AbstractStorage {

public DriversStorage() {

super();

}

@Override

public String getFullInformation() {

//if (super.getCount()==0) throw new RuntimeException("List is empty");

// Виводимо інформацію про всіх водіїв, які містяться у списку

String res = "";

for (Object obj : super.list) {

Driver temp\_drv = (Driver) obj;

res += temp\_drv.getFullInfo() + '\n';

}

return res;

}

}

*Додаток З*Машинний лістинг файлу **VehicleStorage.java**

package Lab\_2.Classes.Storages;

public class VehiclesStorage extends AbstractStorage {

public VehiclesStorage() {

super();

}

// Перевизначення методу displayInformation

@Override

public String getFullInformation() {

// Виводимо інформацію про всі транспортні засоби, які містяться у списку

StringBuilder info = new StringBuilder();

for (Object temp\_vehicle : super.list){

info.append(temp\_vehicle.toString()).append('\n');

}

return info.toString();

}

}

*Додаток І*Машинний лістинг файлу **Interface.java**

package Lab\_2;

import Lab\_2.Classes.Storages.AbstractStorage;

import Lab\_2.Classes.Storages.DriversStorage;

import Lab\_2.Classes.Storages.VehiclesStorage;

import Lab\_2.Classes.Transport.\*;

import Lab\_2.Tests;

import java.util.Scanner;

public class Interface {

Scanner scanner = new Scanner(System.in);

DriversStorage dl = new DriversStorage();

VehiclesStorage vl = new VehiclesStorage();

Driver d5 = new Driver("Petro", "Petrovych", "Petrushenko", 34, dl);

Driver d6 = new Driver("Maria", "Ivanivna", "Ivanenko", dl);

Driver d7 = new Driver("Oksana", "Petrovna", "Petrovych", 23, dl);

public void Menu() {

while (true) {

try {

System.out.println("Меню:");

System.out.println("1. Показати список об'єктів");

System.out.println("2. Додати новий об'єкт");

System.out.println("3. Видалити об'єкт");

System.out.println("4. Додати водія до транспорту");

System.out.println("5. Отримати хеш об'єкту");

System.out.println("6. Порівнняння об'єктів");

//System.out.println("10. Напівавтоматична перевірка всіх методів підряд");

System.out.println("0. Вийти з програми");

//System.out.print("Виберіть пункт меню: ");

switch (ValidInput("Input your choise: ", 0, 10)) {

case 1:

showList();

break;

case 2:

addObj();

break;

case 3:

deleteObj();

break;

case 4:

giveDriver();

break;

case 5:

getHashedObj();

break;

case 6:

getComparedObj();

break;

case 10:

//Tests.TestActions();

Tests.testInterfaceMethods();

break;

case 0:

System.exit(0);

break;

default:

System.out.println("Невірний вибір. Спробуйте ще раз.");

Menu();

break;

}

} catch (Exception ex) {

System.out.println(ex.getMessage());

}

}

}

public void showList() {

while (true) {

System.out.println("Вивести об'єкти на екран:");

System.out.println("1. Показати список водіїв");

System.out.println("2. Показати список транспорту");

System.out.println("3. Повернутись до меню");

try {

switch (ValidInput("Input your choise: ", 1, 3)) {

case 1:

AllObjects(dl);

break;

case 2:

AllObjects(vl);

break;

case 3:

//break;

return;

default:

System.out.println("Невірний вибір. Спробуйте ще раз.");

Menu();

break;

}

} catch (Exception ex) {

System.out.println(ex.getMessage());

}

}

}

public void addObj() {

while (true) {

System.out.println("Додавання об'єктів в програму:");

System.out.println("1. Додати водія");

System.out.println("2. Додати транспорт");

System.out.println("3. Повернутись до меню");

try {

switch (ValidInput("Input your choise: ", 1, 3)) {

case 1:

CreateDriver(dl);

break;

case 2:

CreateVehicle(vl);

break;

case 3:

return;

default:

System.out.println("Невірний вибір. Спробуйте ще раз.");

Menu();

break;

}

} catch (Exception ex) {

System.out.println(ex.getMessage());

}

}

}

public void CreateDriver(AbstractStorage s) {

System.out.print("Введіть прізвище: ");

String lastName = scanner.nextLine();

System.out.print("Введіть ім'я: ");

String firstName = scanner.nextLine();

System.out.print("Введіть по батькові: ");

String middleName = scanner.nextLine();

int age = ValidInput("Введіть вік: ", 20, 100);

try {

Driver driver = new Driver(firstName, lastName, middleName, age, s);

System.out.println("Новий водій створений: " + driver.getFullInfo());

} catch (Exception e) {

System.out.println("Помилка при створенні водія: " + e.getMessage());

}

}

public void CreateVehicle(AbstractStorage s) {

AbstractPublicTransport vehicle;

EnumTransportType t = readTransportType();

System.out.print("Введіть виробника: ");

String manuf = scanner.nextLine();

System.out.print("Введіть номер: ");

String num = scanner.nextLine();

System.out.print("Введіть першу зупинку маршруту: ");

String r1 = scanner.nextLine();

System.out.print("Введіть останню зупинку маршруту: ");

String r2 = scanner.nextLine();

System.out.print("Введіть місткість: ");

int capac = Integer.parseInt(scanner.nextLine());

switch (t) {

case BUS:

vehicle = new Bus(manuf, num, r1, r2, capac, s);

break;

case TRAM:

vehicle = new Tram(manuf, num, r1, r2, capac, s);

break;

case TROLLEYBUS:

vehicle = new TrolleyBus(manuf, num, r1, r2, capac, s);

break;

default:

throw new RuntimeException("Unknown error occurred");

}

System.out.println("Створений транспорт: ");

System.out.println(vehicle.toString());

}

// Метод для вибору типу транспорту

private EnumTransportType readTransportType() {

System.out.println("Оберіть тип транспорту:");

System.out.println("1. Автобус");

System.out.println("2. Трамвай");

System.out.println("3. Тролейбус");

switch (ValidInput("Input your choise: ", 1, 3)) {

case 1:

return EnumTransportType.BUS;

case 2:

return EnumTransportType.TRAM;

case 3:

return EnumTransportType.TROLLEYBUS;

default:

return EnumTransportType.DEFAULT;

}

}

public void deleteObj() {

AbstractStorage st;

System.out.println("Видалення об'єктів в програму:");

System.out.println("1. Видалити водія");

System.out.println("2. Видалити транспорт");

System.out.println("3. Повернутись до меню");

int s = ValidInput("Input your choise: ", 1, 3);

switch (s) {

case 1:

System.out.println(OutputByIndexes(dl));

break;

case 2:

System.out.println(OutputByIndexes(vl));

break;

case 3:

break;

default:

System.out.println("Невірний вибір. Спробуйте ще раз.");

Menu();

break;

}

switch (s) {

case 1:

st = dl;

break;

case 2:

st = vl;

break;

default:

return;

}

int ch = ValidInput("Input index of object to delete", 0, (st.getCount() - 1));

removeObject(st, ch);

}

public void giveDriver() {

System.out.println(OutputByIndexes(dl));

int d\_indx = ValidInput("Input driver`s index to assign to vehicle: ", 0, dl.getCount() - 1);

Driver drv = (Driver) dl.get(d\_indx);

System.out.println(OutputByIndexes(vl));

int v\_indx = ValidInput("Input vehicle`s index to assign to : ", 0, vl.getCount() - 1);

AbstractPublicTransport v = (AbstractPublicTransport) vl.get(v\_indx);

setDriver(v, drv);

}

public String OutputByIndexes(AbstractStorage s) {

if (s.getCount() == 0) throw new RuntimeException("List is empty!");

String str = "";

for (int i = 0; i < s.getCount(); i++) {

str += "i=" + i + ": " + s.get(i).toString() + '\n';

}

return str;

}

// Метод для перевірки, чи введене значення є цілим числом та в межах встановленого діапазону

public int ValidInput(String inputTxt, int low, int up) {

System.out.print(inputTxt);

int input = Integer.MAX\_VALUE;

do {

input = Integer.parseInt(scanner.nextLine());

if (input >= low && input <= up) {

return input;

} else {

throw new RuntimeException("Invalid input, try again");

}

} while (true);

}

// Виводимо повну інформацію

public static void AllObjects(AbstractStorage s) {

if (s.getCount() == 0) throw new RuntimeException("Список пустий!");

if (s instanceof DriversStorage) {

System.out.println("Drivers list:");

} else if (s instanceof VehiclesStorage) {

System.out.println("Vehicles list:");

}

System.out.println(s.getFullInformation());

}

public static void removeObject(AbstractStorage s, int index) {

Object removedObject = s.get(index); // Отримати видалений об'єкт

s.remove(index); // Видалити об'єкт зі списку

if (s instanceof DriversStorage) {

Driver rd = (Driver) removedObject;

System.out.println("Driver removed successfully: " + rd.getFullInfo()); // Вивести інформацію про видаленого водія

} else if (s instanceof VehiclesStorage) {

AbstractPublicTransport rv = (AbstractPublicTransport) removedObject;

System.out.println("Vehicle removed successfully: " + rv); // Вивести інформацію про видалений транспортний засіб

}

}

public static void setDriver(AbstractPublicTransport v, Driver drv) {

v.assignDriver(drv);

System.out.println("Driver assignment ok!");

}

public void getComparedObj() {

System.out.println("Оберіть об'єкт:\n1 - Водій\n2-Транспорт");

AbstractStorage temp = null;

switch (ValidInput("Оберіть: ", 1, 2)) {

case 1:

temp = dl;

break;

case 2:

temp = vl;

break;

}

System.out.println(OutputByIndexes(temp));

Object obj1 = temp.get(ValidInput("Оберіть індекс першого об'єкту для порівняння: ", 0, temp.getCount() - 1));

Object obj2 = temp.get(ValidInput("Оберіть індекс другого об'єкту для порівняння: ", 0, temp.getCount() - 1));

boolean res = CompareObj(obj1, obj2);

if (res == true) {

System.out.println("Об'єкти однакові");

} else {

System.out.println("Об'єкти різні");

}

}

public static boolean CompareObj(Object o1, Object o2) {

if (o1 == null || o2 == null) {

return false; // Если хотя бы один из объектов null, они не равны

}

return o1.equals(o2); // Сравниваем содержимое объектов

}

public void getHashedObj() {

System.out.println("Оберіть об'єкт:\n1 - Водій\n2-Транспорт");

AbstractStorage temp = null;

switch (ValidInput("Оберіть: ", 1, 2)) {

case 1:

temp = dl;

break;

case 2:

temp = vl;

break;

}

System.out.println(OutputByIndexes(temp));

Object obj = temp.get(ValidInput("Оберіть індекс обєкту для отримання хешу: ", 0, temp.getCount() - 1));

System.out.println("Об'єкт: " + obj.toString() + " має такий хеш: [" + getHash(obj) + "]");

}

public static String getHash(Object o) {

String hash = Integer.toString(o.hashCode());

return hash;

}

}