## Міністерство освіти і науки України

# Національний університет “Львівська політехніка”

Кафедра інформаційних систем та мереж



Прикладне програмування

Звіт

До лабораторної роботи № 2

«Основи мови програмування Java та Java Collections»

Виконав:

ст. гр. ІТ–22

Гук П. М.

Прийняла:

Завущак І. І.

«\_\_\_» \_\_\_\_\_\_ 2020 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2020

**Мета**​: ознайомитися із основами ООП мови програмування Java та з Java

Collections.

Хід роботи

**Завдання No 1.**

Реалізувати ієрархію класів Тварини(Animal) та Їжа(Food). Тварини

поділяються на травоїдних та м’ясоїдних. Їжа поділяється відповідно.

● Всеїдних тварин не розглядаємо.

● Тварини повинні мати метод boolean eat(...), який визначає чи може

тварина з’їсти задану їжу.

● Тварини повинні мати метод run() (виводить швидкість бігу тварини)

● Продемонструвати поліморфізм на основі методу run та масиві об’єктів

● Кожний створений клас повинен мати мінімум одне поле та конструктор для усіх полів.

**Код файлу Animal.java**

package task1;

abstract class Animal {

private final int speed;

private final Food.Type type;

public Animal(int speed, Food.Type type) {

this.type = type;

this.speed = speed;

}

public boolean eat(Food food) {

return food.getType() == this.type;

}

public void run() {

System.out.println("Running with speed: " + this.speed);

}

}

class HerbivorousAnimal extends Animal {

public HerbivorousAnimal(int speed) {

super(speed, Food.Type.Plant);

}

}

class СarnivorousAnimal extends Animal {

public СarnivorousAnimal(int speed) {

super(speed, Food.Type.Meal);

}

}

**Код файлу Food.java**

package task1;

abstract class Food {

private final Food.Type type;

private final int weight;

public static enum Type {

Plant, Meal,

}

public Food(Food.Type type, int weight) {

this.type = type;

this.weight = weight;

}

public Type getType() {

return type;

}

}

class Plant extends Food {

public Plant(int weight) {

super(Type.Plant, weight);

}

}

class Meal extends Food {

public Meal(int weight) {

super(Type.Meal, weight);

}

}

**Код файлу Main.java**

package task1;

public class Main {

public static void main(String[] args) {

HerbivorousAnimal cow = new HerbivorousAnimal(1);

СarnivorousAnimal dog = new СarnivorousAnimal(10);

Plant nettle = new Plant(100);

Meal brisket = new Meal(300);

System.out.println("Cow eats plant: " + cow.eat(nettle));

System.out.println("Cow eats meal: " + cow.eat(brisket));

System.out.println("Dog eats plant: " + dog.eat(nettle));

System.out.println("Dog eats meal: " + dog.eat(brisket));

Animal[] animals = { cow, dog };

for (Animal animal: animals) animal.run();

}

}

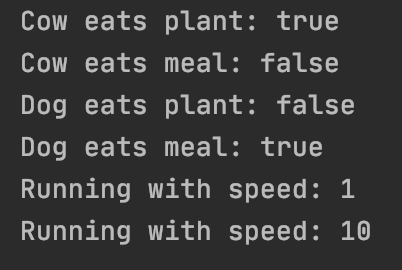
****

Рис.1 Результат виконання роботи програми

**Завдання No 2.**

Написати ієрархію класів та продемонструвати порядок ініціалізації

полів класу. Використати конструктор та статичні блоки ініціалізації в усіх класах (мінімум 3 класи)

**Код файлу Button.java**

package task2;

abstract class Button {

final int width;

final int height;

static{

System.out.println("Button static initializer");

}

Button(int width, int height) {

this.width = width;

this.height = height;

System.out.println("Button constructor");

}

}

class RoundButton extends Button {

final int borderRadius;

static{

System.out.println("RoundButton static initializer");

}

RoundButton(int width, int height, int borderRadius) {

super(width, height);

this.borderRadius = borderRadius;

System.out.println("RoundButton constructor");

}

}

class WindowsButton extends RoundButton {

final double opacity;

static{

System.out.println("WindowsButton static initializer");

}

WindowsButton(int width, int height, int borderRadius, double opacity) {

super(width, height, borderRadius);

this.opacity = opacity;

System.out.println("WindowsButton constructor");

}

}

**Код файлу Main.java**

package task2;

public class Main {

public static void main(String[] args) {

new WindowsButton(100, 30, 10, 0.9);

}

}

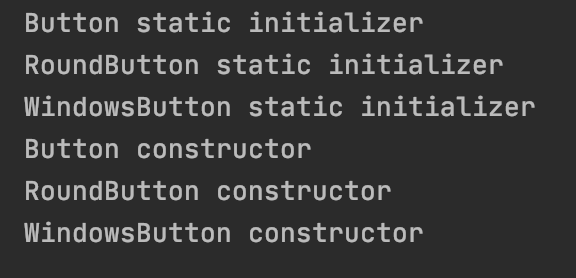


Рис.2 Результат виконання роботи програми

**Завдання No 3.**

Реалізувати структуру даних “черга” та методи add(...) i delete(...).

**Код файлу Queue.java**

package task3;

public class Queue<T> {

private Node tail;

private class Node {

final T value;

Node next;

final Node prev;

Node(T value) {

this(value, null, null);

}

Node(T value, Node next) {

this(value, next, null);

}

Node(T value, Node next, Node prev) {

this.next = next;

this.prev = prev;

this.value = value;

}

}

public void add(T item) {

tail = new Node(item, tail);

}

public void delete() throws Exception {

if (tail == null) {

throw new Exception("Queue is empty");

}

Node curr = tail;

Node prev = null;

while (curr.next != null) {

prev = curr;

curr = curr.next;

}

if (prev != null) {

prev.next = null;

} else {

tail = null;

}

}

public void print() {

System.out.print("Tail ");

if (tail == null) return;

Node temp = tail;

do {

System.out.print(temp.value + " ");

temp = temp.next;

} while (temp != null);

System.out.println();

}

**Код файлу Main.java**

package task3;

public class Main {

public static void main(String[] args) {

try {

Queue<Integer> queue = new Queue<Integer>();

queue.add(1);

queue.add(2);

queue.add(3);

queue.print();

queue.delete();

queue.print();

} catch (Exception ex) {

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

}

}

}

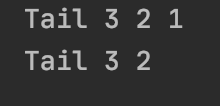
} 

Рис.3 Результат виконання роботи програми

**Завдання No 4.**

Продемонструвати поліморфізм із першого завдання

використовуючи List.

**Код файлу Main.java**

package task4;

import task1.Animal;

import task1.HerbivorousAnimal;

import task1.СarnivorousAnimal;

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

HerbivorousAnimal cow = new HerbivorousAnimal(1);

СarnivorousAnimal dog = new СarnivorousAnimal(10);

ArrayList<Animal> animals = new ArrayList<Animal>();

animals.add(cow);

animals.add(dog);

for (Animal animal: animals) animal.run();

}

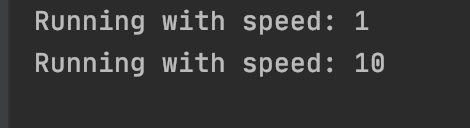
} 

Рис.4 Результат виконання роботи програми

**Завдання № 5.**

Згенерувати масив зі 100 чисел від 1 до 7. Добавити усі числа у Set

та вивести результат.

**Код файлу Main.java**

package task5;

import java.util.HashSet;

public class Main {

public static void main(String[] args) {

int[] numbers = new int[100];

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

numbers[i] = (int)(Math.random() \* 8);

}

System.out.println("Array:");

for (int item: numbers) {

System.out.print(item + " ");

}

HashSet<Integer> uniqueNumbers = new HashSet<Integer>();

for (int item: numbers) uniqueNumbers.add(item);

System.out.println("Set:");

for (int item: uniqueNumbers) {

System.out.print(item + " ");

}

}

}

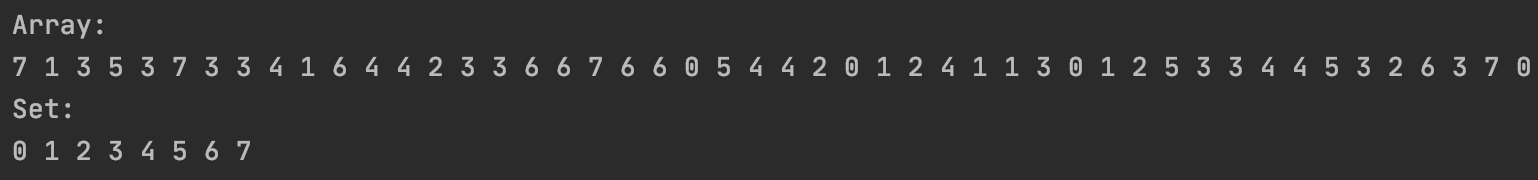
****

Рис.5 Результат виконання роботи програми

**Завдання No 6.**

Створити Map використовуючи як ключ ім’я користувача (User із

лабораторної роботи No 1), а значення - об’єкт User. Продемонструвати додавання, видалення та додавання різних об’єктів з однаковим ключем.

**Код файлу Main.java**

package task6;

import java.util.HashMap;

import java.util.Map;

public class Main {

public static void main(String[] args) {

User user1 = new User("Pavlo", "Huk", "pavlo@gmail.com", 19);

User user2 = new User("Pavlo", "Huk", "pavlo.huk@gmail.com", 19);

Map<String, User> users = new HashMap();

System.out.println("Only one user:");

users.put(user1.firstName, user1);

printMap(users);

System.out.println("Two users:");

users.put(user2.firstName, user2);

printMap(users);

System.out.println("After remove:");

users.remove(user2.firstName);

printMap(users);

}

static void printMap(Map<String, User> map) {

for(Map.Entry<String, User> item : map.entrySet()){

System.out.printf("Key: %s Value: %s \n", item.getKey(), item.getValue().firstName);

}

}

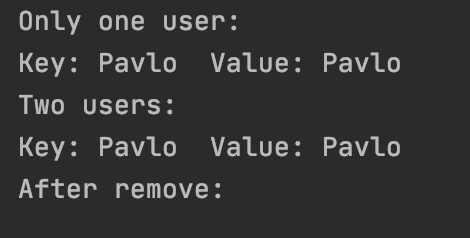
} 

Рис.6 Результат виконання роботи програми

**Висновок:** на даній лабораторній роботі я ознайомився із основами ООП мови програмування Java та з Java Collections.