**МІНІСТЕРСТВО ОСВІТИ І НАУКИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Інститут: **ІКНІ**

Кафедра: **ІСМ**



**ЗВІТ**

До лабораторної роботи №2

**З дисципліни:** «Прикладне програмування»

**Виконав**:  
ст.гр. ІТ-21

Коменда О.С.

**Прийняв:**

Тихонов В.А.

«\_ »\_\_\_\_\_\_\_\_2021 р.

Львів – 2021

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

**Хід роботи**

**Завдання № 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 програми:**

package task1;  
  
public class Task1 {  
 public static void main(String[] args) {  
 Animal[] animals = new Animal[] {  
 new Carnivorous("Jaguar", 80),  
 new Carnivorous("Puma", 75),  
 new Herbivorous("Equus", 88),  
 new Herbivorous("Rabbit", 20)  
 };  
  
 Food[] food = new Food[] {  
 new Herb("grass"),  
 new Meat("meat"),  
 new Herb("roots")  
 };  
  
 for(Animal a : animals) {  
 a.run();  
 for(Food f : food) {  
 System.out.println(a.name + " їсть " + f.type + "? - " + a.eat(f));  
 }  
 System.out.println("");  
 }  
  
 }  
}

package task1;  
  
public class Animal {  
 public String name;  
 public double speed;  
  
 public Animal(String name, double speed) {  
 this.name = name;  
 this.speed = speed;  
 }  
  
 public void run() {  
 System.*out*.println(name + " біжить зі швидкістю " + speed);  
 }  
  
 public boolean eat(Food food) {  
 return true;  
 }  
}

package task1;  
  
public class Carnivorous extends Animal {  
 public Carnivorous(String name, double speed) {  
 super(name, speed);  
 }  
  
 @Override  
 public boolean eat(Food food) {  
 if(food instanceof Meat) {  
 return true;  
 }  
 else {  
 return false;  
 }  
 }  
}

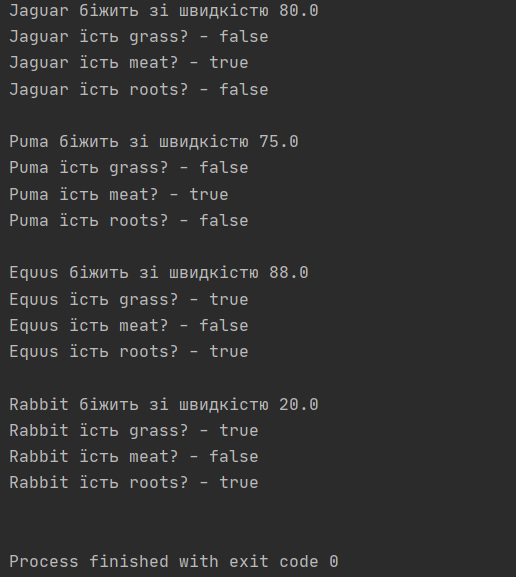
package task1;  
  
public class Herbivorous extends Animal {  
 public Herbivorous(String name, double speed) {  
 super(name, speed);  
 }  
  
 @Override  
 public boolean eat(Food food) {  
 if(food instanceof Herb) {  
 return true;  
 }  
 else {  
 return false;  
 }  
 }  
}

package task1;  
  
public class Food {  
 public String type;  
  
 public Food(String type)  
 {  
 this.type = type;  
 }  
}

package task1;  
  
public class Herb extends Food {  
 public Herb(String type) {  
 super(type);  
 }  
}

package task1;  
  
public class Meat extends Food {  
 public Meat(String type)  
 {  
 super(type);  
 }  
}

**Результат роботи програми:**



**Завдання № 2**

Написати ієрархію класів та продемонструвати порядок ініціалізації полів класу. Використати конструктор та статичні блоки ініціалізації в усіх класах (мінімум 3 класи)

**Код програми:**

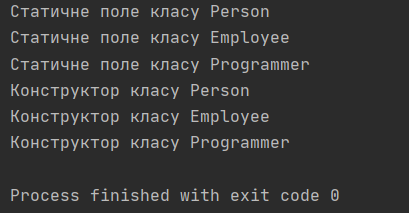
package task2;  
  
public class Task2 {  
 public static void main(String[] args) {  
 Programmer p = new Programmer("Oleg", true, 800);  
 }  
}

package task2;  
  
public class Person {  
 public String name;  
 public static boolean *isHuman*;  
  
 public Person(String name) {  
 this.name = name;  
 System.*out*.println("Конструктор класу Person");  
 }  
  
 static {  
 *isHuman* = true;  
 System.*out*.println("Статичне поле класу Person");  
 }  
}

package task2;  
  
public class Employee extends Person {  
 public boolean hasSalary;  
 public static boolean *isEmployee*;  
  
 public Employee(String name, boolean hasSalary) {  
 super(name);  
 this.hasSalary = hasSalary;  
 System.*out*.println("Конструктор класу Employee");  
 }  
  
 static {  
 *isEmployee* = true;  
 System.*out*.println("Статичне поле класу Employee");  
 }  
}

package task2;  
  
public class Programmer extends Employee {  
 public double salary;  
 public static boolean *isProgrammer*;  
  
 public Programmer(String name, boolean hasSalary, double salary) {  
 super(name, hasSalary);  
 this.salary = salary;  
 System.*out*.println("Конструктор класу Programmer");  
 }  
  
 static {  
 *isProgrammer* = true;  
 System.*out*.println("Статичне поле класу Programmer");  
 }  
}

**Результат роботи програми:**



**Завдання № 3**

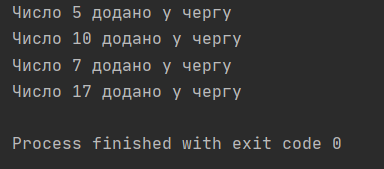
Реалізувати структуру даних “черга” та методи add(...) i delete(...).

**Код програми:**

package task3;  
  
public class Task3 {  
 public static void main(String[] args) {  
 Queue = new Queue(100);  
 queue.add(5);  
 queue.add(10);  
 queue.add(7);  
 queue.add(17);  
 queue.delete();  
 }  
}

package task3;  
  
public class Queue {  
 private int first;  
 private int last;  
 private int size;  
 private int capacity;  
 private int[] array;  
  
 public Queue(int c) {  
 capacity = c;  
 first = size = 0;  
 last = capacity - 1;  
 array = new int[capacity];  
 }  
  
 public void add(int n) {  
 if (size == capacity)  
 return;  
 last = (last + 1) % capacity;  
 array[last] = n;  
 size = size + 1;  
 System.*out*.println("Число " + n + " додано у чергу");  
 }  
  
 public int delete() {  
 if (size == 0)  
 return Integer.*MIN\_VALUE*;  
 int item = array[first];  
 first = (first + 1) % capacity;  
 size = size - 1;  
 return item;  
 }  
}

**Результат роботи програми:**



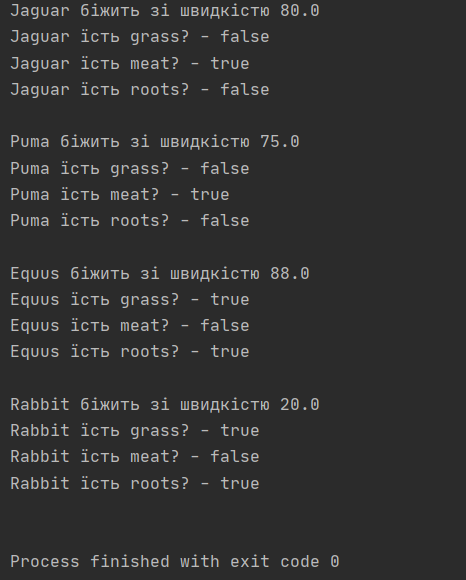
**Завдання № 4**

Продемонструвати поліморфізм із першого завдання використовуючи List.

**Код програми:**

package task4;  
  
import task1.\*;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Task4 {  
 public static void main(String[] args) {  
 List<Animal> animals = new ArrayList<Animal>();  
 animals.add(new Carnivorous("Jaguar", 80));  
 animals.add(new Carnivorous("Puma", 75));  
 animals.add(new Herbivorous("Equus", 88));  
 animals.add(new Herbivorous("Rabbit", 20));  
  
 List<Food> food = new ArrayList<Food>();  
 food.add(new Herb("grass"));  
 food.add(new Meat("meat"));  
 food.add(new Herb("roots"));  
  
 for(Animal a : animals) {  
 a.run();  
 for(Food f : food) {  
 System.*out*.println(a.name + " їсть " + f.type + "? - " + a.eat(f));  
 }  
 System.*out*.println("");  
 }  
 }  
}

**Результат роботи програми:**



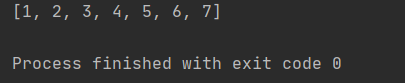
**Завдання № 5**

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

**Код програми:**

package task5;  
import java.util.\*;  
  
public class Task5 {  
 public static void main(String[] args) {  
 int a = 1, b = 7;  
 int[] array = new int[100];  
  
 for(int i = 0; i < array.length; i++) {  
 array[i] = a + (int) (Math.*random*() \* b);  
 }  
  
 Set<Integer> set = new HashSet<Integer>();  
 for (int n: array) {  
 set.add(n);  
 }  
  
 System.*out*.println(set);  
 }  
}

**Результат роботи програми:**



**Завдання № 6.**

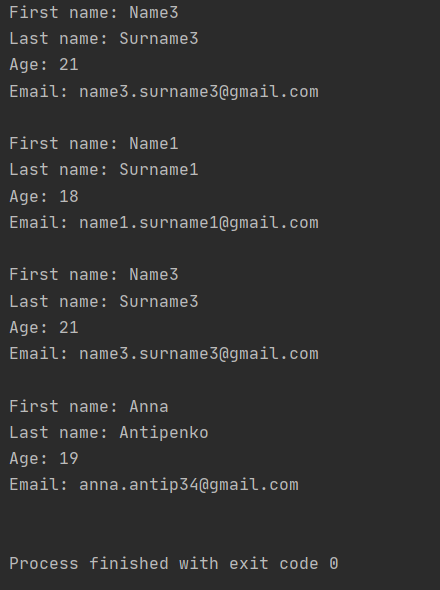
Створити Map використовуючи як ключ ім’я користувача (User із лабораторної роботи № 1), а значення - об’єкт User. Продемонструвати додавання, видалення та додавання різних об’єктів з однаковим ключем.

**Код програми:**

package task6;  
  
import java.util.HashMap;  
import java.util.Map;  
  
public class Task6 {  
 public static void main(String[] args) {  
 Map<String, User> map = new HashMap<String, User>();  
 User u1 = new User("Oleg", "Komenda", 18, "olegkomenda12@gmail.com");  
 User u2 = new User("Name1", "Surname1", 18, "name1.surname1@gmail.com");  
 User u3 = new User("Name2", "Surname2", 25, "name2.surname2@gmail.com");  
 User u4 = new User("Name3", "Surname3", 21, "name3.surname3@gmail.com");  
  
 map.put(u1.firstName, u1);  
 map.put(u2.firstName, u2);  
 map.put(u3.firstName, u3);  
 map.put(u4.firstName, u4);  
 map.put(u3.firstName, u1);  
 map.put(u1.firstName, u4);  
  
 User findFirstUser = map.get("Oleg");  
 User findSecondUser = map.get("Name1");  
 User findThirdUser = map.get("Oleg");  
 User findForthUser = map.get("Name2");  
  
 findFirstUser.PrintInfo();  
 findSecondUser.PrintInfo();  
 findThirdUser.PrintInfo();  
 findForthUser.PrintInfo();  
  
 map.remove("Oleg");  
 map.remove("Name2");  
 map.remove("Name3");  
  
 }  
}

package task6;  
  
public class User {  
 public String firstName;  
 public String lastName;  
 public int age;  
 public String email;  
  
 public User(String firstName, String lastName, int age, String email) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.age = age;  
 this.email = email;  
 }  
  
 public void PrintInfo() {  
 System.*out*.println("First name: " + firstName);  
 System.*out*.println("Last name: " + lastName);  
 System.*out*.println("Age: " + age);  
 System.*out*.println("Email: " + email + "\n");  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if(obj == this) {  
 return true;  
 }  
 if(obj == null) {  
 return false;  
 }  
 if(obj.getClass() != getClass()) {  
 return false;  
 }  
 User u = (User) obj;  
 if(age != u.age) {  
 return false;  
 }  
 else if(email != u.email) {  
 return false;  
 }  
 else if(lastName != u.lastName) {  
 return false;  
 }  
 else if(firstName != u.firstName) {  
 return false;  
 }  
 return true;  
 }  
  
 @Override  
 public int hashCode() {  
 final int prime = 31;  
 int result = 1;  
 result = prime \* result + (firstName == null ? 0 : firstName.hashCode());  
 result = prime \* result + (lastName == null ? 0 : lastName.hashCode());  
 result = prime \* result + age;  
 result = prime \* result + (email == null ? 0 : email.hashCode());  
 return result;  
 }  
}

**Результат роботи програми:**



**Висновок**: під час виконання лабораторної роботи я ознайомився з основами ООП мови програмування Java та з Java Collections.