**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

по дисциплине

«Введение в разработка программного обеспечения»

**Развитие представлений о разработке программ. Объектно-ориентированное программирование в Java**

Выполнил студент группы ИТИ-21

Маслов Д.И.

Проверил: ассистент

Ястребов А.А.

Гомель 2024

Цель работы: исследовать развитие представлений о разработке

программного обеспечения, сосредоточив внимание на объектно-ориентированном программировании (ООП) в *Java*.

**Задание.**

Подготовить отчет, который должен содержать:

* Название работы.
* Цель работы.
* Условие задачи.
* Разработать *UML*‒диаграмму.
* Листинг программы.
* Вывод.

**Ход работы**

**Задание:**

Составить и блок‒схему алгоритма и программу для вычисления значений переменных в соответствии с условием. Вариант задания приведён в таблице 1. Результат выполнения задания представлен на рисунке 1, *UML*‒диаграмма на рисунке 2. Код программы представлен в приложении А.

Таблица 1 – Условие к заданию (Вариант 7)

|  |  |
| --- | --- |
| Вариант | Условие задачи |
| 14 | 1.1 Создать иерархию классов для учета рыб в Аквариуме.  1.2 В классе рыба реализовать поля: вид (хищная, мирная и т.д.), вес,  размер (отдельный класс с полями ширина и длина).  1.3 Создать не менее 5 видов рыб, каждой по 2 экземпляра.  1.4 Вывести информацию о видах рыб, которые являются хищниками  и весят больше 100 гр.  1.5 Вычислить самую длинную рыбу. |



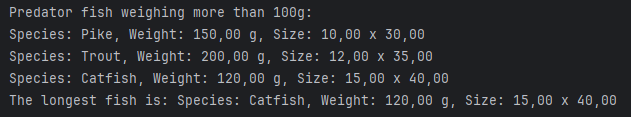


Рисунок 1 ‒ Результат выполнения задания



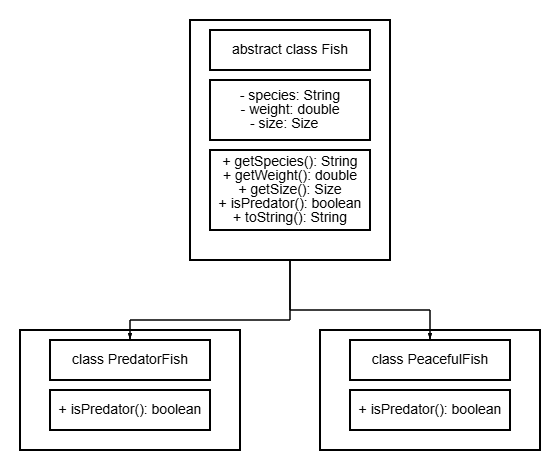


Рисунок 2 ‒ *UML*‒диаграмма

Вывод: в результате выполнения лабораторной работы изучили развитие представлений о разработке программного обеспечения, сосредоточив внимание на объектно-ориентированном программировании (ООП) в *Java*.

**ПРИЛОЖЕНИЕ А**

**Текст программы**

**main.java:**

import java.util.ArrayList;  
import java.util.List;  
  
/\*\*  
 \* Класс, представляющий размер рыбы.  
 \*/  
class Size {  
 private double width;  
 private double length;  
  
 public Size(double width, double length) {  
 this.width = width;  
 this.length = length;  
 }  
  
 public double getWidth() {  
 return width;  
 }  
  
 public double getLength() {  
 return length;  
 }  
}  
  
/\*\*  
 \* Абстрактный класс для всех рыб.  
 \*/  
abstract class Fish {  
 private String species;  
 private double weight;  
 private Size size;  
  
 public Fish(String species, double weight, Size size) {  
 this.species = species;  
 this.weight = weight;  
 this.size = size;  
 }  
  
 public String getSpecies() {  
 return species;  
 }  
  
 public double getWeight() {  
 return weight;  
 }  
  
 public Size getSize() {  
 return size;  
 }  
  
 public abstract boolean isPredator();  
  
 @Override  
 public String toString() {  
 return String.format("Species: %s, Weight: %.2f g, Size: %.2f x %.2f",  
 species, weight, size.getWidth(), size.getLength());  
 }  
}  
  
/\*\*  
 \* Класс, представляющий хищную рыбу.  
 \*/  
class PredatorFish extends Fish {  
 public PredatorFish(String species, double weight, Size size) {  
 super(species, weight, size);  
 }  
  
 @Override  
 public boolean isPredator() {  
 return true;  
 }  
}  
  
/\*\*  
 \* Класс, представляющий мирную рыбу.  
 \*/  
class PeacefulFish extends Fish {  
 public PeacefulFish(String species, double weight, Size size) {  
 super(species, weight, size);  
 }  
  
 @Override  
 public boolean isPredator() {  
 return false;  
 }  
}  
  
public class AquariumDemo {  
 public static void main(String[] args) {  
 List<Fish> fishList = new ArrayList<>();  
  
 // Создание экземпляров рыб  
 fishList.add(new PredatorFish("Pike", 150, new Size(10, 30)));  
 fishList.add(new PredatorFish("Trout", 200, new Size(12, 35)));  
 fishList.add(new PeacefulFish("Goldfish", 50, new Size(5, 10)));  
 fishList.add(new PeacefulFish("Guppy", 5, new Size(2, 4)));  
 fishList.add(new PredatorFish("Catfish", 120, new Size(15, 40)));  
 fishList.add(new PeacefulFish("Neon Tetra", 1, new Size(1, 2)));  
  
 // Вывод информации о хищниках больше 100 г  
 System.out.println("Predator fish weighing more than 100g:");  
 for (Fish fish : fishList) {  
 if (fish.isPredator() && fish.getWeight() > 100) {  
 System.out.println(fish);  
 }  
 }  
  
 // Вычисление самой длинной рыбы  
 Fish longestFish = null;  
 for (Fish fish : fishList) {  
 if (longestFish == null || fish.getSize().getLength() > longestFish.getSize().getLength()) {  
 longestFish = fish;  
 }  
 }  
  
 if (longestFish != null) {  
 System.out.println("The longest fish is: " + longestFish);  
 }  
 }  
}