**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине: «Технологии разработки программного обеспечения»

на тему: «Развитие представлений о разработке программ. Объектно-ориентированное программирование в *Java*»

Выполнил: студент гр. ИТИ-21

Ивашко В. Н.

Принял: преподаватель

Стефановский И. Л.

Гомель 2024

**Цель работы:** изучить основы проектирования и создания программ при помощи объектно-ориентированного языка программирования *Java*.

**Задание:**

1. Разработать UML-диаграмму иерархии классов, согласно варианта

|  |
| --- |
| 1.1 Создать иерархию для учета блюд в меню.  1.2 Меню содержит блюда: напитки и тосты (реализовать через наследование). Напитки и тосты имеют название, стоимость, калорийность и метку *vegan* *friendly*.  1.3 Создать объект меню и не менее 10 различных блюд.  1.4 Вывести все меню.  1.5 Вывести все *vegan* *friendly* блюда.  1.6 Подсчитать среднюю стоимость блюд с калорийностью более 300. |

2. При наименовании компонентов руководствоваться соглашением о наименовании.

3. При описании иерархии использовать наследование и композицию.

4. На основе *UML*-диаграммы разработать иерархию классов на языке Java.

5. Весь код должен быть снабжен элементами документирования.

6. Разработанную иерархию поместить в .*jar* файл для дальнейшего использования в качестве библиотечных классов.

7. Создать консольное приложение для демонстрации работы созданных классов.

8. Составить отчет о проделанной работе.

**Ход работы:**

Разработаем *UML*-диаграмму классов, включающую в себя суперкласс *Dish*, от которого будут наследоваться все остальные блюда. Также опишем классы, реализующие сами наследуемые виды блюд. Полученная диаграмма представлена на рисунке 1.

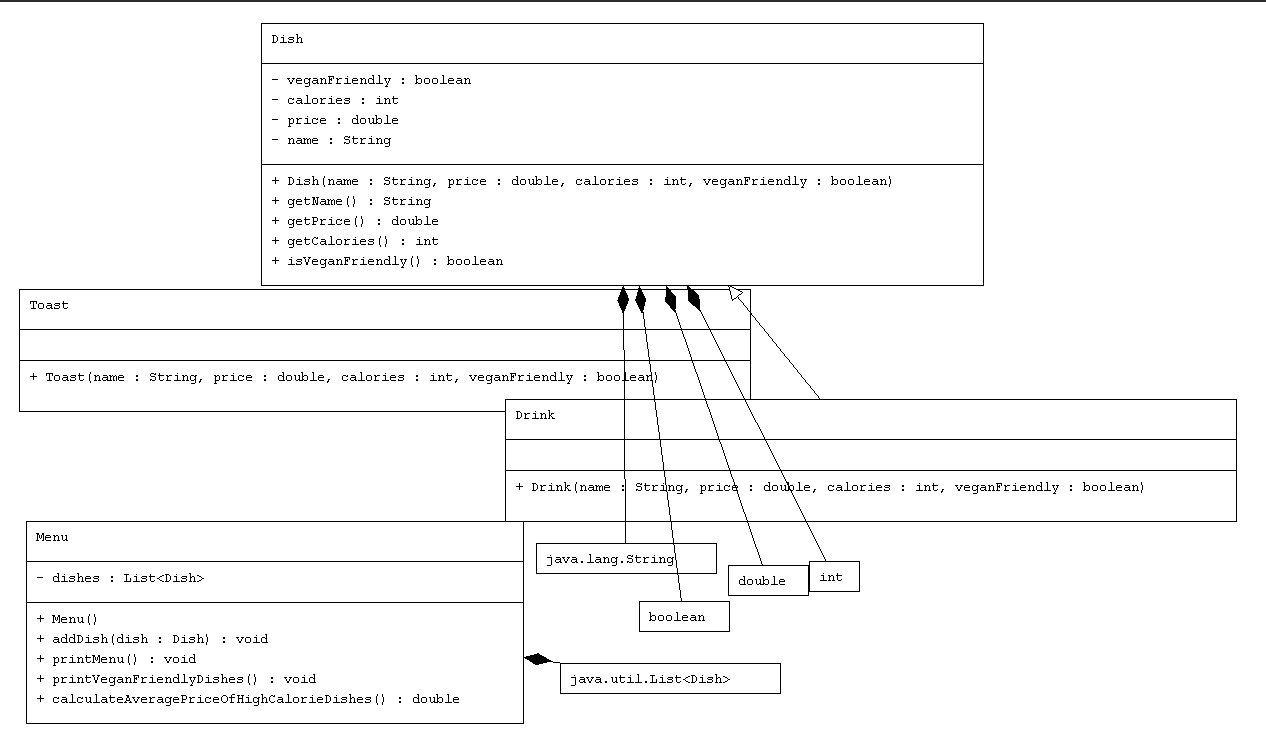


Рисунок 1 – Диаграмма классов

Реализуем эти классы на языке *Java*. В отдельном проекте создадим класс Main, в котором будет проводиться работа с элементами данных классов. Для того, чтобы работать с классами в другом проекте, необходимо создать .*jar* файл из библиотеки классов и подключить его ко второму проекту. Последовательность этих действий представлена на рисунках 2 – 6.

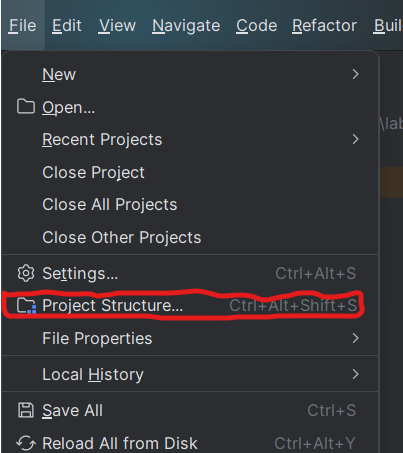


Рисунок 2 – Структура проекта

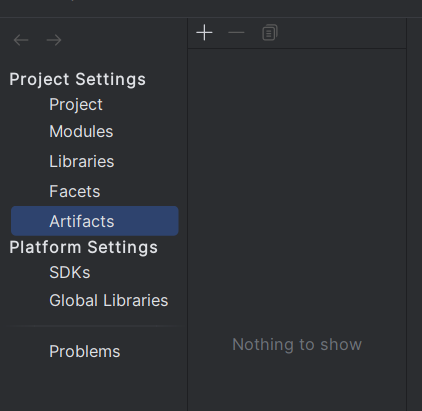


Рисунок 3 – Артефакты

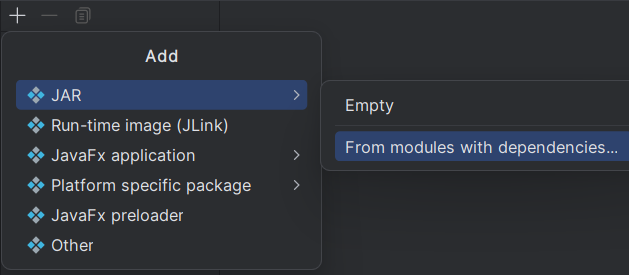


Рисунок 4 – Добавление *jar* файла

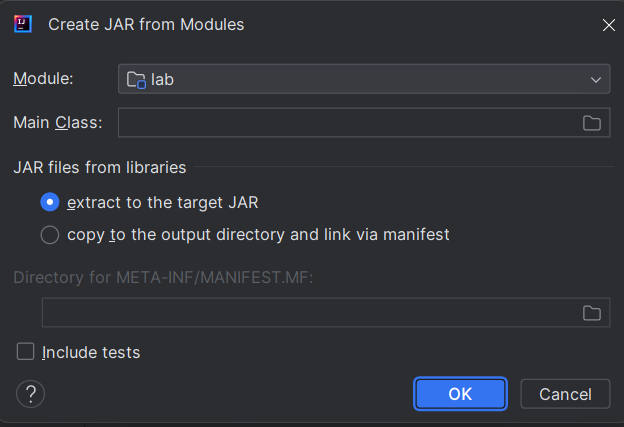


Рисунок 5 – Создание файла

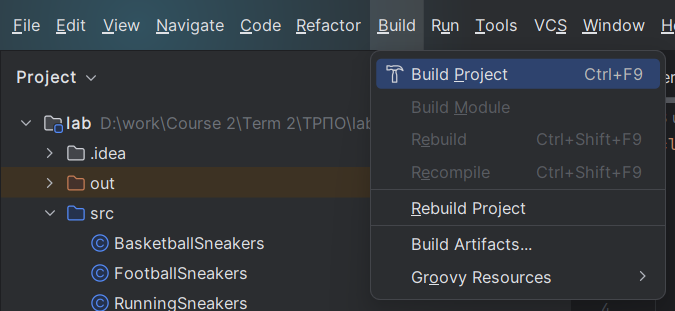


Рисунок 6 – Создание библиотеки классов

После создания файла библиотеки необходимо подключить его в проект с классом *Main*. Для этого в *Main*-проекте нужно зайти в раздел Структура проекта и выбрать раздел Библиотеки. После этого выбираем файл с библиотекой. Его путь представлен на рисунке 7.



Рисунок 7 – Путь к библиотеке классов

После добавления становится возможным работать со всеми доступными классами из данной библиотеки. Результат работы программы представлен на рисунках 8 – 13.

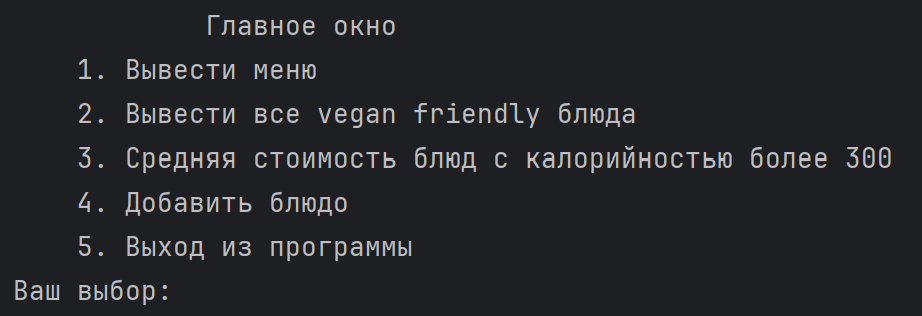


Рисунок 8 – Главное меню

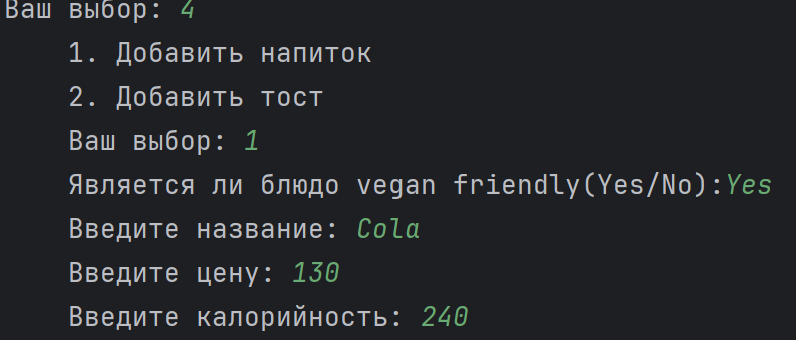


Рисунок 9 – Добавление напитка

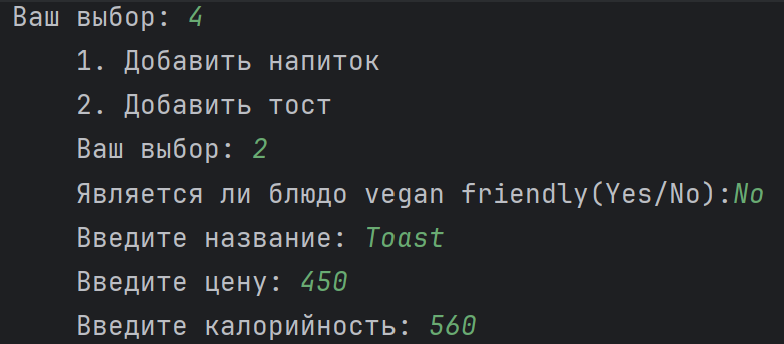


Рисунок 10 – Добавление тоста

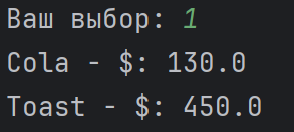


Рисунок 11 – Вывод меню

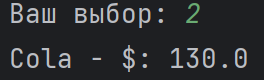


Рисунок 12 – Вывод всех *vegan* *friendly* блюд

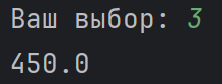


Рисунок 13 – Вывод средней стоимости блюд с калорийностью более 300

В приложении А представлен код всех файлов.

**Выводы:** в результате выполнения данной лабораторной работы были изучены основы проектирования и создания программ при помощи объектно-ориентированного языка программирования *Java*.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Dish.java**

class Dish {  
 private String name;  
 private double price;  
 private int calories;  
 private boolean veganFriendly;  
  
 public Dish(String name, double price, int calories, boolean veganFriendly) {  
 this.name = name;  
 this.price = price;  
 this.calories = calories;  
 this.veganFriendly = veganFriendly;  
 }  
  
 public String getName() {  
 return this.name;  
 }  
  
 public double getPrice() {  
 return this.price;  
 }  
  
 public int getCalories() {  
 return this.calories;  
 }  
  
 public boolean isVeganFriendly() {  
 return this.veganFriendly;  
 }  
}

**Drink.java**

class Drink extends Dish {  
 public Drink(String name, double price, int calories, boolean veganFriendly) {  
 super(name, price, calories, veganFriendly);  
 }  
}

**Toast.java**

class Toast extends Dish {  
 public Toast(String name, double price, int calories, boolean veganFriendly) {  
 super(name, price, calories, veganFriendly);  
 }  
}

**Menu.java**

import java.io.PrintStream;  
import java.util.ArrayList;  
import java.util.Iterator;  
import java.util.List;  
  
class Menu {  
 private List<Dish> dishes = new ArrayList();  
  
 public Menu() {  
 }  
  
 public void addDish(Dish dish) {  
 this.dishes.add(dish);  
 }  
  
 public void printMenu() {  
 if (!this.dishes.isEmpty()) {  
 Iterator var1 = this.dishes.iterator();  
  
 while(var1.hasNext()) {  
 Dish dish = (Dish)var1.next();  
 PrintStream var10000 = System.out;  
 String var10001 = dish.getName();  
 var10000.println(var10001 + " - $: " + dish.getPrice());  
 }  
  
 }  
 }  
  
 public void printVeganFriendlyDishes() {  
 if (!this.dishes.isEmpty()) {  
 Iterator var1 = this.dishes.iterator();  
  
 while(var1.hasNext()) {  
 Dish dish = (Dish)var1.next();  
 if (dish.isVeganFriendly()) {  
 PrintStream var10000 = System.out;  
 String var10001 = dish.getName();  
 var10000.println(var10001 + " - $: " + dish.getPrice());  
 }  
 }  
  
 }  
 }  
  
 public double calculateAveragePriceOfHighCalorieDishes() {  
 if (this.dishes.isEmpty()) {  
 return 0.0;  
 } else {  
 int count = 0;  
 double totalPrice = 0.0;  
 Iterator var4 = this.dishes.iterator();  
  
 while(var4.hasNext()) {  
 Dish dish = (Dish)var4.next();  
 if (dish.getCalories() > 300) {  
 totalPrice += dish.getPrice();  
 ++count;  
 }  
 }  
  
 if (count == 0) {  
 return 0.0;  
 } else {  
 return totalPrice / (double)count;  
 }  
 }  
 }  
}

**Main.java**

import java.util.\*;  
public class Main  
{  
 private static void printUI()  
 {  
 System.out.println("\t\t\tГлавное окно");  
 System.out.println("\t1. Вывести меню");  
 System.out.println("\t2. Вывести все vegan friendly блюда");  
 System.out.println("\t3. Средняя стоимость блюд с калорийностью более 300");  
 System.out.println("\t4. Добавить блюдо");  
 System.out.println("\t5. Выход из программы");  
 }  
  
 public static void main(String[] args)  
 {  
 Scanner scanner = new Scanner(System.in);  
 Menu menu = new Menu();  
 int choice;  
 while (true)  
 {  
 printUI();  
 System.out.print("Ваш выбор: ");  
 choice = Integer.parseInt(scanner.nextLine());  
 switch (choice)  
 {  
 case 1:  
 menu.printMenu();  
 break;  
 case 2:  
 menu.printVeganFriendlyDishes();  
 break;  
 case 3:  
 System.out.println(menu.calculateAveragePriceOfHighCalorieDishes());  
 break;  
 case 4:  
 String input;  
 String name;  
 double price;  
 int calories;  
 boolean veganFriendly;  
 System.out.print("\033[H\033[2J");  
 System.out.flush();  
 System.out.println("\t1. Добавить напиток");  
 System.out.print("\t2. Добавить тост");  
 System.out.print("\n\tВаш выбор: ");  
 choice = Integer.parseInt(scanner.nextLine());  
 System.out.print("\tЯвляется ли блюдо vegan friendly(Yes/No):");  
 input = scanner.nextLine();  
 System.out.print("\tВведите название: ");  
 name = scanner.nextLine();  
 System.out.print("\tВведите цену: ");  
 price = Double.parseDouble(scanner.nextLine());  
 System.out.print("\tВведите калорийность: ");  
 calories = Integer.parseInt(scanner.nextLine());  
 if(input.equalsIgnoreCase("Yes"))  
 {  
 veganFriendly = true;  
 }  
 else  
 {  
 veganFriendly = false;  
 }  
 switch (choice)  
 {  
 case 1:  
 menu.addDish(new Drink(name, price, calories, veganFriendly));  
 break;  
 case 2:  
 menu.addDish(new Toast(name, price, calories, veganFriendly));  
 break;  
 default:  
 System.out.print("\tНеверный ввод");  
 break;  
   
 }  
 break;  
 case 5:  
 scanner.close();  
 return;  
 default:  
 System.out.print("\tНеверный ввод");  
 break;  
 }  
 }  
 }  
}