**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П.О.СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

направление специальности 6-05-0611-01 Информационные системы

и технологии (в игровой индустрии)

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине: «Введение в разработку программного обеспечения»

на тему: **«Объектно-ориентированное программирование в *Java***. **Системы сборки проекта»**

Выполнил: студент гр. ИТИ-21

Калинин В.М.

Принял: преподаватель

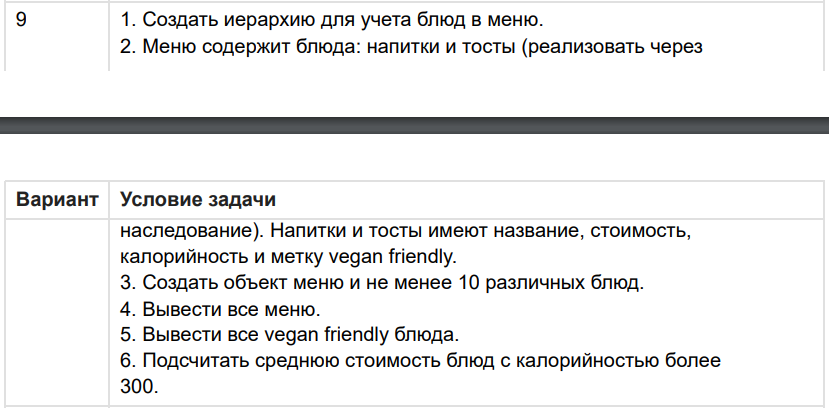
Ястребов А.А.

Гомель 2024

**Цель работы**: изучить основы объектно-ориентированного программирования (ООП) на языке *Java*, а также освоить принципы разработки программного обеспечения с применением ключевых концепций ООП, таких как инкапсуляция, наследование и полиморфизм.

# **Задание:**

1. Разработать *UML*-диаграмму иерархии классов, согласно варианту.
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.



**Ход работы и результаты выполнения:**

Разработанная *UML*-диаграмма иерархии классов представлена на рисунке 1.

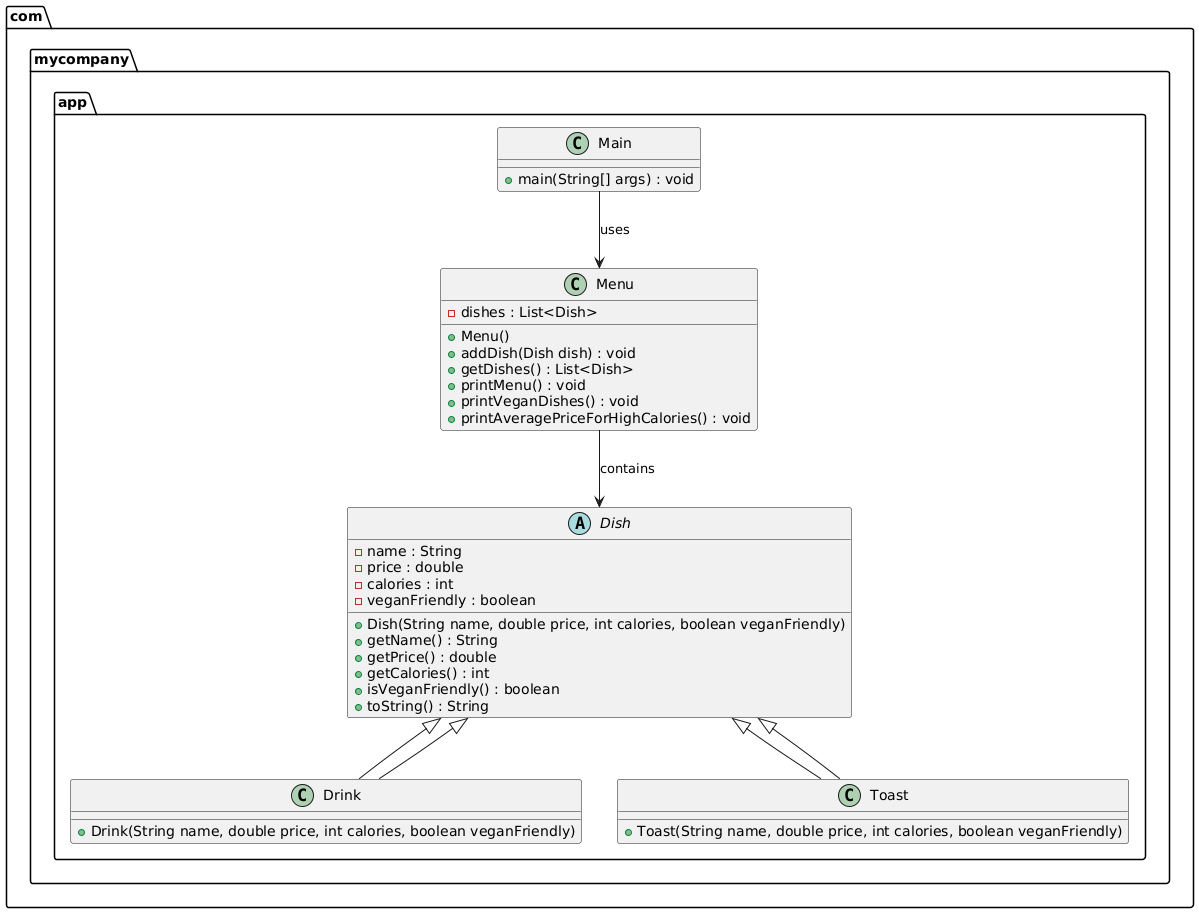


Рисунок 1 – *UML*-диаграмма иерархии классов

В ходе разработки программы были разработаны следующие классы:

* Класс «*Menu*», отвечает за представление меню, хранит список блюд;
* Класс «*Dish*» – компонент меню;
* Класс «*Drink*» – компонент меню, также является подклассом класса *Dish*;
* Класс «*Toast*» – компонент меню, также является подклассом класса *Dish*;
* Класс «*Main*», содержит метод *main*, который создает экземпляр меню, добавляет напитки и тосты, и вызывает методы для вывода информации о меню.

При разработке указанных классов применялись главные принципы ООП, такие как инкапсуляция, наследование и полиморфизм.

После создания всех классов и отладки программы был создан .*jar* файл со всеми разработанными классами для интеграции данных классов в другие проекты.

Первое что пользователь увидит это консольное приложение с выбором на рисунке 2.

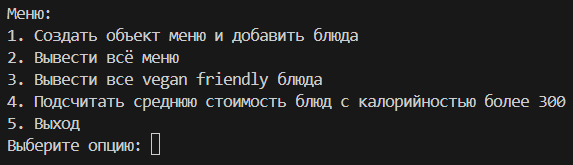


Рисунок 2 – Консольное приложение

Так же было создано консольное приложение, чтобы увидеть какие есть блюда в меню, и какие блюда подходят для веганов. Результат разработки показан на рисунке 3.

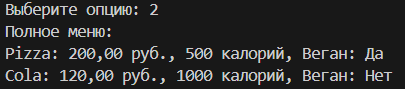


Рисунок 3 – Окно консольного приложения

После показа заказа, пользователь видит всю информацию о меню для веганов. Результат данной операции представлен на рисунке 4.

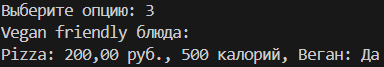


Рисунок 4 – Вывод информации блюд для веганов

Последняя строка в консольном приложении показывает среднюю цену блюд с калорийностью более 300. Результат показан на рисунке 4.



Рисунок 5 – Вывод средней цены блюд с калорийностью более 300

Листинг класса *Main* представлен в приложении А.

**Вывод:** в ходе выполнения лабораторной работы были изучены основы объектно-ориентированного программирования (ООП) на языке *Java*, освоены принципы разработки программного обеспечения с применением ключевых концепций ООП, таких как инкапсуляция, наследование и полиморфизм.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Листинг разработанных классов**

**«Main.java»**

package com.mycompany.app;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

// System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF\_8));

Scanner scanner = new Scanner(System.in);

Menu menu = new Menu();

while (true) {

System.out.println("\nМеню:");

System.out.println("1. Создать объект меню и добавить блюда");

System.out.println("2. Вывести всё меню");

System.out.println("3. Вывести все vegan friendly блюда");

System.out.println("4. Подсчитать среднюю стоимость блюд с калорийностью более 300");

System.out.println("5. Выход");

System.out.print("Выберите опцию: ");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1:

System.out.println("Добавление блюд в меню:");

for (int i = 0; i < 10; i++) {

System.out.print("Введите название блюда: ");

String name = scanner.nextLine();

System.out.print("Введите стоимость: ");

double price = scanner.nextDouble();

System.out.print("Введите калорийность: ");

int calories = scanner.nextInt();

System.out.print("Это блюдо веганское? (yes/no): ");

boolean veganFriendly = scanner.next().equalsIgnoreCase("yes");

scanner.nextLine(); // Consume newline

System.out.print("Это напиток или тост? (drink/toast): ");

String type = scanner.nextLine();

if (type.equalsIgnoreCase("drink")) {

menu.addDish(new Drink(name, price, calories, veganFriendly));

} else if (type.equalsIgnoreCase("toast")) {

menu.addDish(new Toast(name, price, calories, veganFriendly));

} else {

System.out.println("Некорректный ввод типа блюда!");

}

}

break;

case 2:

menu.printMenu();

break;

case 3:

menu.printVeganDishes();

break;

case 4:

menu.printAveragePriceForHighCalories();

break;

case 5:

System.out.println("Выход из программы.");

scanner.close();

return;

default:

System.out.println("Некорректный выбор! Попробуйте снова.");

}

}

}

}

**«*Dish*.*java*»**

package com.mycompany.app;

public abstract 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 name;

}

public double getPrice()

{

return price;

}

public int getCalories()

{

return calories;

}

public boolean isVeganFriendly()

{

return veganFriendly;

}

@Override

public String toString()

{

return String.format("%s: %.2f руб., %d калорий, Веган: %s",

name, price, calories, veganFriendly ? "Да" : "Нет");

}

}

**«*Drink*.*java*»**

package com.mycompany.app;

public class Drink extends Dish

{

public Drink(String name, double price, int calories, boolean veganFriendly)

{

super(name, price, calories, veganFriendly);

}

}

**«*Toast*.*java*»**

package com.mycompany.app;

public class Toast extends Dish

{

public Toast(String name, double price, int calories, boolean veganFriendly)

{

super(name, price, calories, veganFriendly);

}

}

**«*Menu*.*java*»**

package com.mycompany.app;

import java.util.ArrayList;

import java.util.List;

public class Menu

{

private List<Dish> dishes;

public Menu()

{

this.dishes = new ArrayList<>();

}

public void addDish(Dish dish)

{

dishes.add(dish);

}

public List<Dish> getDishes()

{

return dishes;

}

public void printMenu()

{

System.out.println("Полное меню:");

dishes.forEach(System.out::println);

}

public void printVeganDishes()

{

System.out.println("Vegan friendly блюда:");

dishes.stream()

.filter(Dish::isVeganFriendly)

.forEach(System.out::println);

}

public void printAveragePriceForHighCalories()

{

double average = dishes.stream()

.filter(dish -> dish.getCalories() > 300)

.mapToDouble(Dish::getPrice)

.average()

.orElse(0.0);

System.out.printf("Средняя стоимость блюд с калорийностью > 300: %.2f руб.%n", average);

}

}