ГУАП

КАФЕДРА № 42

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПРЕПОДАВАТЕЛЬ

| К.т.н. Доцент |  |  |  | В.А.Ушаков |
| --- | --- | --- | --- | --- |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5 |
| --- |
| **ИЗУЧЕНИЕ ПРОДВИНУТЫХ МЕТОДОВ РАЗРАБОТКИ И ПРИМЕНЕНИЕ ИХ НА ПРАКТИКЕ**  Вариант 5 |
|  |
| по курсу: КРОССПЛАТФОРМЕННОЕ ПРОГРАММИРОВАНИЕ |
|  |

РАБОТУ ВЫПОЛНИЛ

| СТУДЕНТ ГР. № | 4128 |  |  |  | В. А. Воробьев |
| --- | --- | --- | --- | --- | --- |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023

**Цель работы:** Изучить и использовать на практике паттерны проектирования.

З**адание:**

Разработать программу для работы с состояниями объекта. Использовать паттерн состояние.

Пример для воды. У воды три состояние (лёд, жидкость, пар) и она ведёт себя по-разному при нагреве или заморозке в зависимости от состояния. Если воду нагревать, из состояния жидкость она перейдет в пар, а из пара ей уже некуда переходить, так что она останется паром. Если охлаждать пар он перейдет в состояние жидкости. Если еще раз охладить жидкость она перейдет в состояние льда. Если охладить еще раз, вода останется льдом.

Пример логики:

* Пользователь выбирает, нагреть или охладить воду
* Пользователю выводится результат изменения состояния
* Это повторяется пока пользователю не надоест

**Результаты работы программы:**

Программа реализует простую систему управления состояниями студента в учебном процессе. Для этого был использован паттерн состояние, предоставляя каждому состоянию (обучение, сдача экзаменов, комиссия) свои характеристики и поведение. UML диаграмма паттерна представлена на рисунке 1.

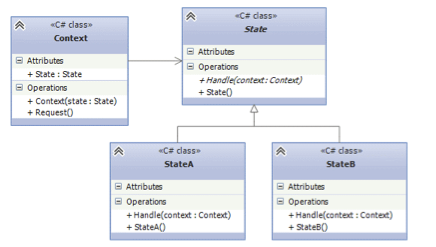


Рисунок 1 - UML-диаграмма паттерна состояние

Основной частью программы является набор классов, представляющих различные состояния и контекст – класс "Студент". Каждое состояние определяет свою реализацию методов для вывода текущего состояния и перехода к следующему. Класс "Студент" хранит текущее состояние и позволяет выполнять действия, изменяющие его состояние.

Пример использования программы представлен в классе "Main". Ввод с консоли позволяет пользователю выбирать действия, такие как переход к следующему состоянию или завершение программы. В процессе выполнения программа выводит текущее состояние студента в зависимости от выбранного действия. Исходный код класса Main представлен в Приложении. Консольный интерфейс программы представлен на рисунках 2 и 3.

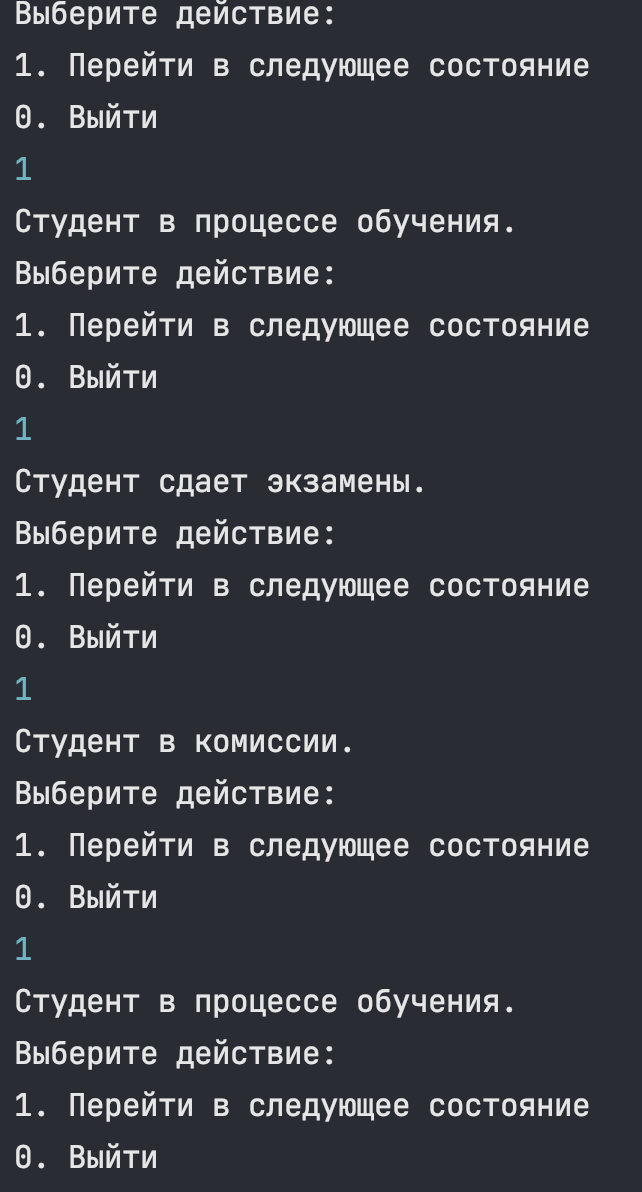


Рисунок 2 – Переключение состояний студента

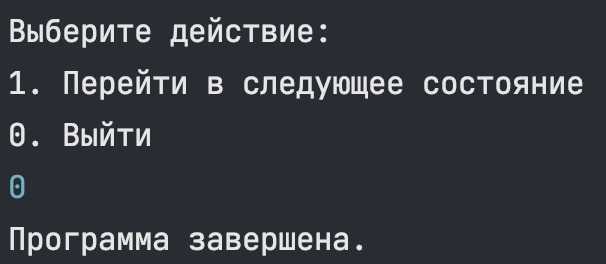


Рисунок 3 – Выход из программы

**Вывод:**

В ходе выполнения лабораторной работы мы успешно освоили процедурную модель программирования на языке Java, а также углубили свои знания в работе c паттернами. Основной задачей было разработать программу, использующую паттерн состояние для работы с состоянием студента.

Мы эффективно реализовали логику, в которой пользователь может менять состояние объекта студент. Применение паттерна состояние позволило нам организовать логику изменения состояний в зависимости от действий пользователя, обеспечивая четкую структуру и управление состояниями объекта. Полученные знания и навыки в рамках этой лабораторной работы будут полезны при разработке более сложных программ с использованием объектно-ориентированного программирования и применением паттернов проектирования.

**ПРИЛОЖЕНИЕ**

Main.java

package lab5;

import java.util.Scanner;

interface State {

void askState();

void nextState(Student student);

}

class StudyingState implements State {

@Override

public void askState() {

System.out.println("Студент в процессе обучения.");

}

@Override

public void nextState(Student student) {

student.setState(new TakingExamState());

}

}

class TakingExamState implements State {

@Override

public void askState() {

System.out.println("Студент сдает экзамены.");

}

@Override

public void nextState(Student student) {

student.setState(new CommissionState());

}

}

class CommissionState implements State {

@Override

public void askState() {

System.out.println("Студент в комиссии.");

}

@Override

public void nextState(Student student) {

student.setState(new StudyingState());

}

}

class Student {

private State currentState;

public Student(State initialState) {

this.currentState = initialState;

}

public void setState(State state) {

this.currentState = state;

}

public void askState() {

currentState.askState();

}

public void study() {

currentState.nextState(this);

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Student student = new Student(new StudyingState());

while (true) {

System.out.println("Выберите действие:");

System.out.println("1. Перейти в следующее состояние");

System.out.println("0. Выйти");

int choice = scanner.nextInt();

switch (choice) {

case 1 -> student.study();

case 0 -> {

System.out.println("Программа завершена.");

scanner.close();

System.exit(0);

}

default -> System.out.println("Неверный ввод. Попробуйте снова.");

}

student.askState();

}

}

}