Зразок оформлення титульного аркуша лабораторної роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**ЛАБОРАТОРНА РОБОТА No4**

з дисципліни «Об’єктно-орієнтоване програмування»

на тему

Реалізація поведінкових шаблонів проектування

Варіант No16

Виконав студент

ІІ курсу групи КП-ХХ

Палій Дмитро Володимирович

залікова книжка КП-8321

Перевірив

доцент, к.т.н. Заболотня Т.М.

Оцінка

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, підпис)

КИЇВ 2020

**Мета роботи**

Ознаойомлення з основними характеристиками

шаблонів «Стратегія», «Шаблонний метод» та «Стан», запам’ятовування

поширених ситуацій, коли використання цих шаблонів є доцільним,

набуття вмінь та навичок реалізації шаблонів під час створення

програмного коду.

**Постановка завдання**

Варіант 16

1) Дано список балів – рейтинг студентів наприкінці семестру. За

допомогою шаблона проектування «Стратегія» реалізувати механізм

його перетворення у список оцінок або список заліків-незаліків в

залежності від того, яка форма контролю передбачена у навчальному

плані для цього предмета.

2) Процес приготування різних блюд з однакового набору продуктів

складається з таких етапів: взяти певну (невелику) кількість продуктів,

помити їх, нарізати – це спільні етапи. Сам процес приготування: або

просто змішати нарізане, або зажарити, або зварити тощо – даний етап

для кожного рецепту свій . Останній етап – додати сіль, цукор, різни

приправи за смаком – також може бути реалізований по-різному. Блюда

для приготування з одного і того самого набору продуктів придумати

самостійно. Показати реалізацію як мінімум трьох блюд.

**Обгрунтування вибору шаблонів**

**Завдання 1:** обрано шаблон «Стратегія», оскільки потрібно забезпечувати різні варіанти алгоритму.

**Завдання 2:** обрано шаблон «Компонувальник», оскільки необхідно забезпечити одноразову реалізацію інваріантних частин

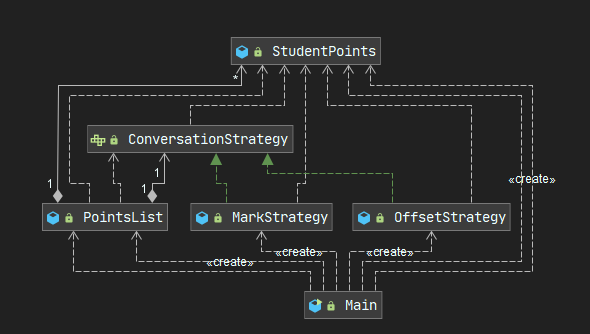
алгоритму, залишаючи реалізацію поведінки, що змінюється, на розсуд

підкласів.

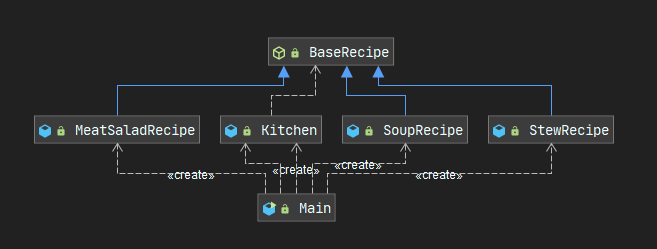
**Тексти коду програм**

Діаграма класів:

**Завдання 1:**

****

**Завдання 2:**

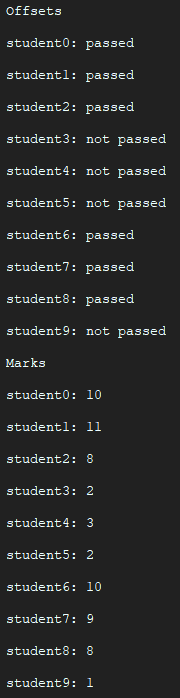


|  |
| --- |
| **Завдання 1** |
| *public class* Main {  *private static final double maxMark* = 100.0;  *public static void* main(String[] args) {  *var* pointsList = *new* ArrayList<StudentPoints>();  *for* (*int* i = 0; i < 10; i++) {  *var* name = "student" + i;  *var* mark = Math.*random*() \* *maxMark*;  *var* student = *new* StudentPoints(name, mark);  pointsList.add(student);  System.*out*.println(student.name + ' ' + *new* DecimalFormat("##.##").format(student.points));  }   *var* pointsToOffsetList = *new* PointsList(*new* OffsetStrategy(), pointsList);  *var* offsetList = pointsToOffsetList.convertMarksList();  System.*out*.println("Offsets");  *for* (String entry: offsetList) {  System.*out*.println(entry);  }   *var* pointsToMarksList = *new* PointsList(*new* MarkStrategy(), pointsList);  *var* markList = pointsToMarksList.convertMarksList();  System.*out*.println("Marks");  *for* (String entry: markList) {  System.*out*.println(entry);  }  } }  *public class* MarkStrategy *implements ConversationStrategy* {   @Override  *public* ArrayList<String> convert(ArrayList<StudentPoints> marksList) {  *var* gradedStudents = *new* ArrayList<String>();  *for* (StudentPoints student: marksList) {  *var* entry = student.name + ": " + (*int*)(student.points \* 12.0 / 100.0);  gradedStudents.add(entry);  }  *return* gradedStudents;  } }  *public class* OffsetStrategy *implements ConversationStrategy* {  *private final int* minMark = 60;   @Override  *public* ArrayList<String> convert(ArrayList<StudentPoints> marksList) {  *var* gradedStudents = *new* ArrayList<String>();  *for* (StudentPoints student: marksList) {  *var* entry = student.name + ": ";  *if* (student.points >= minMark) entry += "passed";  *else* entry += "not passed";  gradedStudents.add(entry);  }  *return* gradedStudents;  } }  *public class* PointsList {  *ConversationStrategy* strategy;  ArrayList<StudentPoints> marksList;  *public* PointsList(*ConversationStrategy* strategy, ArrayList<StudentPoints> marksList) {  *this*.strategy = strategy;  *this*.marksList = marksList;  }  *public* ArrayList<String> convertMarksList() {  *return* strategy.convert(marksList);  } }  *public class* StudentPoints {  *public double* points;  *public* String name;  *public* StudentPoints(String name, *double* mark) {  *this*.name = name;  *this*.points = mark;  } } |

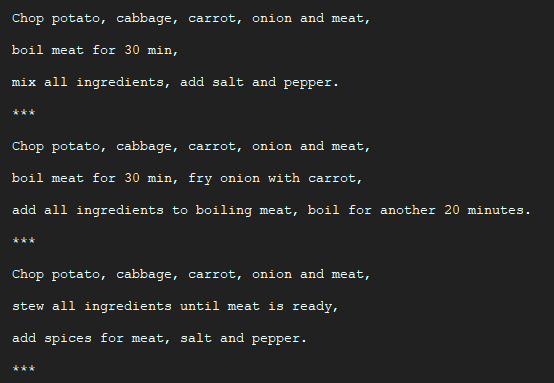
|  |
| --- |
| **Завдання 2** |
| *public class* Main {   *public static void* main(String[] args) {  *var* kitchen = *new* Kitchen();  *var* salad = kitchen.cookADish(*new* MeatSaladRecipe());  System.*out*.println(salad);  System.*out*.println("\*\*\*");  *var* soup = kitchen.cookADish(*new* SoupRecipe());  System.*out*.println(soup);  System.*out*.println("\*\*\*");  *var* stew = kitchen.cookADish(*new* StewRecipe());  System.*out*.println(stew);  System.*out*.println("\*\*\*");  } }  *public abstract class* BaseRecipe {  *public* String prepareIngredients() {  *return* "Chop potato, cabbage, carrot, onion and meat,\n";  }  *public abstract* String mainActions();  *public abstract* String finalAction(); }  *public class* Kitchen {  *public* String cookADish(BaseRecipe recipe) {  String dish = "";  dish += recipe.prepareIngredients() +  recipe.mainActions() +  recipe.finalAction();  *return* dish;  } }  *public class* MeatSaladRecipe *extends* BaseRecipe {   @Override  *public* String mainActions() {  *return* "boil meat for 30 min,\n";  }   @Override  *public* String finalAction() {  *return* "mix all ingredients, add salt and pepper.";  } }  *public class* SoupRecipe *extends* BaseRecipe {  @Override  *public* String mainActions() {  *return* "boil meat for 30 min, fry onion with carrot,\n";  }   @Override  *public* String finalAction() {  *return* "add all ingredients to boiling meat, boil for another 20 minutes.";  } }  *public class* StewRecipe *extends* BaseRecipe {  @Override  *public* String mainActions() {  *return* "stew all ingredients until meat is ready,\n";  }   @Override  *public* String finalAction() {  *return* "add spices for meat, salt and pepper.";  } } |

**Результати виконання**

**Завдання 1**

****

**Завдання 2**

****

**Висновки**

У цій лабораторній роботі я ознайомився з основними характеристиками шаблонів «Заступник», «Фасад», «Адаптер» та «Легковаговик», запам’ятав поширені ситуації, коли використання цих шаблонів є доцільним, набув вмінь та навичок реалізації шаблонів під час створення програмного коду.