Министерство образования и науки Российской Федерации
Санкт-Петербургский Политехнический Университет Петра Великого
—
Институт компьютерных наук и кибербезопасности

ЛАБОРАТОРНАЯ РАБОТА №2

«Рефакторинг программы»

по дисциплине «Объектно-ориентированное программирование»

Санкт-Петербург
2024

1 ЦЕЛЬ РАБОТЫ

Целью работы – освоение особенностей этапов анализа предметной области и проектирования архитектуры программного обеспечения в объектной модели.

2 ЗАДАЧИ

В рамках выполнения лабораторной работы необходимо решить следующие задачи:

— анализ предметной области заданной программы, реализованной в процедурной парадигме программирования с выделением ключевых абстракций;

— анализ недостатков архитектуры и кода исходного проекта;

— проектирование новой архитектуры программы в объектной парадигме программирования (рефакторинг) с использованием UML-диаграммы классов (см. UML.docx);

— программная реализация новой архитектуры на языке программирования Python или C++;

— реализовать unit-тесты для основного функционала программы.

3 УСЛОВИЕ

До начала выполнения лабораторной работы исходная программа, для которой будет выполнятся рефакторинг, должна быть согласовано. Выбранная программа должна удовлетворять следующим требованиям:

— программа должна быть реализована в процедурной парадигме;

— в рамках предметной области программы должна быть возможность выделить не менее четырёх ключевых абстракций (классов);

— программа должна содержать не менее 300 строк кода;

— язык программирования исходного проекта – Python, Си (проекты на других языках программирования могут быть использованы только в случае индивидуального согласования с преподавателем).

В случае отсутствия подходящего проекта может быть использован один из предложенных в приложение 1 вариантов.

При выполнения лабораторной работы необходимо соблюдать следующие условия:

1. Итоговая архитектура проекта в объектной парадигме должна содержать не менее пяти классов (с учётом ключевых абстракций предметной области и механизмов реализации).
2. В случае использования языка программирования Python все методы должны содержать аннотацию типов данных (механизм type hinting);
3. Должно быть разработано не менее пяти unit-тестов.
4. Unit-тесты должны покрывать ключевое поведение основных классов программы.

4 ЭТАПЫ ВЫПОЛНЕНИЯ

Процесс выполнения лабораторной работы включает в себя следующие этапы:

1. Первый этап (*занятие №1*):

— выполнение декомпозиции функционала исходного проекта в объектной парадигме (выделение ключевых абстракций предметной области);

— проектирование архитектуры, поддерживающий полный перечень функционала исходного проекта;

— оформление итоговой архитектуры в виде UML-диаграммы классов.

2. Второй этап (*занятие №2*):

— рефакторинг кода проекта в соответствии с разработанной архитектурой;

— разработка unit-тестов для проекта.

5 ДОПОЛНИТЕЛЬНЫЕ ЗАДАНИЯ

5.1 Дополнительные UML-диаграммы (структуры)

Цель дополнительного задания – получение опыта работы со структурной UML-диаграммой объектов.

В рамках выполнения дополнительного задания необходимо:

1. Изучить нотацию UML-диаграммы объектов.
2. Выбрать как минимум два специфичных состояния программы для их описания с помощью UML-диаграммы объектов.
3. Построение как минимум двух UML-диаграмм объектов.

В отчёте необходимо привести:

1. Описание выбранных состояние программы.
2. Построенные UML-диаграммы.

5.2 Дополнительные UML-диаграммы (поведения)

Цель дополнительного задания – получение опыта работы с поведенческими UML-диаграммой объектов.

В рамках выполнения дополнительного задания необходимо:

1. Изучить нотацию UML-диаграммы активности.
2. Выбрать сценарий использования программы для его описания с помощью UML-диаграммы активностей.
3. Построение UML-диаграмм активностей.

В отчёте необходимо привести:

1. Описание выбранного сценария использования программы.
2. Построенную UML-диаграмму.

5.3 Дополнительные UML-диаграммы (поведения)

Цель дополнительного задания – изучение реализации принципов наследования и полиморфизма компиляторами языка программирования (ЯП) Си++.

В рамках выполнения дополнительного задания необходимо:

1. Изучить механизм виртуальных таблиц (virtual function table) функций ЯП Си++.
2. Реализовать тестовую программу, включающую четыре класса:
   — интерфейс;
   — абстрактный класс, конкретно реализующий как минимум одну из функций интерфейса;
   — две конкретных реализации абстрактного класса, выполняющих переопределение (override) конкретной функции абстрактного класса.
3. С помощью отладчика продемонстрировать возможность подмены функции конкретного класса на функцию из родительского класса:
   — виртуальную функцию родительского класса (virtual);

— абстрактную функцию родительского класса без реализации (pure virtual);

В отчёте необходимо привести:

1. Выбранную модель и версию компилятора ЯП Си++;
2. Описание механизма работы наследования с помощью virtual function table;
3. Исходный код тестовой программы;
4. Примеры подмены функции (до изменения, после изменения) с указанием разницы в консольном выводе программы.

6 ТРЕБОВАНИЯ К ОТЧЕТУ

Отчет должен включать следующие пункты:

1. Цель работы.
2. Задачи с указанием выполненных дополнительных заданий.
3. Ход работы – краткое описание этапов выполнения работы:
    a. краткое описание исходного проекта с указанием предметной области, функционала проекта и важных особенностей реализации;
    b. описание недостатков исходного проекта, связанных с: изъянами архитектуры, низким качеством кода.
    c. описание разработанной архитектуры программы в объектной парадигме, представленное: UML-диаграммой классов, таблицей с перечнем классов с текстовым описанием их назначения;
    d. описание особенностей программной реализации новой архитектуры;
    e. таблица с описанием разработанных unit-тестов и примерами тестирования программы.
4. Выводы.

Приложение 1 – Перечень предлагаемых для рефакторинга проектов

№	Ссылка на репозиторий	ЯП	Предметная область
1	https://github.com/RuyaKH/Pygame-Hangman	Python	Игра на угадывание слов «виселица»
2	https://github.com/idriss-ensias/anim-snake-pygame/tree/master	Python	Игра «змейка»
3	https://github.com/domarp-j/minesweeper	Python	Игра «сапёр»
4	https://github.com/kovzol/tank	Python	Игра «танки»
5	https://github.com/Asimudin/pygame-pacman-ai-1	Python	Игра «PacMan»
6	https://github.com/NikolayZdravkov/PYTHON-CHESS-PROJECT	Python	Игра «шахматы»
7	https://github.com/AndreaVidali/ChineseCheckersAI	*Python*	*Игра «шашки» (китайские)*
8	https://github.com/Rashadmlkv/Checker	*Python*	*Игра «шашки»*
9	https://github.com/yuraxdrumz/tank-game-with-python	*Python*	*Игра «танки»*
10	https://github.com/iamabhi898/Sudoku-with-Python3	*Python*	*Игра «судоку»*