Федеральное государственное бюджетное образовательное учреждение высшего образования

«Омский государственный технический университет»

Кафедра «Автоматизированные системы обработки информации и управления»

Расчетно-графическая работа

по дисциплине «Конструирование программного обеспечения»

на тему: «Расчёт зарплаты»

Выполнил:

студент гр. ПЭ-171

Ибраимов А.А.

Проверил: доц.

Убалехт И.П.

Омск 2020

**Задание**

Вариант: 6.

Текст задания: «Предметная область «Начисление заработной платы». Есть предприятие в нём несколько групп сотрудников, инженеры, рабочие, менеджеры по продажам. Программа должна отражать начисление зарплат для всех выше перечисленных категорий сотрудников, следующим образом: для инженеров зарплата начисляется как оклад плюс премия если есть, для рабочих в зависимости от отработанных часов, сверхурочные часы оплачиваются в двойном размере, для менеджеров по продажам, оклад плюс бонусы за каждую совершённую сделку».

Содержание

[Введение 4](#_Toc41288426)

[1. Архитектура системы 5](#_Toc41288427)

[2. Программный дизайн системы 6](#_Toc41288428)

[3. Демонстрация работы системы 8](#_Toc41288429)

[Заключение 9](#_Toc41288430)

[Библиографический список 10](#_Toc41288431)

# Введение

Конструирование программного обеспечения - дисциплина программной инженерии. Это детальное создание работающего программного обеспечения посредством комбинации кодирования, верификации, модульного тестирования, интеграционного тестирования и отладки. Оно связано со всеми другими дисциплинами программной инженерии, наиболее сильно с проектированием и тестированием [1].

Основы построения программного обеспечения включают:

• минимизацию сложности;

• предвидение изменений;

• создание для проверки;

• повторное использование;

• стандарты в разработке.

ОО-конструирование ПО - это метод разработки ПО, который строит архитектуру всякой программной системы на модулях, выведенных из типов объектов, с которыми система работает (а не на одной или нескольких функциях, которые она должна предоставлять) [2].

# Архитектура системы

В первую очередь, в начале разработки системы, было принято решение разбить её на 2 слоя (рисунок №1):

* Слой представления – обеспечивает взаимодействие с конечным пользователем;
* Бизнес-слой – содержит в себе всю бизнес логику и выполняет необходимы расчеты.

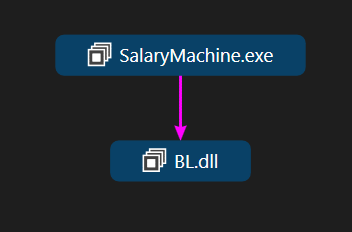


Рисунок №1 – Архитектура системы

Такое решение обусловлено возможностью быстрых и легких изменений системы, таких как добавление/редактирование слоя представления, ведь в таком случае внесенные изменения не повлияют на работоспособность бизнес-слоя, и т.п.

# Программный дизайн системы

Основные архитектурные задачи пришлись на разработку бизнес-слоя приложения (рисунок 2).

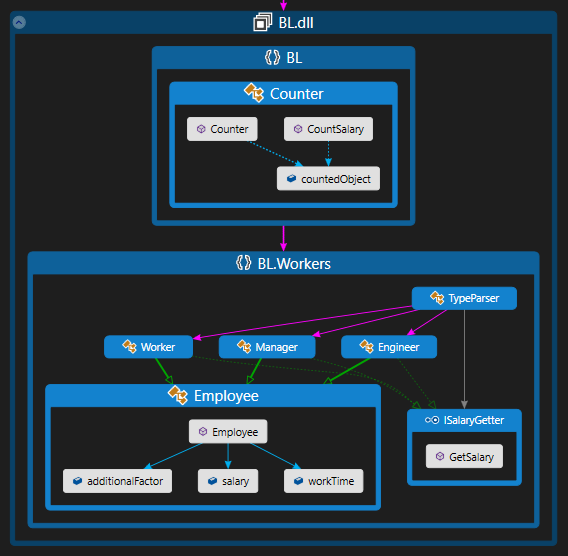


Рисунок №2 – Дизайн бизнес-слоя

В классе Counter была использована инъекция зависимостей (рисунок №3), потому что становится легче создавать альтернативные реализации данного типа сервиса, а потом указывать, какая именно реализация должна быть использована, причём без изменений в объектах, которые этот сервис используют. Это особенно полезно в юнит-тестировании, потому что вставить реализацию «заглушки» сервиса в тестируемый объект очень просто.

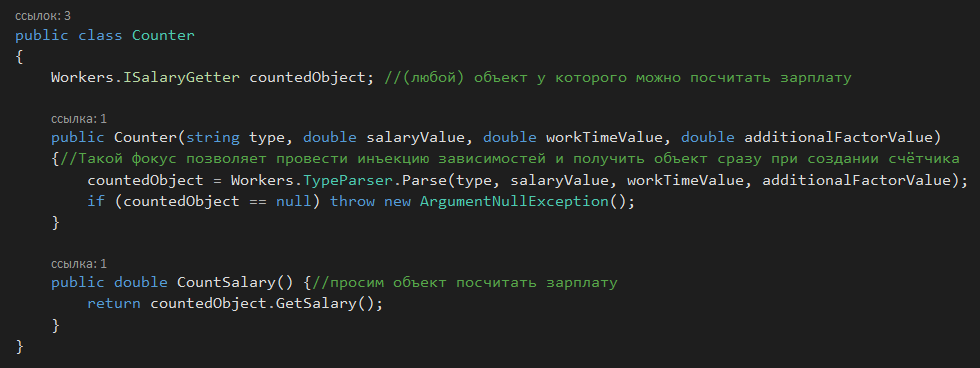


Рисунок №3 – Инъекция зависимостей

Перед созданием классов работников, был создан абстрактный класс Employee, наследуясь от которого наследник получал все необходимые базовые поля и повысить использование кода.

Поскольку для каждого типа работника был создан отдельный класс, было принято решение о разработки единого интерфейса, который позволил бы запрашивать зарплату у каждого вида работника вне зависимости от типа (рисунок №4).

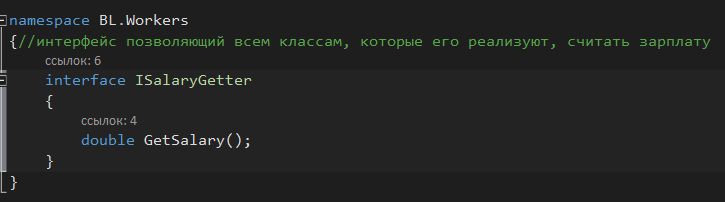


Рисунок №4 – Интерфейс

Выделение отдельного класса TypeParser обусловлено одной из концепций SOLID, а именно single responsibility, что в конечном итоге смогло вынести обязанность определения типа работника в отдельный класс и повысить сцепленность (cohesion) системы, не повышая связность.

# Демонстрация работы системы

Запустив программу, пользователь вводит/выбирает данные, затем жмёт на кнопку расчета и видит результат вычислений (рисунок№5).

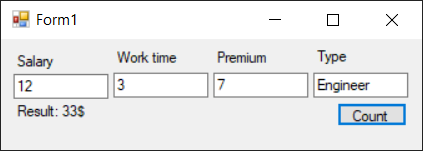


Рисунок №5 – Работа системы

При некорректном вводе данных система сообщает об ошибке (рисунок №6).

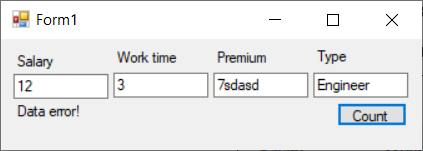


Рисунок №6 – Ошибка ввода

# Заключение

В рамках выполнения работы изучили и применили на практике основные концепции проектирования программного обеспечения, также в рамках проектирования использовали различные паттерны и углубили знания о них.

# Библиографический список

1. SWEBOOK V3.0, Software engineering: cs.fit.edu. – URL: <https://cs.fit.edu/~kgallagher/Schtick/Serious/SWEBOKv3.pdf> (дата обращения: 23.04.2020). – Текст: электронный.
2. Meyer B. Object-Oriented Software Construction, B. Meyer – 2-е изд., – Аппер Садл Ривер, Нью-Джерси, 1997, –1254 стр.