Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский Государственный Технологический Университет»

Кафедра информационных систем и программирования

Отчет по лабораторной работе №6

По дисциплине «Рефакторинг и работа с унаследованным кодом»

«Рефакторинг с использованием порождающих паттернов проектирования»

Выполнил студент

Группы 21-КБ-ПР2

Жулькин Д.П.

Преподаватель:

Степанова Елизавета Владимировна

Краснодар

2024

Цель работы: Изучить техники рефакторинга с использованием порождающих паттернов проектирования.

Задание:

1) Преобразовать процесс создания объектов в методе Main класса Program в соответствии с предложенными в пункте 2.2 изменениями, применив описанные в теории методы рефакторинга.

2) Проверить согласованность результатов и корректность работы нового варианта программы с результатами старой версии.

3) Оформить отчёт.



Рисунок 1 – Вычленение метода CreateBill из Main

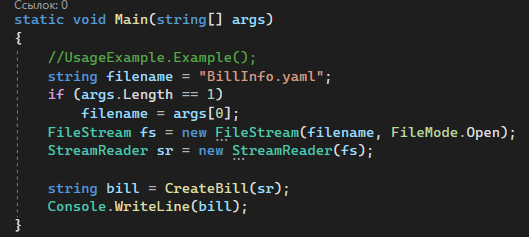


Рисунок 2 – Состояние метода Main

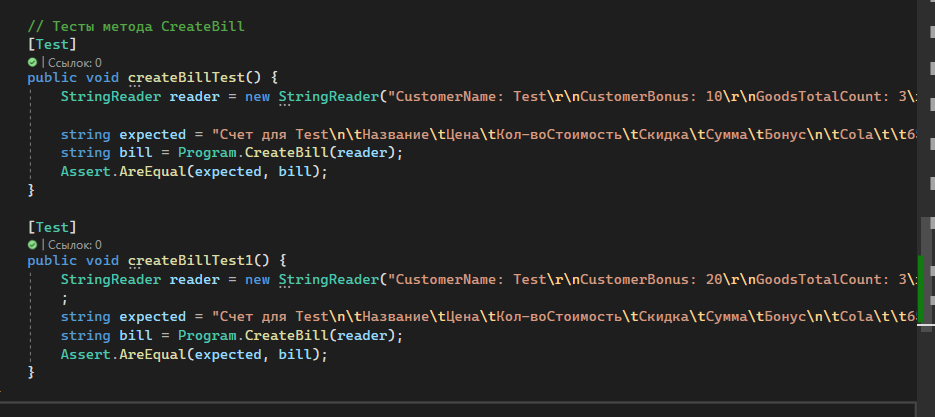


Рисунок 3 – Новые тесты для метода CreateBill

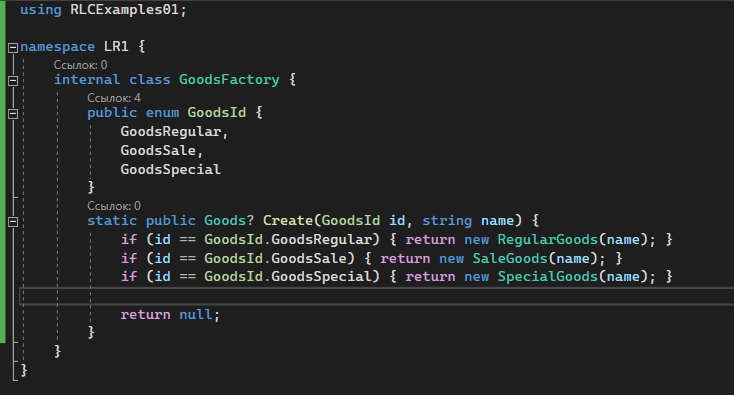


Рисунок 4 – Новый класс GoodsFactory



Рисунок 5 – Новый класс ContentFile

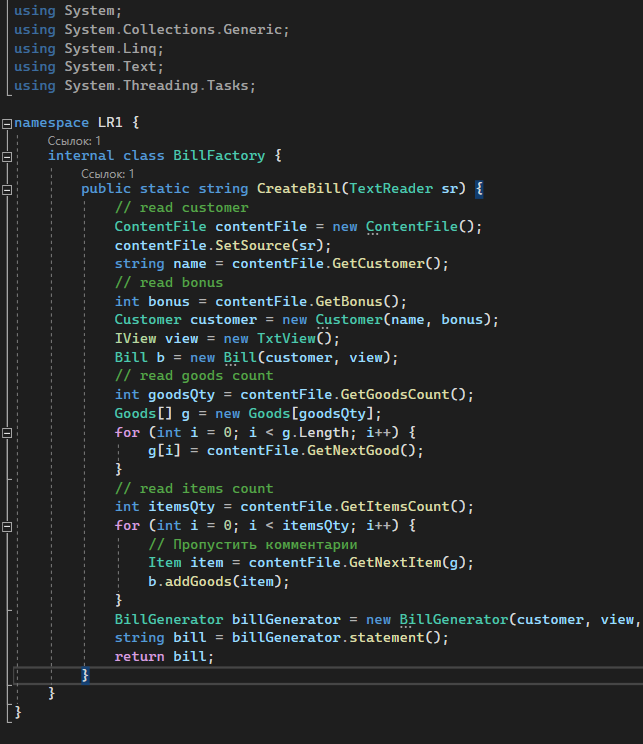


Рисунок 6 – Метод CreateBill перенесен в новый класс BillFactory

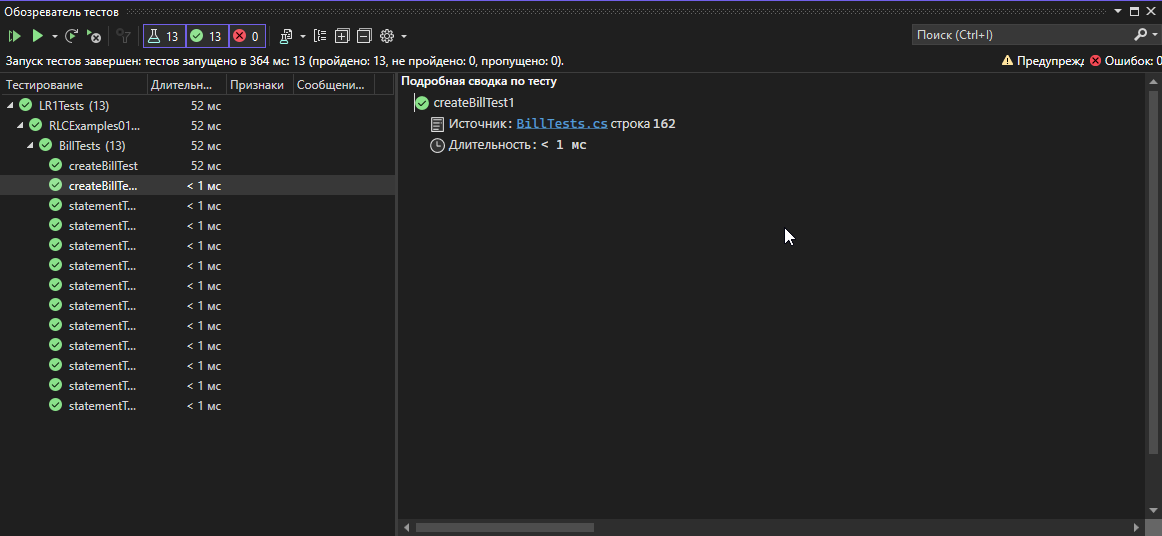


Рисунок 7 – Все тесты выполняются успешно

Вывод: все задания выполнены успешно. Логика программы не была нарушена, об этом свидетельствуют успешно пройденные тесты.

**1) Что такое паттерн проектирования?**

**Паттерн проектирования** — это обобщенное решение проблемы проектирования программного обеспечения, которое может быть применено в различных контекстах. Он представляет собой описание архитектурного элемента, который может быть использован многократно для решения повторяющихся проблем.

Паттерны проектирования не являются готовыми решениями, а скорее шаблонами, которые могут быть адаптированы к конкретным задачам. Они помогают разработчикам создавать более надежные, гибкие и масштабируемые системы.

**2) Для чего используется паттерн Factory method?**

**Паттерн Factory method** используется для создания объектов, не оговаривая конкретные классы этих объектов. Это позволяет создавать объекты без необходимости знать их конкретную реализацию.

Паттерн Factory method может быть полезен в следующих случаях:

* **Когда необходимо создать объект, но его конкретная реализация может меняться в зависимости от контекста.**
* **Когда необходимо создать объект, не зная его конкретного класса.**
* **Когда необходимо сделать код более гибким и масштабируемым.**

**3) Как проверять результаты работы программы в случае ее неприспособленности для автоматизированного тестирования?**

В случае, если программа не приспособлена для автоматизированного тестирования, можно использовать следующие методы проверки ее работы:

* **Ручное тестирование:** тестировщики вручную выполняют различные сценарии работы программы и проверяют ее результаты.
* **Визуальное тестирование:** тестировщики проверяют пользовательский интерфейс программы на наличие ошибок.
* **Нагрузочное тестирование:** тестировщики проверяют работоспособность программы при высоких нагрузках.
* **Отладка:** разработчики используют специальные инструменты для отладки программы и поиска ошибок.

**Дополнительные методы:**

* **Использование инструментов статического анализа:** эти инструменты могут помочь найти потенциальные ошибки в коде программы.
* **Использование тестовых фреймворков:** хотя программа может быть не приспособлена для автоматизированного тестирования, можно использовать тестовые фреймворки для упрощения ручного тестирования.

**Важно отметить, что отсутствие автоматизированного тестирования может значительно увеличить время и трудоемкость разработки программы.**

**Рекомендуется:**

* **По возможности приспособить программу для автоматизированного тестирования.**
* **Использовать различные методы тестирования для обеспечения всесторонней проверки работы программы.**