РАЗРАБОТКА МАСШТАБИРУЕМЫХ ПРОГРАММ НА ПРИМЕРЕ ГЕНЕРАТОРА КОНТРОЛЬНЫХ СУММ

Никитин Д.А.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Парафиянович Т.А. – канд.пед.наук, доцент, доцент кафедры ИРТ

В статье речь идёт о выборе структурных элементов системы, способах их соединения, проектировании архитектуры программы и разработке гибкого и масштабируемого программного средства на языке C++ в виде консольной утилиты; в качестве примера определена тема «Генератор контрольных сумм»; масштабирование системы в данном случае заключается в применении порождающего шаблона проектирования «Фабричный метод».

Разработка программных средств является сложным процессом, состоящим из определённых этапов. При рассмотрении функциональной составляющей, самым важным этапом выступает проектирование системы. Проектирование программного продукта – это процесс создания проекта программного обеспечения. Проектированию подлежат: архитектура, устройство компонентов и пользовательские интерфейсы [1]. В данном исследовании речь идёт о проектировании архитектуры программы. При грамотном выборе структурных элементов системы и способах их соединения можно получить программный продукт, который в последующем будет легко поддерживать. Практическая деятельность по разработке программ показывает, что процесс написания программных модулей в таком случае станет проще.

Важным моментом в проектировании архитектуры являются абстракции и интерфейсы, позволяющие создать независимость между различными слоями программы. В некоторых случаях при разработке программных средств требования, предъявляемые к программе, могут изменяться, а хорошо построенная архитектура позволит минимизировать усилия на изменения направлений разработки.

В рамках исследования поставлена цель разработать программное средство на языке C++, позволяющее определять контрольные суммы файлов. Оно должно быть представлено в виде консольной утилиты. На этапе проектирования определено требование о возможности выбора

алгоритма расчёта контрольных сумм пользователем. Проблема в данном случае сводится к выбору объекта конкретного типа на основании полученной строки.

Наиболее простым способом выступает условный оператор или механизм управления выбором «switch-case». В такой ситуации, при рассмотрении системы на бесконечности, такая конструкция станет громоздкой, а внутри каждого блока будет присутствовать несчётное количество дублирующегося кода.

Для решения этой задачи были рассмотрены порождающие шаблоны проектирования. В частности был рассмотрен шаблон «Фабричный метод». В некоторых случаях использование шаблонов проектирования может быть неуместно и вызовет сложности в разработке, поэтому перед применением шаблона необходимо ознакомиться с руководством, в котором описаны случаи, когда применение шаблона уместно. В рекомендациях по применению шаблона «Фабричный метод» одним из случаев является ситуация, когда заранее неизвестно объект, какого типа должен быть создан [2].

Суть шаблона заключается в том, что он предлагает создать абстракцию в виде создаваемого объекта и создателя объектов, а на их основе разрабатывать конкретные объекты и создателей соответственно. Однако в руководстве не сказано, как реализовать выбор необходимого объекта. Фабричный метод можно использовать в совокупности с условными конструкциями. В таком случае объём работы снизится, однако этот способ остаётся сложным и запутанным.

Разработка происходит в объектно-ориентированной парадигме, а соответственно каждый метод хеширования имеет свой уникальный тип или же имя класса, это свойство может послужить критерием выбора. В таком случае наименования классов должны соответствовать наименованиям алгоритмов хеширования.

Основная идея решения заключается в создании контейнера, в котором будет расположен список всех объектов. Тогда для определения конкретного элемента можно проводить перебор по списку и сравнивать тип с введённой строкой. Такое решение подразумевает изменение ролей участников шаблона, а именно, в качестве конкретного создателя выступает контейнер со списком всех элементов.

Однако в программе должны быть дополнительные команды, например справка или просмотр версии программы. Исходя из реализованного абстрактного создателя можно, по аналогии с создателем алгоритмов хеширования, реализовать контейнер для дополнительных команд программы. Таким образом, абстрактный создатель необходимо сделать шаблонным классом. Для абстрактного создателя необходимо добавить шаблонный список (в качестве списка выбран вектор) и объект для передачи, также шаблонного типа.

Выбранный способ поиска объектов обладает независимостью от основного слоя программы и расширяемости. При появлении новых видов объектов необходимо выполнить единственное действие – добавить объект в список.

Для наглядного представления структуры проекта составлена упрощённая диаграмма классов (рисунок 1).

Рисунок 1 – Упрощённая диаграмма классов разработанного программного средства

В ходе исследования были детально рассмотрены варианты для разработки масштабируемого программного средства, спроектирована структура с применением шаблона «Фабричный метод» и его расширением в виде списка объектов для создания. Ознакомиться с детальной реализацией структуры и самой программы можно в репозитории GitHub «denden1s/HashSums».

Список использованных источников:

1. Проектирование программного обеспечения [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Проектирование_программного_обеспечения. – Дата доступа: 30.03.2022.

2. Фабричный метод (Factory Method) [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/patterns/2.1.php. Дата доступа: 30.03.2022.