Міністерство освіти і науки України
Національний технічний університет України “Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №5
З дисципліни «Технології розроблення програмного забезпечення»
Тема: **«Шаблони ADAPTER, BUILDER, COMMAND, CHAIN OF RESPONSIBILITY, PROTOTYPE»**
HTTP-сервер

Виконав:
Студент групи ІА-22
Білецький С. В.

Перевірив:
Мягкий М. Ю.

Київ-2024

Зміст

Тема:
Шаблони «adapter», «builder», «command», «chain of responsibility», «prototype»

Мета:
Навчитися застосовувати шаблони проєктування «Adapter», «Builder», «Command», «Chain of Responsibility» та «Prototype» у процесі розробки програмного забезпечення. Ознайомитися з їхньою концепцією, особливостями реалізації та можливостями використання для створення гнучких і масштабованих рішень.

Завдання:
HTTP-сервер. Сервер повинен мати можливість розпізнавати вхідні запити і формувати коректні відповіді (згідно протоколу HTTP), надавати сторінки chtml (html сторінки з додаванням найпростіших C# конструкцій на розсуд студента), вести статистику вхідних запитів, обробку запитів у багатопотоковому/подієвому режимах. **(Builder)**

Хід роботи

1. Реалізувати не менше 3-х класів відповідно до обраної теми

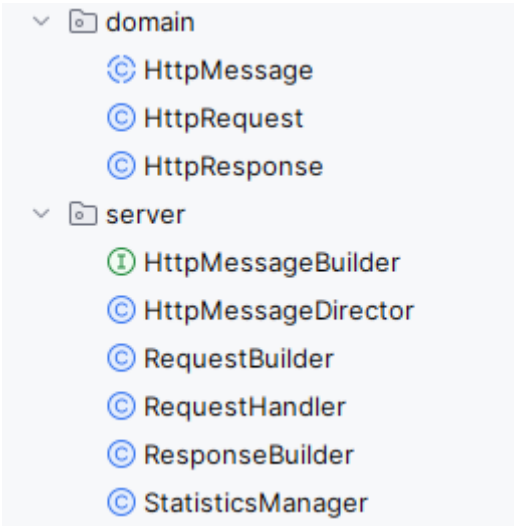

Рис. 1 — Структура класів

У ході виконання лабораторної роботи було реалізовано функціонал 7 класів, що зображені на рис. 1 (певні класи реалізовані частково та мають потенційні можливості для доповнення у зв’язку з тим, що система ще не повністю реалізована). Детальний опис кожного з класів:

1. **HttpMessage.** Це базовий клас для HTTP-повідомлень, який включає спільні для запиту та відповіді властивості: headers (заголовки) та body (тіло повідомлення). Слугує основою для створення спеціалізованих повідомлень (запитів і відповідей).
2. **HttpRequest.** Наслідує клас HttpMessage та додає специфічні для HTTP-запитів поля: method (HTTP-метод, наприклад, GET або POST) і path (шлях до ресурсу). Використовується для опису вхідних запитів у системі.
3. **HttpResponse.** Наслідує клас HttpMessage та додає поле statusCode, яке вказує на статус відповіді (наприклад, 200 для успіху або 404 для помилки). Використовується для створення вихідних відповідей.
4. **HttpMessageBuilder.** Інтерфейс, який визначає основні методи для білдерів (будівельників) HTTP-повідомлень. Забезпечує гнучкість і уніфікацію процесу створення як запитів, так і відповідей. Містить методи для налаштування заголовків (addHeader), тіла (setBody) та фінальної збірки повідомлення (build).
5. **RequestBuilder.** Конкретна реалізація HttpMessageBuilder для побудови об'єктів HttpRequest. Забезпечує методи для налаштування HTTP-методу (method) та шляху (path) запиту. Слугує для створення та налаштування HTTP-запитів у системі.
6. **ResponseBuilder.** Конкретна реалізація HttpMessageBuilder для побудови об'єктів HttpResponse. Додає методи для встановлення статус-коду (statusCode) відповіді. Використовується для створення та кастомізації HTTP-відповідей.
7. **HttpMessageDirector.** Визначає методи для керування білдерами HTTP-повідомлень. Забезпечує готові рішення для налаштування повідомлень із загальними стандартними заголовками, наприклад, buildJsonHttpMessage для JSON-формату або buildUTF8Message для тексту з кодуванням UTF-8. Директор абстрагує процес створення повідомлень, спрощуючи їх ініціалізацію та налаштування.

2. Реалізувати один з розглянутих шаблонів за обраною темою

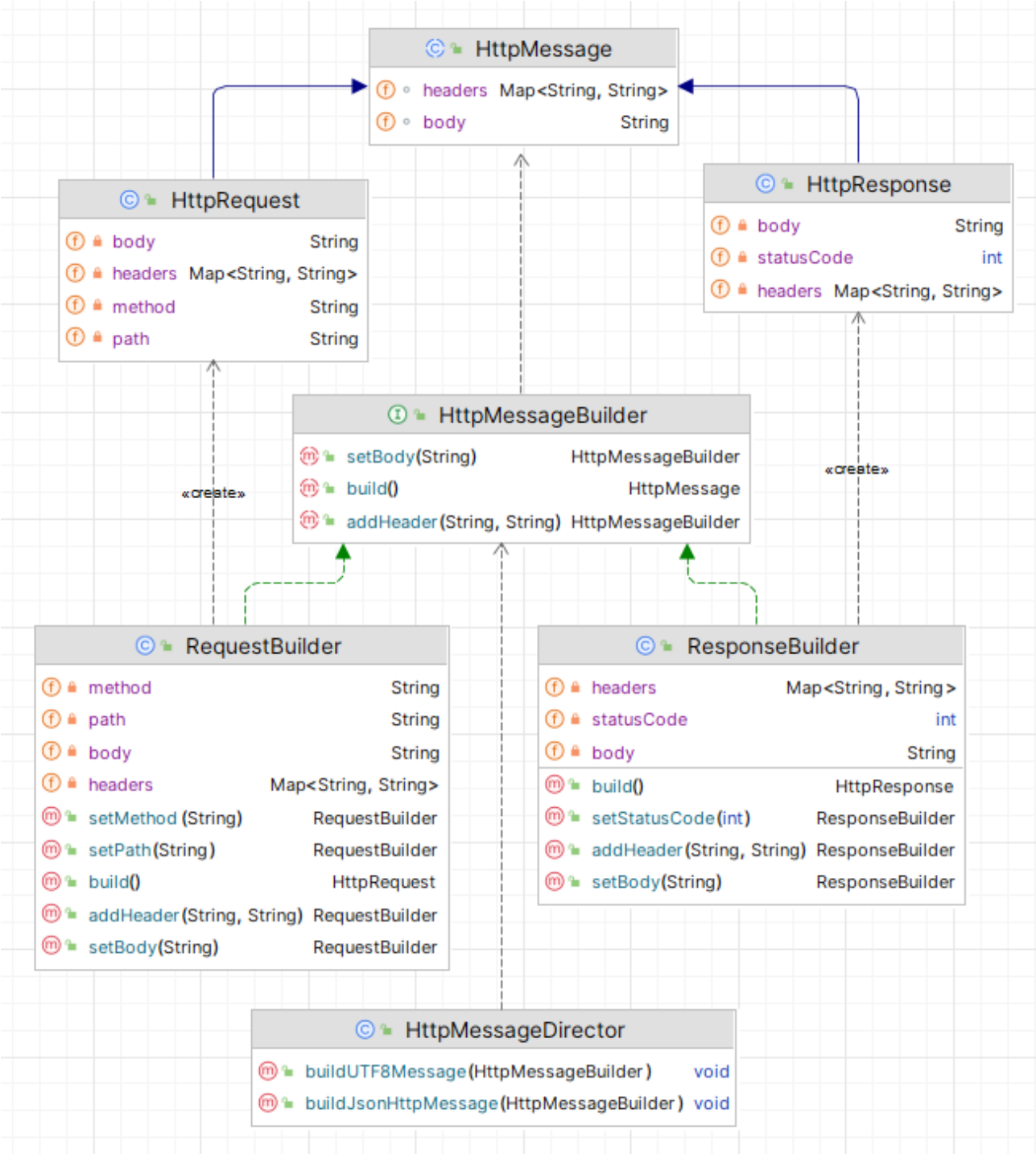

Рис. 2 — Реалізація патерну

У проєктах, де є складні об'єкти з багатьма опціональними параметрами, пряме створення таких об'єктів може бути незручним і призводити до нечитабельного коду. У нашому випадку, HTTP-повідомлення мають різні комбінації заголовків, методів, статус-кодів і тіл, що робить їхню ініціалізацію складною і громіздкою. Патерн Builder дозволяє вирішити цю проблему:

1. Забезпечує гнучкість при створенні об'єктів за допомогою поетапного налаштування.
2. Дозволяє уникнути конструкцій із багатьма параметрами, спрощуючи читабельність і підтримку коду.
3. Інкапсулює логіку створення об'єктів, забезпечуючи чітке розділення відповідальностей.

Реалізація

Патерн Builder було використано для створення та конфігурації HTTP-повідомлень, таких як запити (HttpRequest) та відповіді (HttpResponse), забезпечуючи зручний, поетапний спосіб їх побудови.

У нашій системі були створені окремі білдери: RequestBuilder для запитів і ResponseBuilder для відповідей. Вони реалізують спільний інтерфейс HttpMessageBuilder, що дозволяє уніфікувати процес налаштування повідомлень, наприклад, додавання заголовків або встановлення тіла повідомлення.

Для автоматизації додавання стандартних заголовків, таких як Content-Type або Charset, було реалізовано HttpMessageDirector. Він виступає директором, який використовує білдери для створення повідомлень із попередньо визначеними конфігураціями, наприклад, для JSON-формату або кодування UTF-8.

Переваги використання Builder:

1. Поетапне створення: Усі частини HTTP-повідомлення (заголовки, тіло, статус-код) можуть налаштовуватися окремо, що полегшує їх конфігурацію та тестування.
2. Модульність: Завдяки спільному інтерфейсу (HttpMessageBuilder) білдери для запитів та відповідей легко розширювати чи замінювати.
3. Читабельність коду: Код стає інтуїтивно зрозумілим, оскільки створення об'єкта виглядає як послідовність логічних кроків.
4. Повторне використання: Директор (HttpMessageDirector) дозволяє застосовувати готові сценарії створення стандартних повідомлень (наприклад, JSON-відповідей) у різних частинах системи.
5. Мінімізація помилок: Поетапний процес знижує ризик некоректної ініціалізації складних об'єктів.

Висновки:

У рамках лабораторної роботи ми опрацювали різні шаблони проєктування, зокрема Builder, який застосували для поетапного створення HTTP-запитів і відповідей. Реалізація патерну Builder спростила процес конфігурації складних об’єктів, підвищивши їхню читабельність і модульність. Це дозволило ефективно розділити логіку побудови повідомлень, полегшивши підтримку та розширення системи, продемонструвавши переваги цього патерну для створення масштабованих і легко налаштовуваних застосунків.