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 №8
Технологія розроблення програмного забезпечення

«COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR»
«Застосування шаблону «COMPOSITE» в темі Download Manager»
Варіант 26

Виконала
студентка групи ІА-13
Танасієнко Анастасія Вячеславівна

Київ 2023р.

Мета: дослідження шаблонів «COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR» реалізація шаблону проектування «COMPOSITE»

Шаблон проектування «COMPOSITE»

Шаблон проектування "Composite" відноситься до структурних патернів і використовується для об'єднання об'єктів в структури деревоподібної ієрархії для представлення їх як одиничних об'єктів. Цей шаблон дозволяє клієнтам обробляти як одиничні об'єкти, так і композитні структури однаковим способом.

Основні учасники шаблону "Composite":

Component (Компонент): Визначає інтерфейс для всіх конкретних об'єктів та композитів.

Leaf (Листок): Представляє одиничний об'єкт, який не має підкомпонентів.

Composite (Композит): Представляє композит, який має підкомпоненти. Реалізує методи інтерфейсу Component для роботи з підкомпонентами.

Принцип роботи шаблону полягає в тому, що інтерфейс для індивідуальних об'єктів і композитів є ідентичним, що дозволяє їх використовувати уніфіковано.

Шаблон проектування «FLYWEIGHT»

Шаблон проектування "Flyweight" відноситься до структурних патернів і використовується для оптимізації роботи з об'єктами, які мають спільні частини, шляхом винесення загальної частини в зовнішню структуру. Цей шаблон дозволяє економити ресурси, особливо коли кількість об'єктів велика і багато з них може використовувати одні й ті ж ресурси.

Основні учасники шаблону "Flyweight":

Flyweight (Легковаговик): Визначає інтерфейс, через який конкретні легковаговики можуть отримати та використовувати зовнішні ресурси.

ConcreteFlyweight (Конкретний легковаговик): Реалізує інтерфейс легковаговика та зберігає внутрішній стан, який може бути спільно використаний.

UnsharedConcreteFlyweight (Неподілений конкретний легковаговик): Має власні внутрішній стан, який не може бути спільно використаний з іншими об'єктами.

FlyweightFactory (Фабрика легковаговиків): Відповідає за управління легковаговиками, забезпечуючи можливість вилучення та повернення легковаговиків.

Принцип роботи шаблону полягає в тому, що спільні ресурси витягуються в окремий об'єкт (легковаговик), і замість створення нового об'єкта для кожного випадку використання ресурсів, використовується вже існуючий об'єкт.

Шаблон проектування «INTERPRETER»

Шаблон проектування "Interpreter" відноситься до поведінкових патернів і використовується для визначення граматики для мови та інтерпретації речень цієї мови. Цей шаблон дозволяє створити інтерпретатор, який читає, розуміє та виконує вирази, визначені у мові.

Основні учасники шаблону "Interpreter":

AbstractExpression (Абстрактний вираз): Визначає інтерфейс для інтерпретаторів.

TerminalExpression (Термінальний вираз): Реалізує інтерфейс абстрактного виразу для термінальних символів граматики.

NonterminalExpression (Нетермінальний вираз): Реалізує інтерфейс абстрактного виразу для нетермінальних символів граматики, об'єднує термінальні вирази та інші нетермінальні вирази.

Context (Контекст): Містить інформацію, яку інтерпретатор використовує для виконання операцій.

Client (Клієнт): Створює об'єкт контексту та вирази для інтерпретації мови.

Принцип роботи шаблону полягає в тому, що ми визначаємо граматику мови та створюємо вирази для інтерпретації цієї граматики. Клієнт створює об'єкт контексту та вирази, інтерпретатор яких здатен виконувати вирази, представлені у мові.

Шаблон проектування «VISITOR»

Шаблон проектування "Visitor" відноситься до поведінкових патернів і використовується для визначення нової операції над об'єктами без зміни їхньої структури. Цей шаблон дозволяє визначати нові алгоритми, не змінюючи класи об'єктів, над якими ці алгоритми використовуються.

Основні учасники шаблону "Visitor":

Visitor (Відвідувач): Визначає інтерфейс з методами відвідувача для кожного класу елемента.

ConcreteVisitor (Конкретний відвідувач): Реалізує інтерфейс відвідувача та визначає конкретні алгоритми для кожного класу елемента.

Element (Елемент): Визначає інтерфейс для об'єктів, над якими можуть бути використані операції відвідувача.

ConcreteElement (Конкретний елемент): Реалізує інтерфейс елемента та визначає специфічну для класу операцію відвідувача.

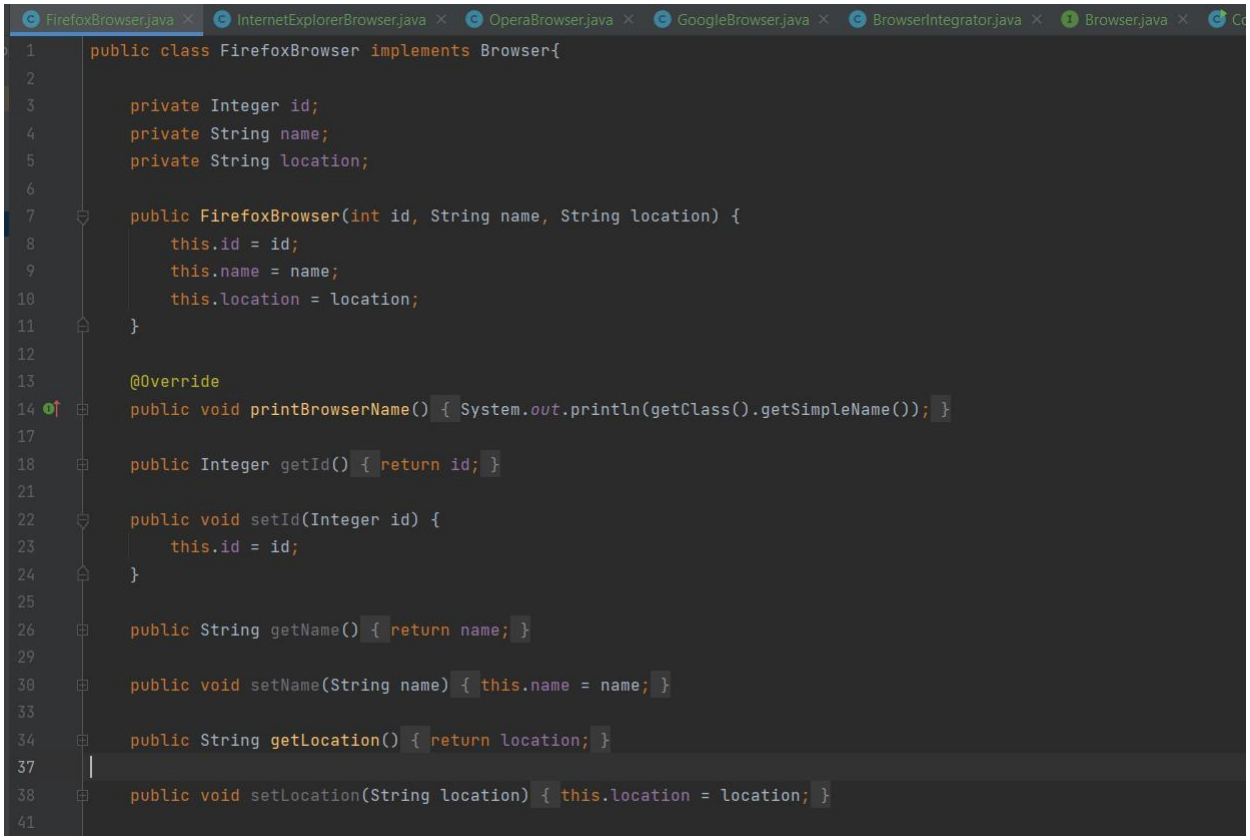
ObjectStructure (Об'єктна структура): Зберігає колекцію об'єктів і надає інтерфейс для їхньої ітерації.

Принцип роботи шаблону полягає в тому, що нові алгоритми визначаються в класі відвідувача, і кожен клас елемента має метод accept(), який приймає відвідувача та викликає відповідний метод відвідувача.

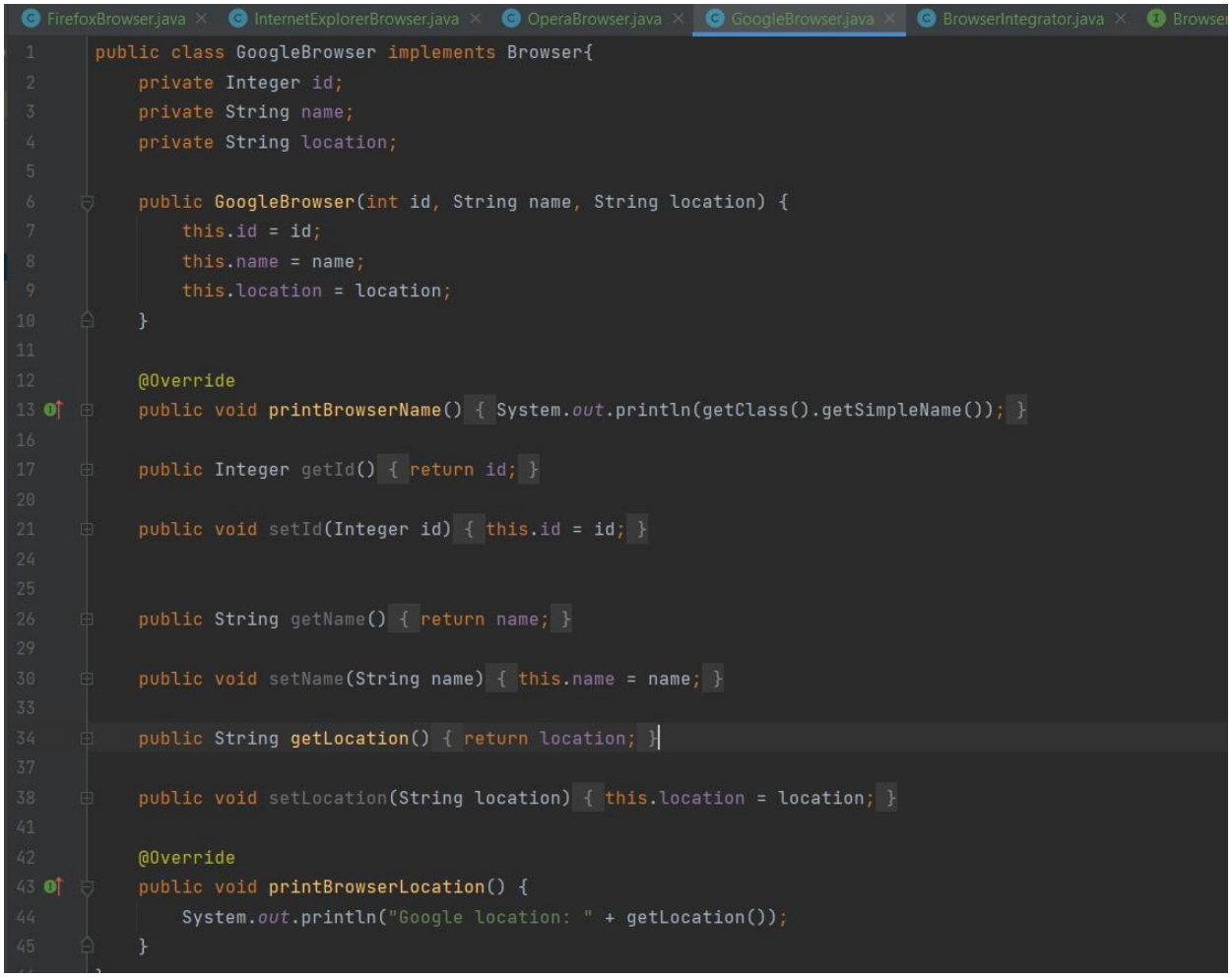
Дослідження шаблону «Composite»

Інтерфейс "Browser" має методи "printBrowserName" (вивести назву браузера) і "printBrowserLocation" (вивести розташування браузера).

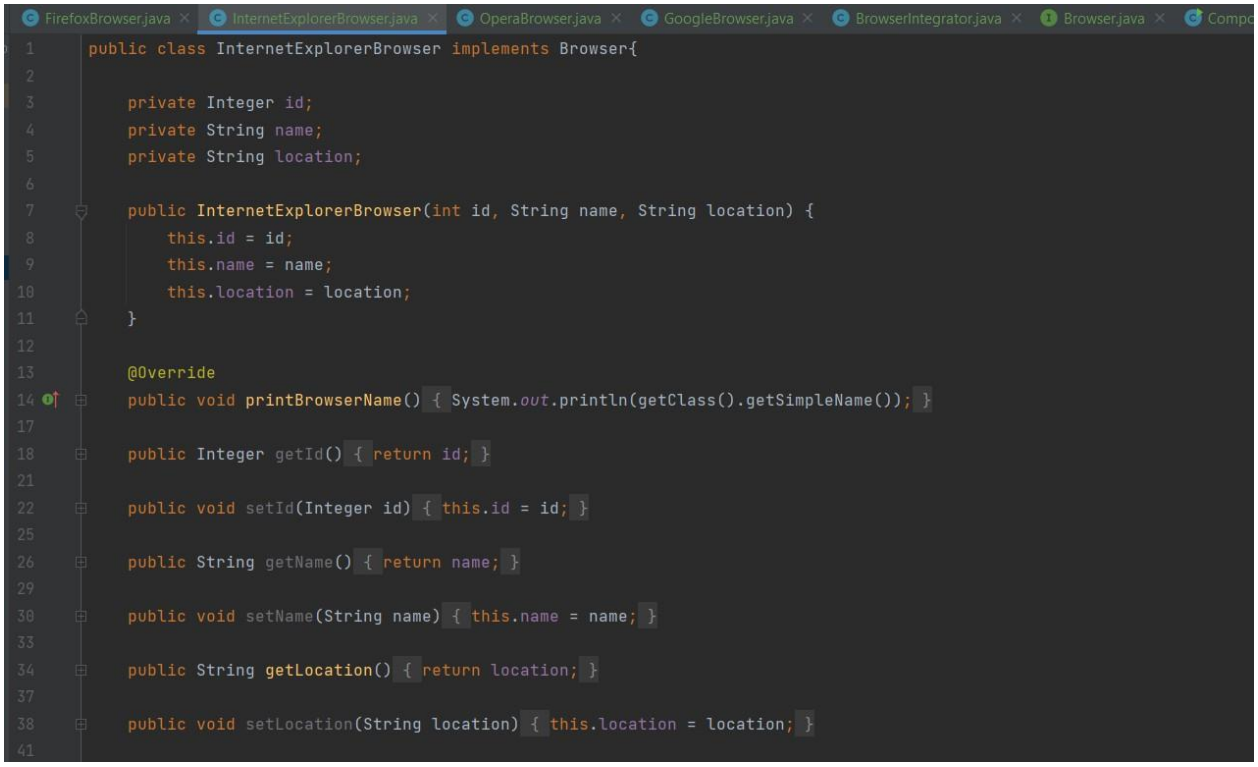
Цей клас FirefoxBrowser реалізує інтерфейс Browser. Він має поля для ідентифікації, назви та розташування браузера. Конструктор приймає ці параметри при створенні об'єкта. Методи printBrowserName та printBrowserLocation виводять відповідно назву браузера та його розташування. Клас надає також методи доступу до полів (getId, getName, getLocation) та їх змінення (setId, setName, setLocation).

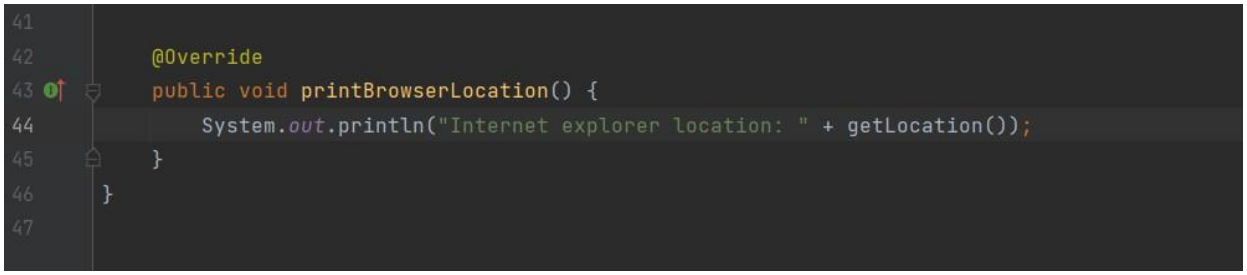


Цей клас GoogleBrowser реалізує інтерфейс Browser та має поля для ідентифікації, назви та розташування браузера. Конструктор приймає ці параметри при створенні об'єкта. Методи printBrowserName та printBrowserLocation виводять відповідно назву браузера та його розташування. Клас надає також методи доступу до полів (getId, getName, getLocation) та їх змінення (setId, setName, setLocation).

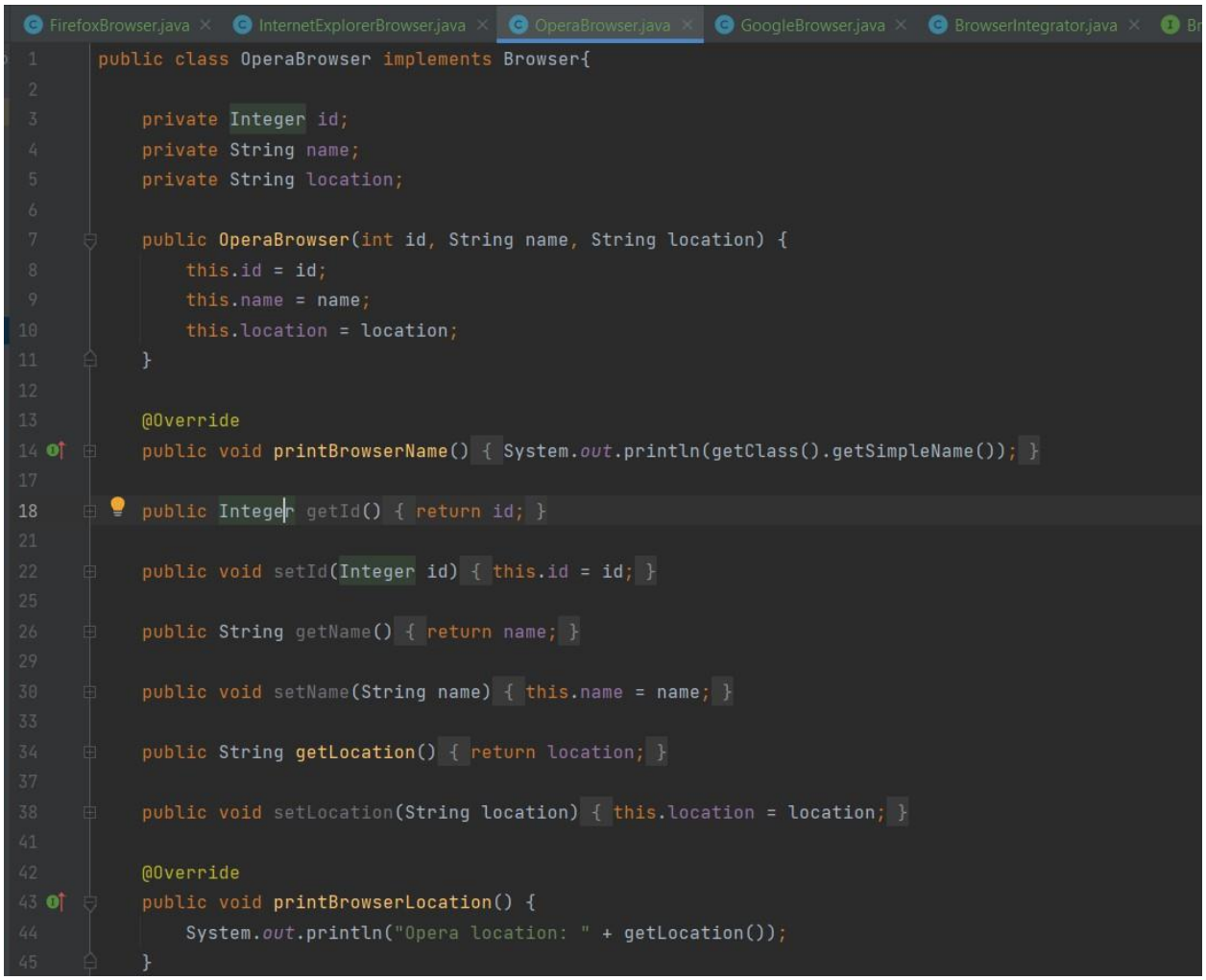


Цей клас InternetExplorerBrowser реалізує інтерфейс Browser та має поля для ідентифікації, назви та розташування браузера. Конструктор приймає ці параметри при створенні об'єкта. Методи printBrowserName та printBrowserLocation виводять відповідно назву браузера та його розташування. Клас надає також методи доступу до полів (getId, getName, getLocation) та їх змінення (setId, setName, setLocation).

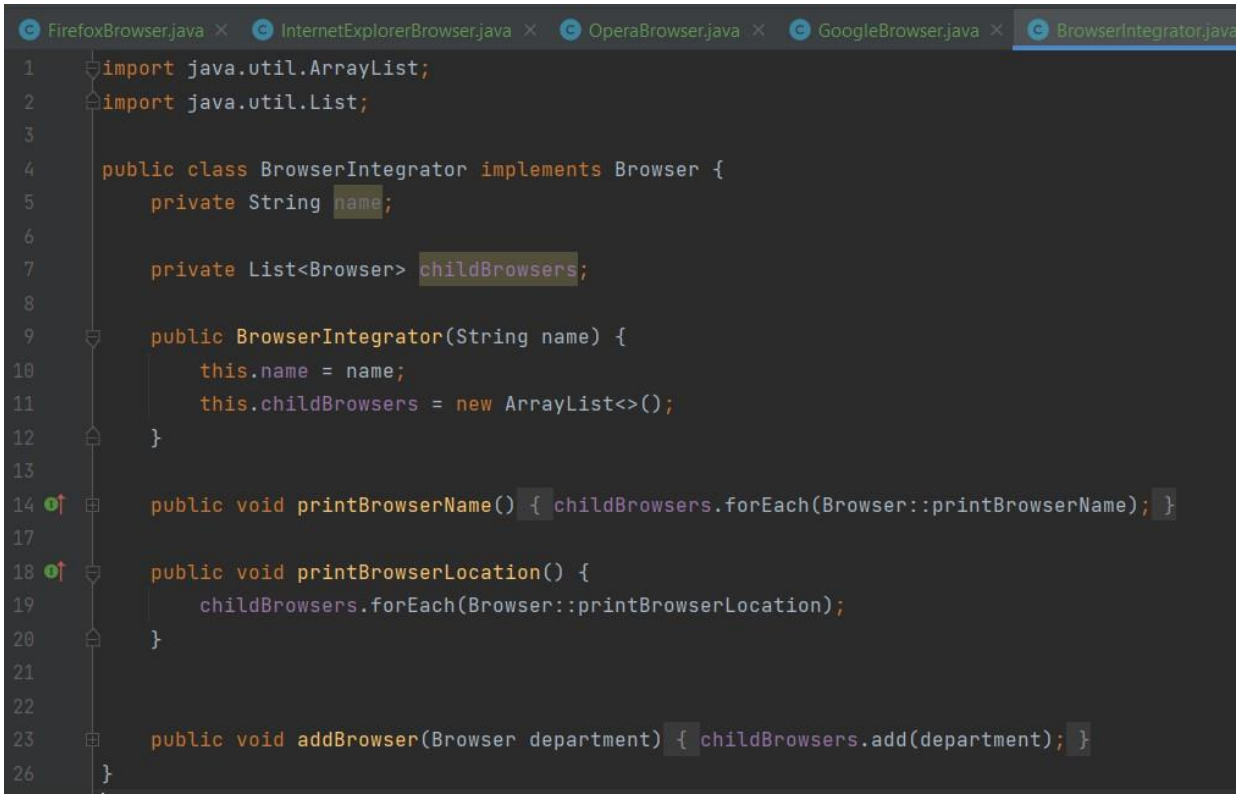




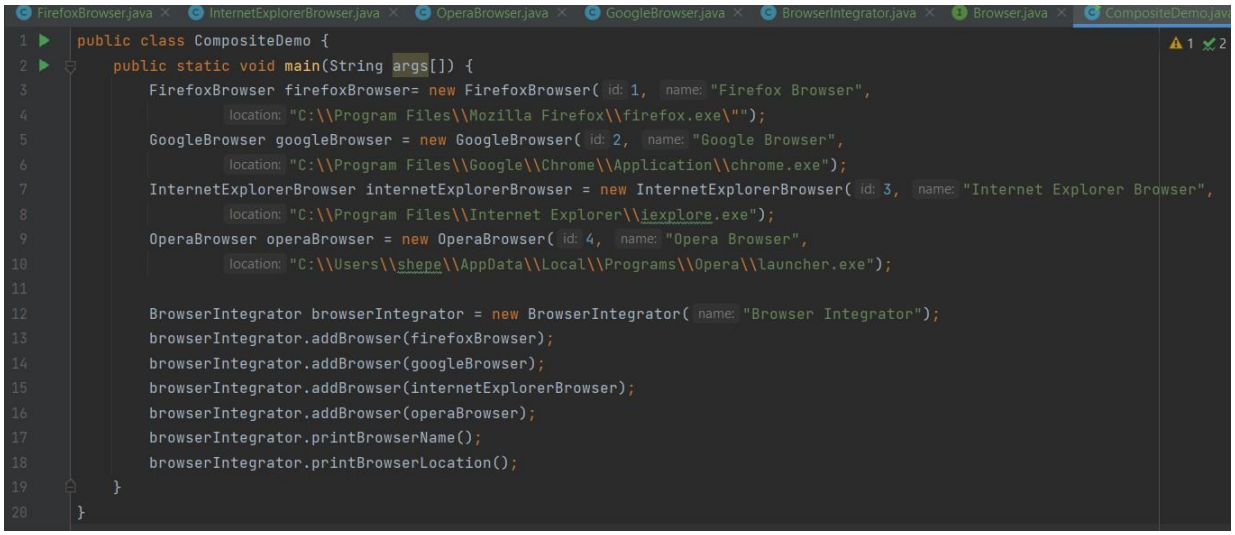
Цей клас OperaBrowser реалізує інтерфейс Browser та має поля для ідентифікації, назви та розташування браузера. Конструктор приймає ці параметри при створенні об'єкта. Методи printBrowserName та printBrowserLocation виводять відповідно назву браузера та його розташування. Клас надає також методи доступу до полів (getId, getName, getLocation) та їх змінення (setId, setName, setLocation).



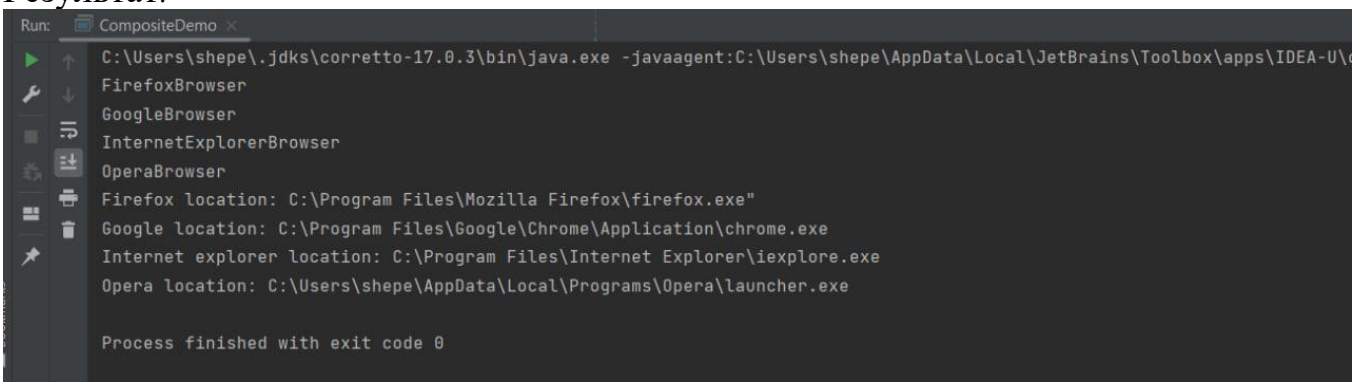
Клас BrowserIntegrator реалізує інтерфейс Browser та представляє інтегратор браузерів. Він має поле для назви та список дочірніх браузерів (childBrowsers). Конструктор приймає назву та ініціалізує порожній список дочірніх браузерів. Методи printBrowserName та printBrowserLocation викликають відповідні методи для всіх дочірніх браузерів. Метод addBrowser додає новий браузе до списку дочірніх браузерів.



У цьому класі CompositeDemo створюються об'єкти різних браузерів (FirefoxBrowser, GoogleBrowser, InternetExplorerBrowser, OperaBrowser) і об'єкт BrowserIntegrator, який представляє інтегратор браузерів. До інтегратора додаються різні браузери за допомогою методу addBrowser. Потім викликаються методи printBrowserName та printBrowserLocation, які виводять назви та розташування всіх браузерів, включаючи дочірні браузери, що є частиною інтегратора.



Результат:



Висновок:

Лабораторна робота спрямована на вивчення та реалізацію шаблону проектування "Composite". У процесі вивчення цього шаблону були розглянуті основні концепції, такі як компоненти, листки і композити, що дозволяють створювати структури деревоподібних ієрархій.

У реалізації було показано, як створити абстрактний клас Component, який описує загальний інтерфейс для всіх елементів структури. Класи Leaf представляють одиничні об'єкти, а Composite дозволяють об'єднувати об'єкти в більш складні структури.

Основний висновок полягає в тому, що шаблон "Composite" дозволяє створювати ієрархії об'єктів так, що клієнт може взаємодіяти з одиничними об'єктами і композитами однаковим способом. Це полегшує роботу зі структурами, де одні об'єкти можуть містити інші, а клієнту не потрібно відмежовувати їхні типи під час використання.