**Породжувальні патерни проектування** (або просто **шаблони проектування**) - це повторювані рішення для типових проблем, які виникають під час розробки програмного забезпечення. Вони допомагають зробити код більш гнучким, зрозумілим та легким для підтримки.

**Існують такі породжувальні патерни проектування:**

* **Фабричний метод:** Визначає інтерфейс для створення об'єктів, але дозволяє підкласам вирішувати, який клас інстанціювати. Це корисно, коли потрібно створювати різні типи об'єктів на основі деяких умов.
* **Абстрактна фабрика:** Надає інтерфейс для створення сімейств пов'язаних або залежних об'єктів, не вдаючись до їх конкретних класів. Це зручно, коли потрібно створювати комплекти об'єктів, що належать до однієї категорії.
* **Будівельник:** Розділяє процес створення складного об'єкта на кілька кроків, дозволяючи отримувати різні представлення цього об'єкта. Це корисно, коли процес створення об'єкта складний і має багато варіацій.
* **Прототип:** Створює нові об'єкти шляхом копіювання існуючого об'єкта (прототипу). Це корисно, коли потрібно створювати багато схожих об'єктів.
* **Одинак:** Гарантує, що від класу існує лише один екземпляр. Це корисно, коли потрібен єдиний глобальний об'єкт для доступу до ресурсів або стану.

**Для чого використовують шаблони проектування:**

* **Збільшення гнучкості:** Дозволяють легко змінювати поведінку системи без зміни основного коду.
* **Покращення повторюваності:** Усувають необхідність повторно вирішувати ті самі проблеми.
* **Спрощення розуміння коду:** Роблять код більш структурованим і зрозумілим.
* **Зменшення кількості помилок:** Допомагають уникати типових помилок при розробці програмного забезпечення.

**Коли використовувати той чи інший шаблон:**

* **Фабричний метод:** Коли потрібно створювати об'єкти різних класів на основі деяких умов.
* **Абстрактна фабрика:** Коли потрібно створювати сімейства пов'язаних об'єктів.
* **Будівельник:** Коли процес створення об'єкта складний і має багато варіацій.
* **Прототип:** Коли потрібно швидко створювати багато схожих об'єктів.
* **Одинак:** Коли потрібен єдиний глобальний об'єкт.

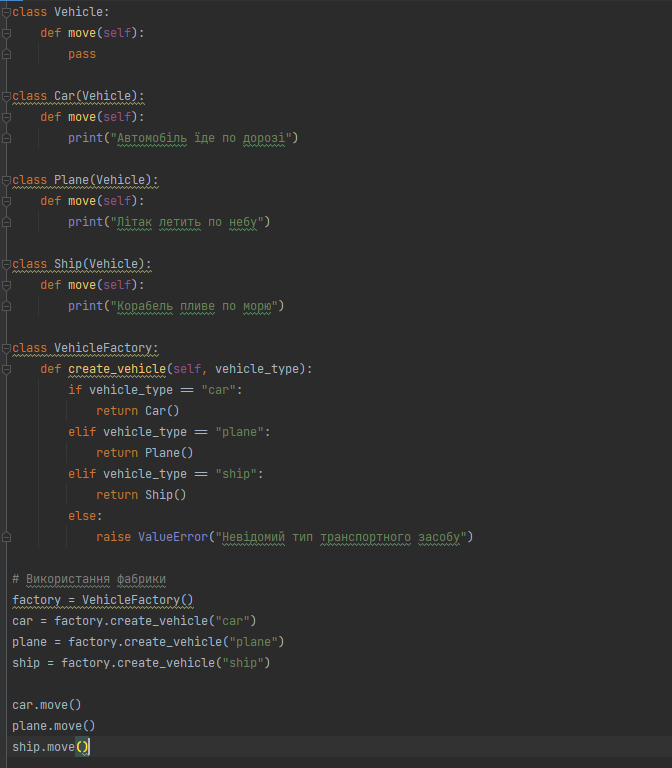


Рисунок 1-Код до завдання

* **Клас Vehicle:** Це базовий клас для всіх транспортних засобів.
* **Класи Car, Plane, Ship:** Конкретні реалізації різних транспортних засобів.
* **Клас VehicleFactory:**
  + Метод create\_vehicle приймає тип транспортного засобу.
  + В залежності від типу, створює відповідний об'єкт і повертає його.

**Переваги цього підходу:**

* **Гнучкість:** Легко додавати нові типи транспортних засобів без зміни існуючого коду.
* **Розширюваність:** Можна додати додаткові параметри для кожного типу транспортного засобу (наприклад, кількість пасажирів, швидкість).
* **Централізоване управління:** Вся логіка створення транспортних засобів зосереджена в одному місці.

**Можливі розширення:**

* **Додати підтримку гібридних транспортних засобів:** Електромобілі, гібриди.
* **Впровадити валідацію вхідних даних:** Перевіряти, чи коректні передані параметри.
* **Використати фабричний метод для створення різних моделей транспортних засобів:** Легковий автомобіль, вантажівка.

Цей приклад демонструє, як фабричний метод може бути використаний для створення об'єктів різного типу, що мають спільний базовий інтерфейс. Це особливо корисно в системах, де тип об'єкта визначається динамічно під час виконання програми.