Паттерны проектирования: Паттерн Фабричного метода в TypeScript

Освойте Паттерн Фабричного метода и создавайте объекты с легкостью.

Добро пожаловать в серию «Паттерны проектирования в TypeScript», в которой представлены некоторые полезные паттерны проектирования в веб-разработке с использованием TypeScript.

Предыдущие статьи следующие:

Паттерн стратегии в TypeScript

Паттерн цепочки ответственности в TypeScript

Паттерн наблюдателя в TypeScript

Паттерн метода паттерна в TypeScript

Паттерн адаптера в TypeScript

Паттерн фабричного метода в TypeScript

Абстрактный фабричный паттерн в TypeScript

Паттерны проектирования очень важны для веб-разработчиков, и мы можем писать лучший код, освоив их. В этой статье я буду использовать TypeScript, чтобы представить Паттерн Фабричного метода.

Паттерн Фабричного метода, также известный как Паттерн Фабрики, иногда еще называют полиморфным фабричным паттерном, который в свою очередь относится к порождающим паттернам.

В Паттерне Фабричного метода родительский класс фабрики отвечает за определение публичного интерфейса для создания продуктовых объектов, а подкласс фабрики отвечает за создание специфических продуктовых объектов. Цель этого состоит в том, чтобы оставить создание специфических свойство продукта подклассу фабрики. То есть именно в подклассе фабрики определить, какой конкретный класс продукта должен быть создан.

На картинке выше я смоделировал процесс покупки автомобиля пользователем. Bytefer и Chris1993 заказали модели SuperX01 и SuperX02 на фабриках SuperX01 и SuperX02 соответственно, а затем фабрика произвела соответствующие модели и доставила их пользователям после завершения производства.

Давайте посмотрим, как можно использовать Паттерн Фабрики для описания процесса производства конкретной модели автомобиля на автомобильном заводе.

Чтобы лучше понять последующий код, давайте сначала внимательно изучим соответствующую диаграмму UML:

Паттерн Фабричного метода включает следующие роли:

Продукт (Vehicle): абстрактный продукт

Конкретный продукт (SuperX01): конкретный продукт

Factory (VehicleFactory): абстрактная фабрика

ConcreteFactory(SuperX01Factory): конкретная фабрика

Затем мы определяем абстрактный класс Vehicle и два его подкласса SuperX01 и SuperX02 для конкретных двух различных типов транспортных средств.

Затем мы определяем класс VehicleFactory для представления завода по производству автомобилей. Абстрактный класс содержит абстрактный метод produceVehicle, который является так называемым фабричным методом.

На основе абстрактного класса VehicleFactory мы определяем фабричные классы SuperX01Factory и SuperX02Factory для производства моделей автомобилей SuperX01 и SuperX02:

После создания фабричных классов SuperX01Factory и SuperX02Factory мы можем начать производить автомобили:

Если запустить приведенный выше код, то в терминале вы увидите следующий результат:

Итак, давайте подведем итоги и определим сценарии использования Паттерна Фабричного метода:

В Паттерне Фабричного метода абстрактный фабричный класс должен только предоставить интерфейс для создания продуктов, а его подклассы определяют конкретные создаваемые объекты. Используя объектно-ориентированный полиморфизм и принцип подстановки Лисков, в ходе работы программы, объекты подкласса переопределяют объекты родительского класса, делая расширение системы более простой задачей.

Если у вас есть какие-либо вопросы, пожалуйста, оставьте мне сообщение. Позже я продолжу знакомить вас с другими паттернами. Если вам интересно, вы можете подписаться на меня в Medium или Twitter.