УДК 004.05

СРАВНЕНИЕ ФАБРИК. ФАБРИЧНЫЙ МЕТОД, АБСТРАКТНАЯ ФАБРИКА И ДРУГИЕ

И.В. МИСЕВИЧ
(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматриваются вариации порождающих шаблонов проектирования – фабрики, их особенности, отличия. Представлен краткий обзор шаблонов проектирования – фабрики. Приведены примеры реализации данных шаблонов.

Для упрощения проектирования и поддержки программ часто используют типовые унифицированные решения – паттерны. За удобное и безопасное создание новых объектов или даже целых семейств объектов отвечают порождающие паттерны, а именно фабрики. Во многих книгах и источниках определения «фабрик» дается авторами по-своему. Это создает большую путаницу при чтении различных материалов. В данной статье рассмотрим вариации фабрик, что бы понять разницу между ними.

Основной раздел

Фабрика – это общая концепция проектирования функций, методов и классов, когда какая-то одна часть программы отвечает за создание других частей программы.

В общей концепции имеется в виду:

- функция или метод, создающий все объекты программы;
- класс, создающий пользователей системы;
- статический метод, оборачивающий конструктор класса.

Итак, выделяют следующие фабрики:

- создающий метод;
- статический фабричный метод;
- простая фабрика;
- фабричный метод;
- абстрактная фабрика.

Рассмотрим каждую из них в подробности.

Создающий метод – это простой метод-обертка над вызовом конструктора продукта. Выделив создающий метод, изолируются любые изменения в конструировании продуктов от основного кода. Например, можно вовсе убрать вызов конструктора из создающего метода, отдавая вместо нового некий существующий объект.

Многие называют его фабричным методом, только потому, что он создает новые объекты. Ошибочное мнение: «этот метод создает объекты, а раз все фабрики создают что-то, значит этот метод – фабричный». И это вносит основную путаницу между понятием Создающего метода и паттерном Фабричный метод.

В этом примере метод next является создающим методом:

Листинг 1 – Пример создающего метода

```
class Number {
   private $value;

   public function __construct($value) {
      $this->value = $value;
   }

   public function next() {
      return new Number ($this->value + 1);
   }
}
```

Статический фабричный метод – вариация создающего метода, объявленная как static. Если этот метод создает объекты своего же класса, то, он выступает в роли альтернативного конструктора.

Это может быть полезно, если:

- требуется создать разные по функциональности конструкторы, у которых бы совпадали сигнатуры, например, Random(int max) и Random(int min). Это невозможно во многих языках программирования, но создав статический метод, можно обойти это ограничение;

- хочется повторно использовать готовые объекты, вместо создания новых, например, паттерн Одиночка. При вызове конструктора всегда создается новый объект. Это можно обойти, если вынести вызов конструктора в новый метод. В этом методе можно сначала поискать готовый объект в кеше, и только если его нет, создать новый объект.

В данном примере метод load является статическим фабричным методом – он предоставляет удобный способ загрузить пользователя из базы данных:

Листинг 2 – Пример статического фабричного метода

```
class User {
   private $name, $email, $phone;

   public function __construct($name, $email, $phone) {
      $this->name = $name;
      $this->email = $email;
      $this->phone = $phone;
   }

   public static function load($id) {
      list($name, $email, $phone) = DB::load_data('users', 'name', 'email', 'phone');
      $user = new User($name, $email, $phone);
      return $user;
   }
}
```

Паттерн Простая фабрика – это класс, в котором есть один метод с большим условным оператором, выбирающим создаваемый продукт. Этот метод вызывают с неким параметром, по которому определяется, какой из продуктов нужно создать. У простой фабрики, как правило, нет подклассов.

Обычно, простую фабрику путают с общим понятием Фабрики или с любым из фабричных паттернов. Однако если объявить класс простой фабрики абстрактным, это не делает его одним и тем же, что и абстрактная фабрика.

Пример простой фабрики представлен на листинге 3.

Листинг 3 – Пример простой фабрики

```
class UserFactory {
   public static function create($type) {
      switch ($type) {
         'user': return new User();
         'customer': return new Customer();
         'admin': return new Admin();
         default:
            throw new Exception('Wrong user type passed.'); }
   }
}
```

Паттерн Фабричный метод – это устройство классов, при котором подклассы могут переопределять тип создаваемого в суперклассе продукта. Если имеется иерархия продуктов и абстрактный создающий метод, который переопределяется в подклассах, то это паттерн Фабричный метод (листинг 4).

Листинг 4 – Пример фабричного метода

```
class Department {
   public abstract function createEmployee($id);

   public function fire($id) {
      $employee = $this->createEmployee($id);
      $employee->paySallary();
      $employee->dismiss();
   }
}

class ITDepartment extends Department {
   public function createEmployee($id) {
      return new Programmer($id);
   }
```

```
}

class AccountingDepartment extends Department {
   public function createEmployee($id) {
      return new Accountant($id);
   }
}
```

Паттерн Абстрактная фабрика – это устройство классов, облегчающее создание семейств продуктов.

Для примера рассмотрим классы Транспорт + Двигатель + Управление. Вариациями этого семейства могут стать:

Автомобиль + ДвигательВнутреннегоСгорания + Руль

Самолет + РеактивныйДвигатель + Штурвал

Если у вас нет семейств продуктов, значит, абстрактная фабрика не может быть применена.

Заключение

Представлен краткий обзор шаблонов проектирования – фабрики. Приведены примеры реализации данных шаблонов, что показывает отличие между ними. Эта статья может служить отправной точкой в изучении паттернов проектирования, а в частности разобраться в схожих по названию, но разных по содержанию паттернах.

ЛИТЕРАТУРА

1. CPP Rreference [Электронный ресурс] Паттерны проектирования (Design Patterns). – Режим доступа: http://cpp-reference.ru/patterns/. – Дата доступа: 25.09.2017.
2. Tproger [Электронный ресурс] Шаблоны проектирования. – Режим доступа: https://tproger.ru/translations/design-patterns-for-beginners/. – Дата доступа: 25.09.2017.
3. Technerium [Электронный ресурс] Фабрика – паттерн разработки. – Режим доступа: http://www.technerium.ru/izuchenie-java-na-praktike/fabrika-pattern-razrabotki. – Дата доступа: 26.09.2017.