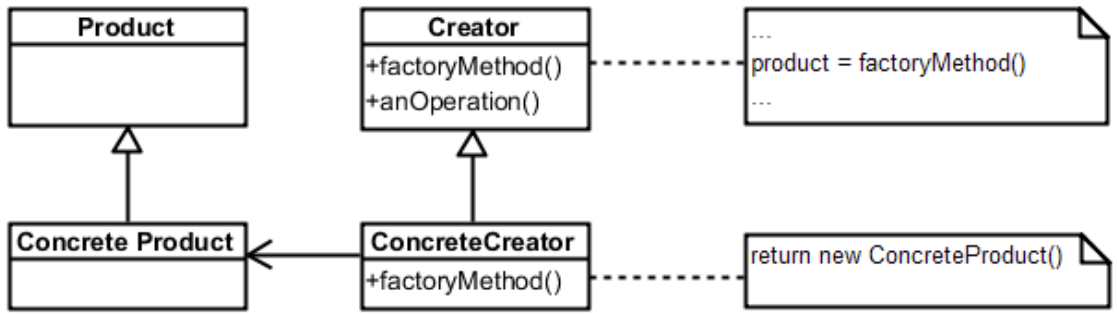
Порождающие и поведенческие паттерны, детали реализации

Юрий Литвинов
yurii.litvinov@gmail.com

26.04.2017г

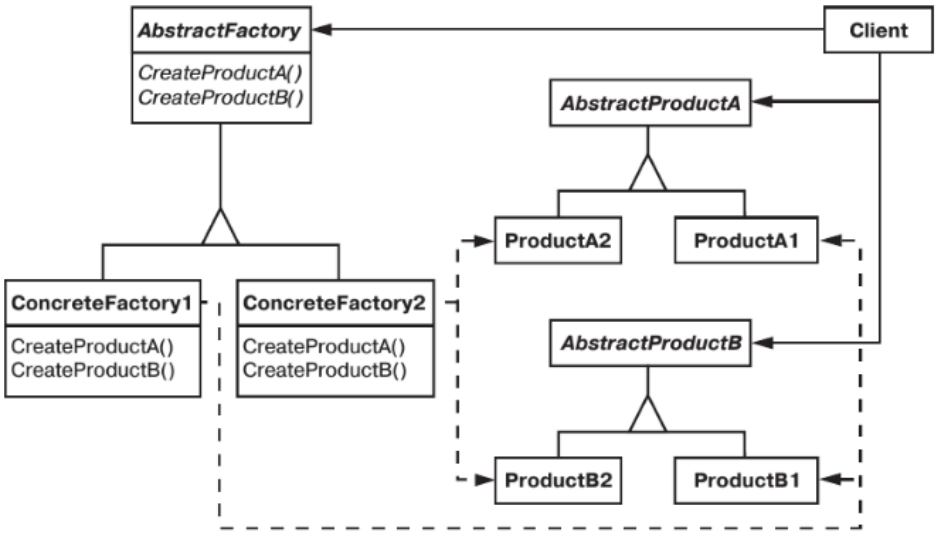
“Фабричный метод” (Factory Method), детали реализации



- Абстрактный Creator или реализация по умолчанию
  - Второй вариант может быть полезен для расширяемости
- Параметризованные фабричные методы
- Если язык поддерживает инстанциацию по прототипу (JavaScript, Smalltalk), можно хранить порождаемый объект
- Creator не может вызывать фабричный метод в конструкторе
- Можно сделать шаблонный Creator

“Абстрактная фабрика” (Abstract Factory), детали реализации

- Хорошо комбинируются с паттерном “Одиночка”
- Если семейств продуктов много, то фабрика может инициализироваться *прототипами*, тогда не надо создавать сотню подклассов



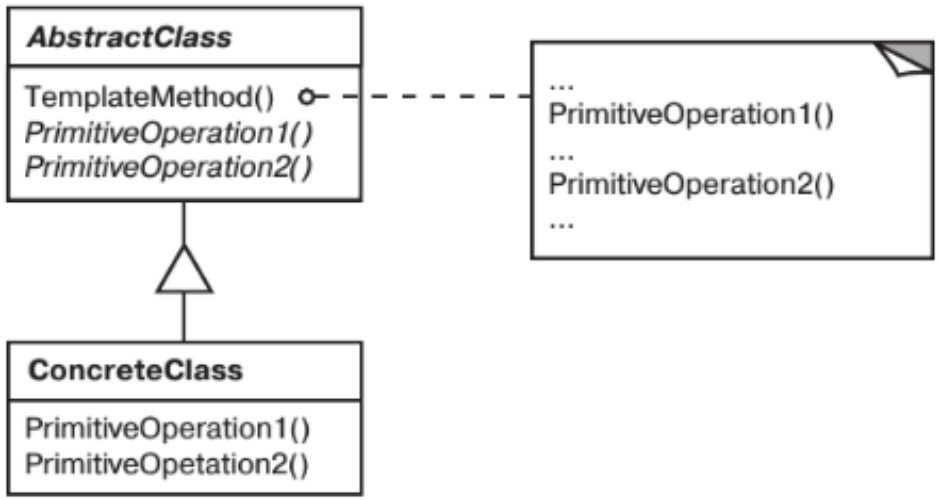
- Прототип на самом деле может быть классом (например, Class в Java)
- Если виды объектов часто меняются, может помочь параметризация метода создания
  - Может пострадать типобезопасность

“Прототип” (Prototype), детали реализации

- Паттерн интересен только для языков, где мало runtime-информации о типе (C++)
- Реестр прототипов, обычно ассоциативное хранилище
- Операция Clone
  - Глубокое и мелкое копирование
  - В случае, если могут быть круговые ссылки
  - Сериализовать/десериализовать объект (но помнить про идентичность)
- Инициализация клона
  - Передавать параметры в Clone — плохая идея

“Строитель” (Builder), детали реализации

“Шаблонный метод” (Template Method), детали реализации



- Сам шаблонный метод, как правило, невиртуальный
- Примитивные операции могут быть виртуальными или чисто виртуальными
  - Лучше их делать protected
  - Чем их меньше, тем лучше
- Лучше использовать соглашения об именовании, например, называть операции с Do

“Посредник” (Mediator), детали реализации

“Команда” (Command), детали реализации

“Цепочка ответственности” (Chain of Responsibility), детали реализации

“Наблюдатель” (Observer), детали реализации

“Состояние” (State), детали реализации

“Посетитель” (Visitor), детали реализации

“Хранитель” (Memento), детали реализации

“Интерпретатор” (Interpreter), детали реализации

“Итератор” (Iterator), детали реализации