1.Какие типы паттернов бывают?

**Порождающие паттерны** — это паттерны, которые абстрагируют процесс порождения классов и объектов. (**Абстрактная фабрика (Abstract Factory)**, **Строитель (Builder)**, **Фабричный метод (Factory Method)**, **Прототип (Prototype)**, **Одиночка (Singleton))**

Другая группа паттернов - **структурные паттерны** - рассматривает, как классы и объекты образуют более крупные структуры - более сложные по характеру классы и объекты(**Адаптер (Adapter)**, **Мост (Bridge)**, **Компоновщик (Composite)**, **Декоратор (Decorator)**, **Фасад (Facade)**, **Приспособленец (Flyweight)**, **Заместитель (Proxy))**

Третья группа паттернов называются **поведенческими** - они определяют алгоритмы и взаимодействие между классами и объектами, то есть их поведение. (**Цепочка обязанностей (Chain of responsibility),Команда (Command)**, **Интерпретатор (Interpreter)**, **Итератор (Iterator)**, **Посредник (Mediator)**, **Хранитель (Memento)**, **Наблюдатель (Observer)**, **Состояние (State)**, **Стратегия (Strategy)**, **Шаблонный метод (Template method)**, **Посетитель (Visitor).**

2. Нарисуете диаграмму классов и поясните принцип работы паттерна Singleton. Назовите условия применения.

**Одиночка** (Singleton, Синглтон) - порождающий паттерн, который гарантирует, что для определенного класса будет создан только один объект, а также предоставит к этому объекту точку доступа.

Синглтон позволяет создать объект только при его необходимости. Если объект не нужен, то он не будет создан. В этом отличие синглтона от глобальных переменных.

В классе определяется статическая переменная - ссылка на конкретный экземпляр данного объекта и приватный конструктор. В статическом методе getInstance() этот конструктор вызывается для создания объекта, если, конечно, объект отсутствует и равен null.

3. Как сделать потокобезопасную, с отложенной реализацией версию Singleton.

public sealed class SingletonTwo

{ private static SingletonTwo \_instance;   
private static readonly object Locker = new object();   
private SingletonTwo() { }   
public static SingletonTwo GetInstance()   
{ if (\_instance == null)   
{ lock (Locker)  
{   
if (\_instance == null)   
{   
\_instance = new SingletonTwo();   
}   
}  
}   
return \_instance;   
}   
}

///////////////

public sealed class SingletonThree {

private static readonly SingletonThree Instance = new SingletonThree();

static SingletonThree() { }

private SingletonThree() { }

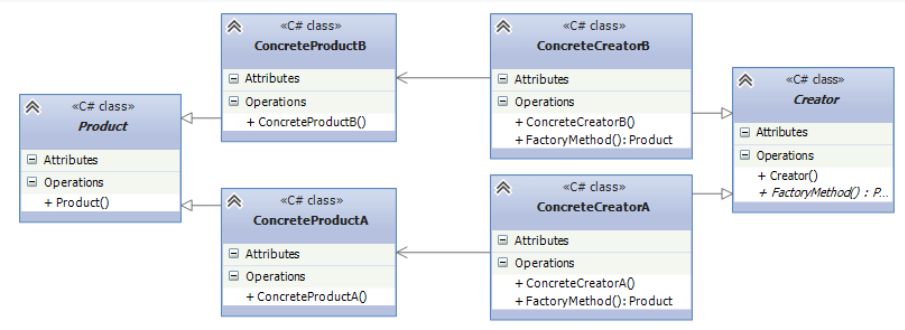
public static SingletonThree GetInstance()   
{ return Instance; } }

4. Нарисуете диаграмму классов и поясните принцип работы паттерна Factory Method. Назовите условия применения

Фабричный метод (Factory Method) — это паттерн, который определяет интерфейс для создания объектов некоторого класса, но непосредственное решение о том, объект какого класса создавать происходит в подклассах. То есть паттерн предполагает, что базовый класс делегирует создание объектов классам-наследникам.

**Когда надо применять паттерн**

* Когда заранее неизвестно, объекты каких типов необходимо создавать
* Когда система должна быть независимой от процесса создания новых объектов и расширяемой: в нее можно легко вводить новые классы, объекты которых система должна создавать.
* Когда создание новых объектов необходимо делегировать из базового класса классам наследникам



**Участники**

* Абстрактный класс **Product** определяет интерфейс класса, объекты которого надо создавать.
* Конкретные классы **ConcreteProductA** и **ConcreteProductB** представляют реализацию класса Product. Таких классов может быть множество
* Абстрактный класс **Creator** определяет абстрактный фабричный метод FactoryMethod(), который возвращает объект Product.
* Конкретные классы **ConcreteCreatorA** и **ConcreteCreatorB** - наследники класса Creator, определяющие свою реализацию метода FactoryMethod(). Причем метод FactoryMethod() каждого отдельного класса-создателя возвращает определенный конкретный тип продукта. Для каждого конкретного класса продукта определяется свой конкретный класс создателя.

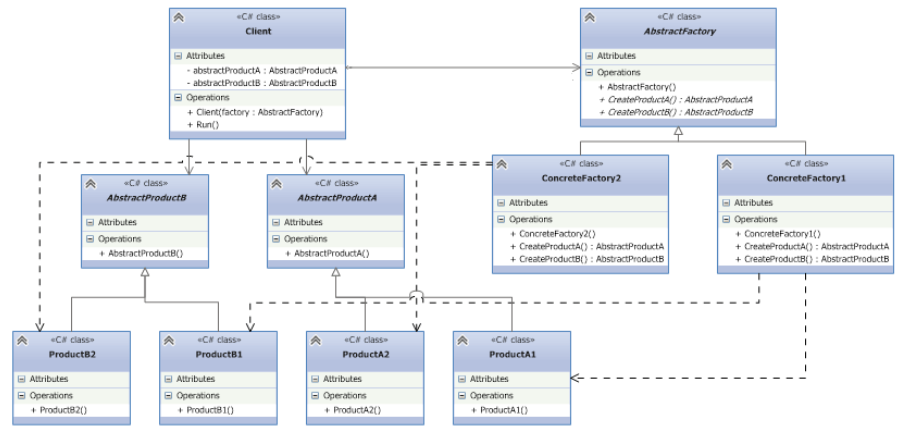
Таким образом, класс Creator делегирует создание объекта Product своим наследникам. А классы ConcreteCreatorA и ConcreteCreatorB могут самостоятельно выбирать какой конкретный тип продукта им создавать.

5. Нарисуете диаграмму классов и поясните принцип работы паттерна Abstract Factory. Назовите условия применения

Паттерн "Абстрактная фабрика" (Abstract Factory) предоставляет интерфейс для создания семейств взаимосвязанных объектов с определенными интерфейсами без указания конкретных типов данных объектов.

**Когда использовать абстрактную фабрику**

* Когда система не должна зависеть от способа создания и компоновки новых объектов
* Когда создаваемые объекты должны использоваться вместе и являются взаимосвязанными

****

Паттерн определяет следующих участников:

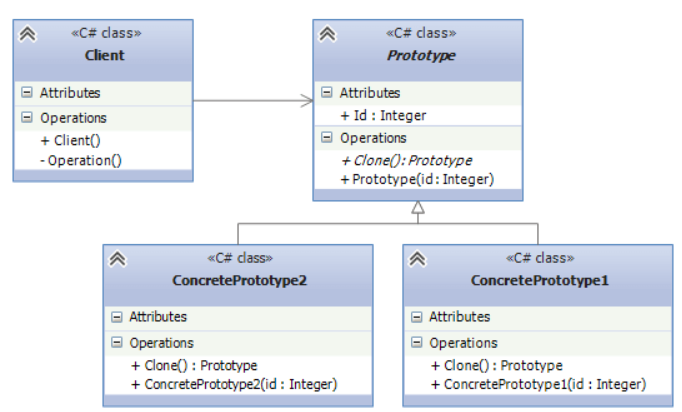
* Абстрактные классы **AbstractProductA** и **AbstractProductB** определяют интерфейс для классов, объекты которых будут создаваться в программе.
* Конкретные классы **ProductA1 / ProductA2** и **ProductB1 / ProductB2** представляют конкретную реализацию абстрактных классов
* Абстрактный класс фабрики **AbstractFactory** определяет методы для создания объектов. Причем методы возвращают абстрактные продукты, а не их конкретные реализации.
* Конкретные классы фабрик **ConcreteFactory1** и **ConcreteFactory2** реализуют абстрактные методы базового класса и непосредственно определяют какие конкретные продукты использовать
* Класс клиента **Client** использует класс фабрики для создания объектов. При этом он использует исключительно абстрактный класс фабрики AbstractFactory и абстрактные классы продуктов AbstractProductA и AbstractProductB и никак не зависит от их конкретных реализаций

6. Нарисуете диаграмму классов и поясните принцип работы паттерна Prototype. Назовите условия применения

Паттерн Прототип (Prototype) позволяет создавать объекты на основе уже ранее созданных объектов-прототипов. То есть, по сути, данный паттерн предлагает технику клонирования объектов.

Когда использовать Прототип?

* Когда конкретный тип создаваемого объекта должен определяться динамически во время выполнения
* Когда нежелательно создание отдельной иерархии классов фабрик для создания объектов-продуктов из параллельной иерархии классов (как это делается, например, при использовании паттерна Абстрактная фабрика)
* Когда клонирование объекта является более предпочтительным вариантом нежели его создание и инициализация с помощью конструктора. Особенно когда известно, что объект может принимать небольшое ограниченное число возможных состояний.



**Участники**

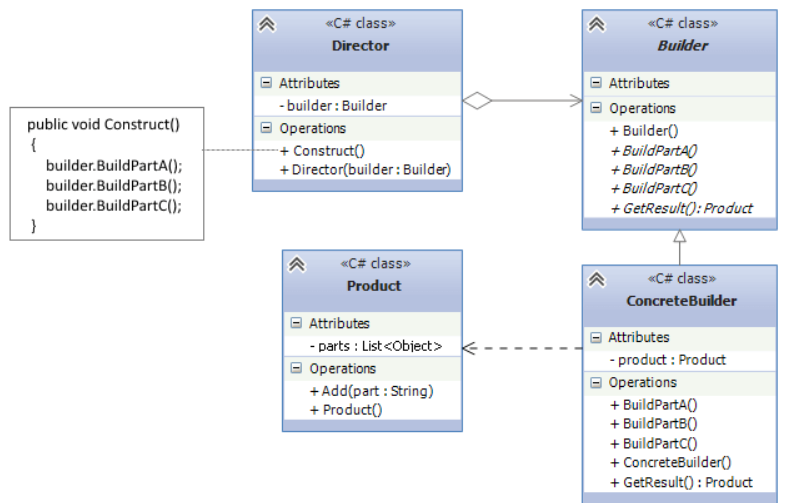
* **Prototype**: определяет интерфейс для клонирования самого себя, который, как правило, представляет метод Clone()
* **ConcretePrototype1** и **ConcretePrototype2**: конкретные реализации прототипа. Реализуют метод Clone()
* **Client**: создает объекты прототипов с помощью метода Clone()

7. Нарисуете диаграмму классов и поясните принцип работы паттерна Builder. Назовите условия применения

Строитель (Builder) - шаблон проектирования, который инкапсулирует создание объекта и позволяет разделить его на различные этапы.

**Когда использовать паттерн Строитель?**

* Когда процесс создания нового объекта не должен зависеть от того, из каких частей этот объект состоит и как эти части связаны между собой
* Когда необходимо обеспечить получение различных вариаций объекта в процессе его создания



**Участники**

* **Product**: представляет объект, который должен быть создан. В данном случае все части объекта заключены в списке parts.
* **Builder**: определяет интерфейс для создания различных частей объекта Product
* **ConcreteBuilder**: конкретная реализация Buildera. Создает объект Product и определяет интерфейс для доступа к нему
* **Director**: распорядитель - создает объект, используя объекты Builder

8. Что такое Пул объектов (Object pool), в чем суть паттерна?

[Пул объектов](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D0%BF%D1%83%D0%BB) — порождающий шаблон проектирования, набор инициализированных и готовых к использованию объектов.

Основная идея паттерна Object Pool состоит в том, чтобы избежать создания новых экземпляров класса в случае возможности их повторного использования.

9. Какое основное назначение паттерна отложенная инициализация (Lazy initialization)?

**Отложенная** **инициализация** объекта означает, что его создание откладывается до первого использования. (В этом разделе термины **отложенная** **инициализация** и **отложенное** создание экземпляра являются синонимами.) **Отложенная** **инициализация** в **основном** используется, чтобы повысить быстродействие, избежать ресурсоемких вычислений и уменьшить требования к памяти программы.