1. Какие типы паттернов бывают?

Виды шаблонов проектирования

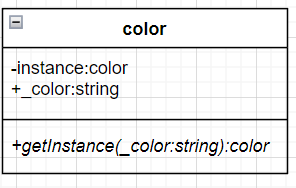
* [Порождающие](https://habrahabr.ru/company/mailru/blog/325492/#porozhdayuschie-shablony-proektirovaniya)
* [Структурные](https://habrahabr.ru/company/mailru/blog/325492/#strukturnye-shablony-proektirovaniya)
* [Поведенческие](https://habrahabr.ru/company/mailru/blog/325492/#povedencheskie-shablony-proektirovaniya)

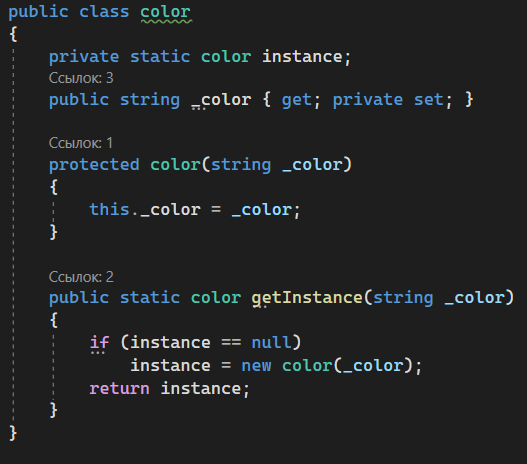
В программной инженерии порождающими называют шаблоны, которые используют механизмы создания объектов, чтобы создавать объекты подходящим для данной ситуации способом. Базовый способ создания может привести к проблемам в архитектуре или к её усложнению. Порождающие шаблоны пытаются решать эти проблемы, управляя способом создания объектов.

Структурными называют шаблоны, которые облегчают проектирование, определяя простой способ реализации взаимоотношений между сущностями. Эти шаблоны в основном посвящены компоновке объектов (object composition). То есть тому, как сущности могут друг друга использовать. Ещё одно объяснение: структурные шаблоны помогают ответить на вопрос «Как построить программный компонент?»

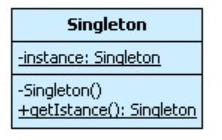
Поведенческие шаблоны проектирования определяют алгоритмы и способы реализации взаимодействия различных объектов и классов. Они обеспечивают гибкость взаимодействия между объектами. Они связаны с присвоением обязанностей (responsibilities) объектам. От структурных шаблонов они отличаются тем, что не просто описывают структуру, но и очерчивают шаблоны передачи данных, обеспечения взаимодействия. То есть поведенческие шаблоны позволяют ответить на вопрос «Как реализовать поведение в программном компоненте?»

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

**\*Пример из лабораторной работы:**



\*В общем случае:

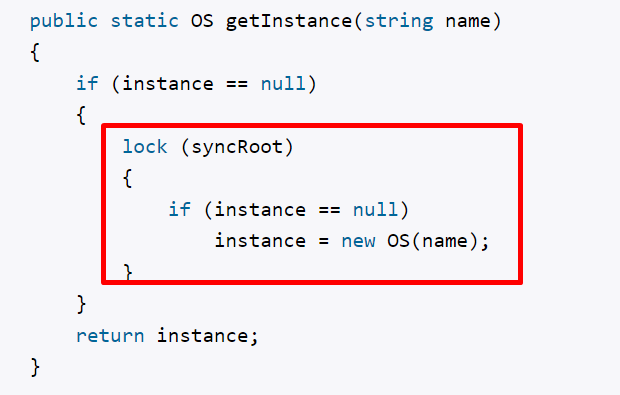


Условия применения: создавать единственный экземпляр некоторого типа, предоставлять к нему доступ извне и запрещать создание нескольких экземпляров того же типа.

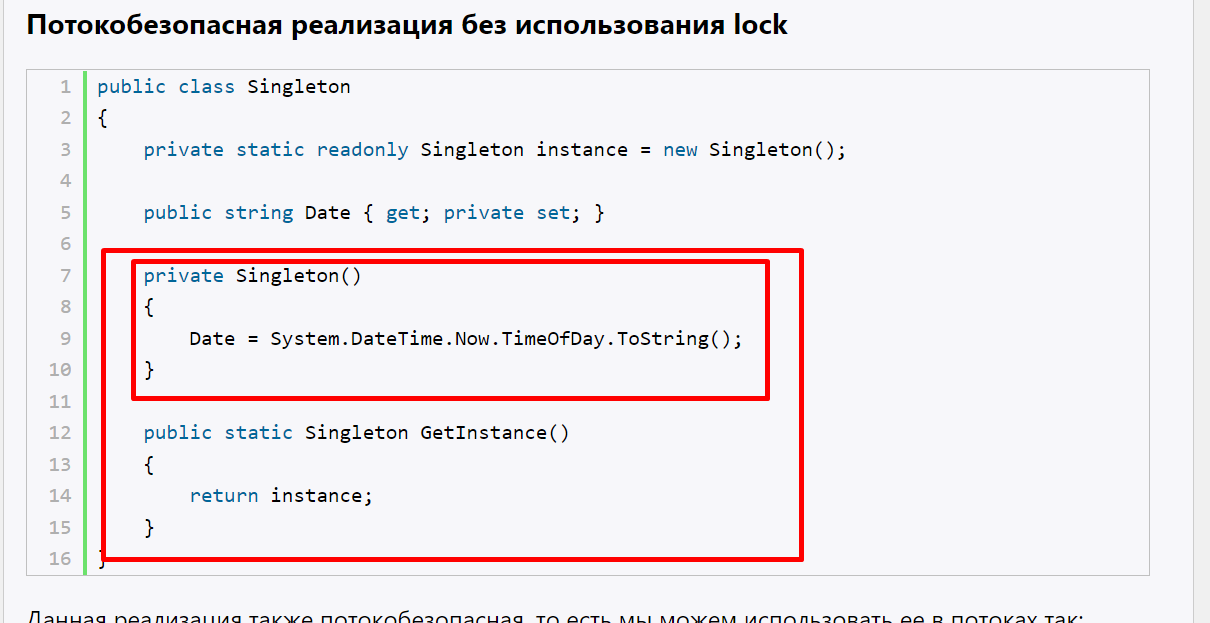
3. Как сделать потокобезопасную, с отложенной реализацией версию Singleton.

При применении паттерна синглтон в многопоточным программах мы можем столкнуться с проблемой доступа с нашему синглтону одновременно с разных потоков, это не позволяет предсказать конечный результат выполнения программы и теряется весь смысл использования данного паттерна.

Чтобы избежать одновременного доступа к коду из разных потоков критическая секция заключается в блок **lock**.

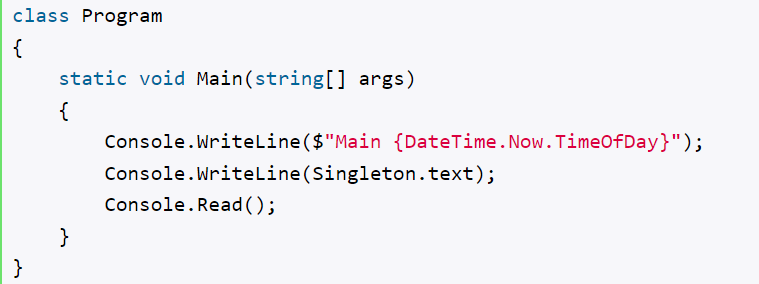


ИЛИ



Определение объекта синглтона в виде статического поля класса открывает нам дорогу к созданию **Lazy-реализации** паттерна Синглтон, то есть такой реализации, где данные будут инициализироваться только перед непосредственным использованием.





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

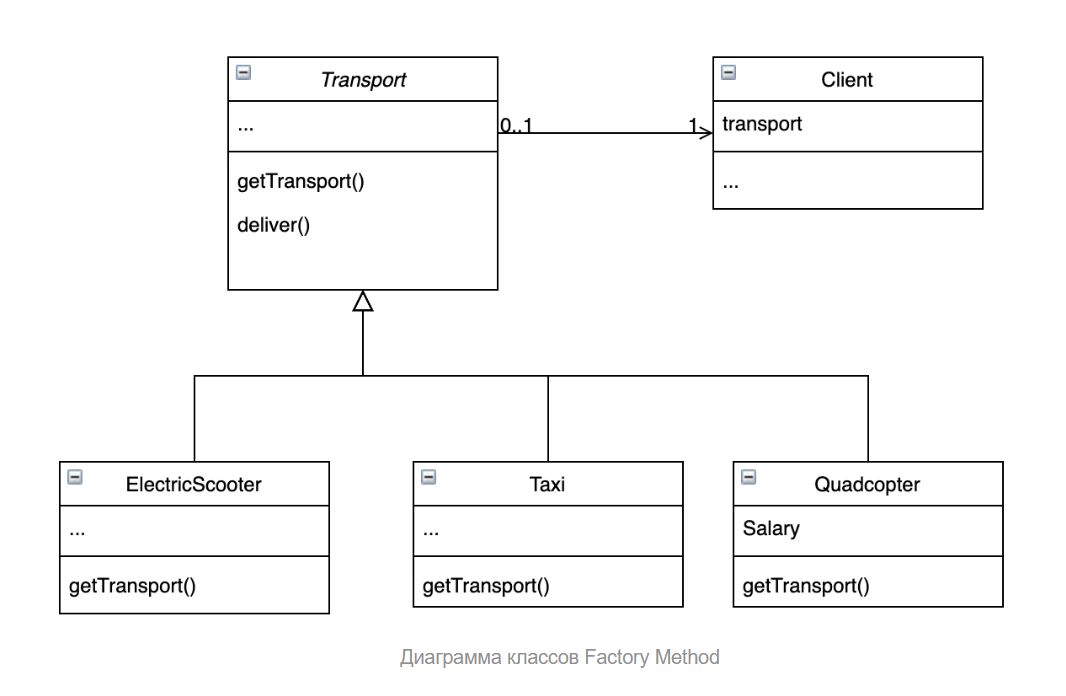
**Фабричный метод (Factory method)** также известный как Виртуальный конструктор (Virtual Constructor) - пораждающий шаблон проектирования, определяющий общий интерфейс создания объектов в родительском классе и позволяющий изменять создаваемые объекты в дочерних классах.

[Фабричный метод](https://www.udemy.com/course/design-patterns-cj/?referralCode=2F11FCD4981B4F368F2F)предлагает создавать объекты транспорта через вызов специального метода. Подклассы класса, который содержит фабричный метод могут изменять создаваемые объекты конкретных создаваемых транспортов. На первый взгляд, это может показаться бессмысленным: мы просто переместили вызов конструктора из одного конца программы в другой. Но теперь вы сможете переопределять фабричный метод в подклассе, чтобы изменить тип создаваемого транспорта.

Чтобы такая система заработала, все возвращаемые объекты имеют общий интерфейс, а подклассы могут производить объекты различных классов, имеющих общий интерфейс.

Для клиента фабричного метода нет разницы между создаваемыми объектами, так как он трактует их как некий абстрактный Транспорт. Для него важно, чтобы данный объект мог доставить еду из пункта А в пункта В, а как конкретно он это будет делать, неважно.

Посмотрим на диаграмму классов такого подхода.



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 Factory Method

-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

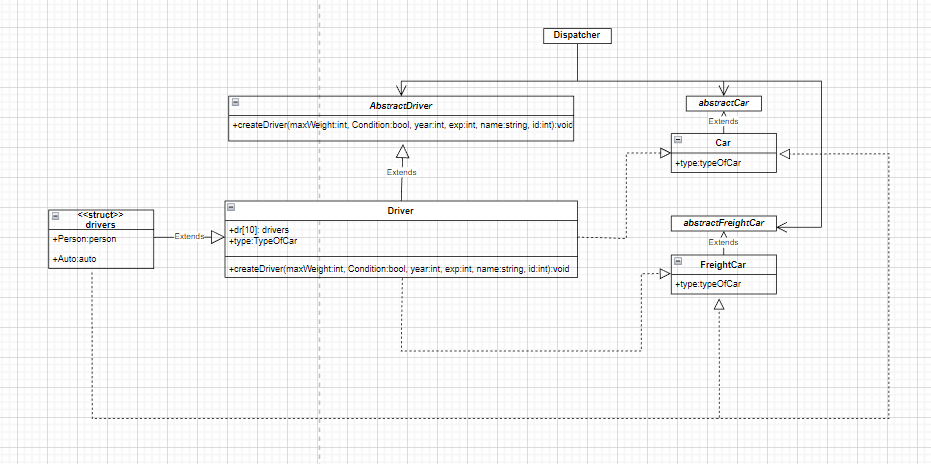
5. Нарисуете диаграмму классов и поясните принцип работы паттерна Abstract Factory. Назовите условия применения

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

Самый первый вопрос, на который нужно ответить самому себе, изучая данный паттерн: «Что же такое Абстрактная фабрика». Самый простой и точный ответ, гласит, что Абстрактная фабрика – это «фабрика фабрик». Но здесь появляется второй вопрос: «Для чего вообще может кому-нибудь понадобиться «фабрика фабрик»? Чтобы на него ответить рассмотрим пример из реальной жизни.  
  
Допустим, вы решили полностью взять под свой контроль рынок автомобилей. Как это сделать? Вы можете создать свою марку автомобиля, своё производство, провести масштабную рекламную компанию и т.д. Но, в этом случае вам придётся сражаться с такими гигантами авторынка, как Toyota или Ford. Не факт, что из данной борьбы вы выйдите победителем. Гораздо лучшим решением будет скупить заводы всех этих компаний, продолжить выпускать автомобили под их собственными марками, а прибыль класть себе в карман. Если я не ошибаюсь, такая структура в экономике называется – холдинг. Вот этот холдинг и будет Абстрактной фабрикой или «фабрикой фабрик». В нашей программе Абстрактная фабрика (холдинг) будет представлена интерфейсом или абстрактным классом. Предприятия, входящие в холдинг, представлены классами, реализующими данный интерфейс.



**Для лабораторной работы:**



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**

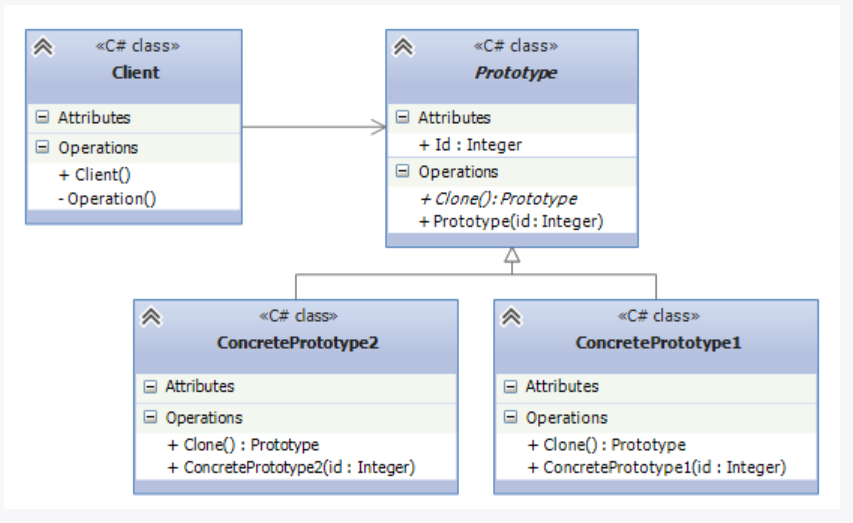
**Когда создаваемые объекты должны использоваться вместе и являются взаимосвязанными**

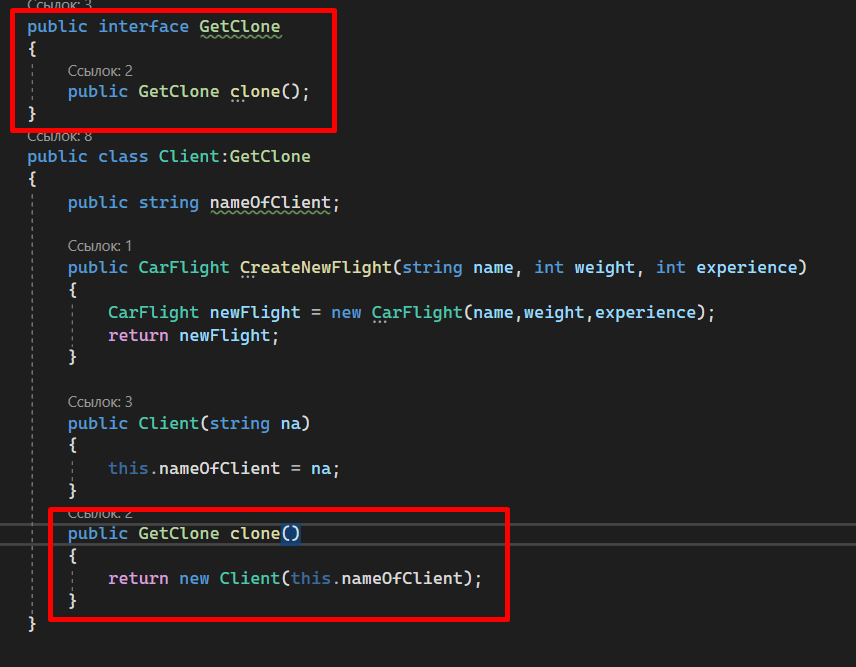
6. Нарисуете диаграмму классов и поясните принцип работы паттерна Prototype. Назовите условия применения

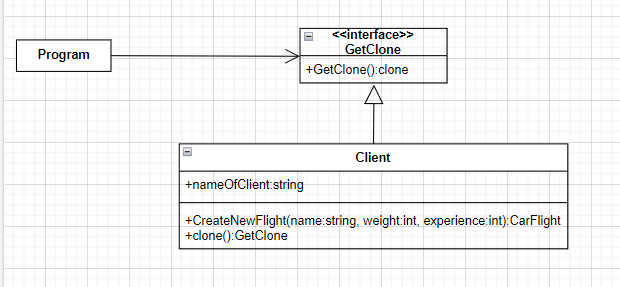
Паттерн Прототип (Prototype) позволяет создавать объекты на основе уже ранее созданных объектов-прототипов. То есть по сути данный паттерн предлагает технику клонирования объектов.

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.



**Для лабораторной работы:**

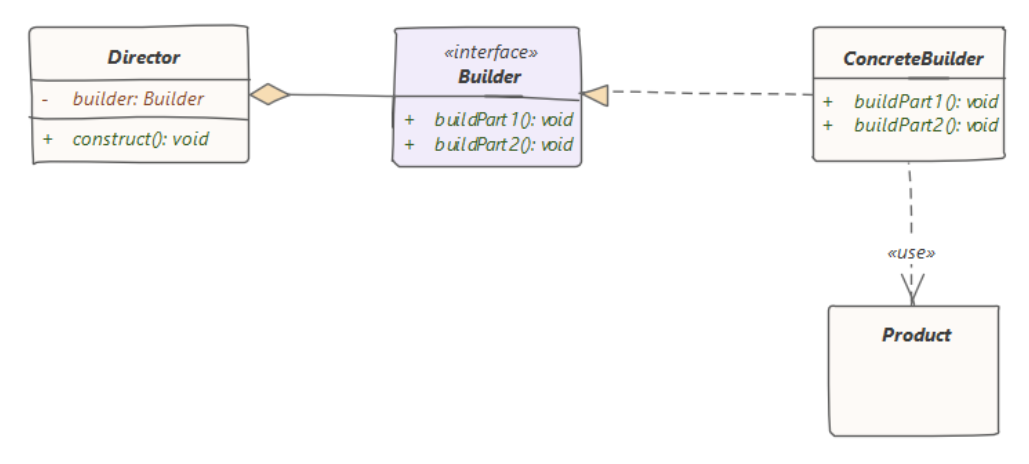


7. Нарисуете диаграмму классов и поясните принцип работы паттерна Builder. Назовите условия применения

### Паттерн проектирования Builder

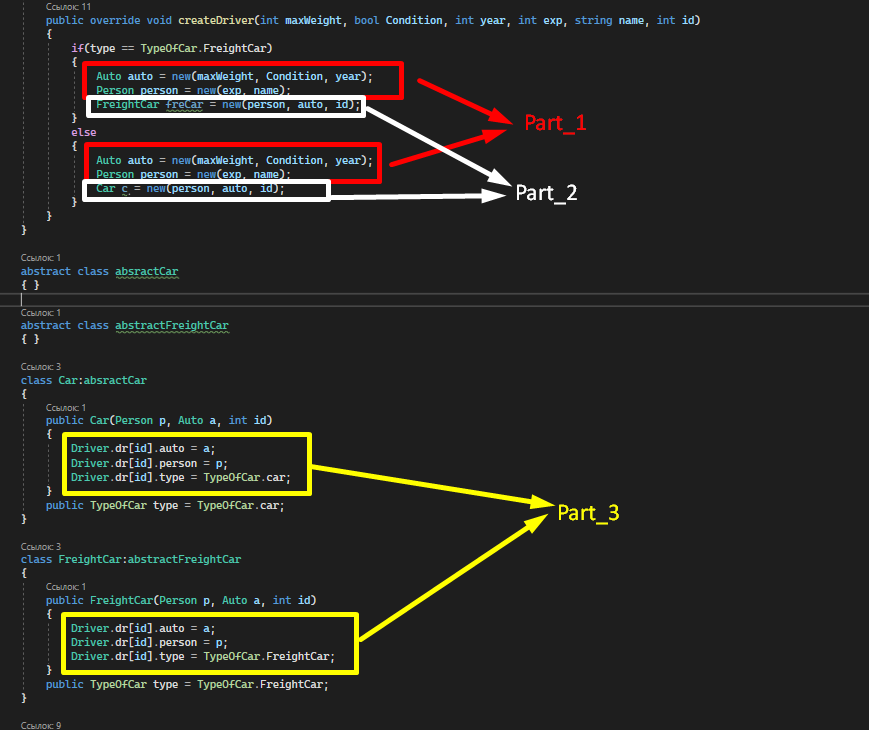
* Паттерн проектирования Builder разработан для обеспечения гибкого решения различных задач создания объектов в объектно-ориентированном программировании.
* Паттерн проектирования Builder позволяет отделить построение сложного объекта от его представления.
* Паттерн Builder создает сложные объекты, используя простые объекты и поэтапный подход.
* Паттерн предоставляет один из лучших способов создания сложных объектов.
* Это один из [паттернов проектирования банды четырех (GoF),](https://en.wikipedia.org/wiki/Design_Patterns) которые описывают, как решать периодически возникающие задачи проектирования в объектно-ориентированном программном обеспечении.
* Этот паттерн полезен для создания разных иммутабельных объектов с помощью одного и того же процесса построения объекта.

Паттерн Builder — это паттерн проектирования, который позволяет поэтапно создавать сложные объекты с помощью четко определенной последовательности действий. Строительство контролируется объектом-распорядителем (director), которому нужно знать только тип создаваемого объекта.



**Для лабораторной работы:**

В 5 вопросе UML-диаграмма для этого класса



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.

Пул объектов предназначен хранения готовых к использованию объектов. Когда системе требуется новый объект, он запрашивается из Пула, минуя процесс порождения. А после использования возвращается обратно в Пул вместо уничтожения.

Шаблон применяется для повышения производительности, если:

* объекты часто создаются и уничтожаются;
* в системе существует ограниченное количество объектов типа, хранимого в Пуле;
* создание и/или уничтожение объекта являются очень затратными операциями.

Пул объектов может работать как с интерфейсами, так и с конкретными реализациями. Все зависит от архитектуры разрабатываемой системы и решаемых задач.

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

Отложенная инициализация или «ленивая» инициализация — это способ доступа к объекту, скрывающий за собой механизм, позволяющий отложить создание этого объекта до момента первого обращения. Необходимость ленивой инициализации может возникнуть по разным причинам: начиная от желания снизить нагрузку при старте приложения и заканчивая оптимизацией редко используемого функционала. И действительно, не все функции приложения используются всегда и, тем более, сразу, потому создание объектов, реализующих их, вполне рационально отложить до лучших времён.