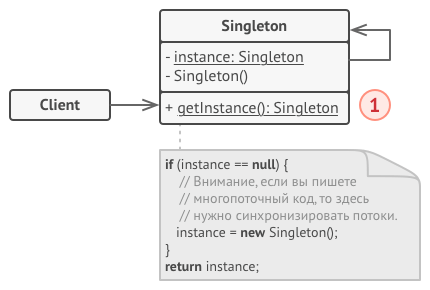
**ОДИНОЧКА** - гарантирует, что класс имеет только 1 экземпляр, и предоставляет глобальную точку доступа к этому экземпляру. Можно сказать, что класс нарушает "принцип одной обязанности" и имеет их две: свою основную обязанность и обязанность по созданию и предоставлению себя. Поэтому некоторые разработчики выносят функционал Одиночки в отдельную абстракцию.

ПРИМЕНЕНИЕ:

- когда должен быть единственный экземпляр какого-то класса, доступный всем клиентам. Например, общий доступ к базе данных из разных частей программы

- когда хочется иметь больше контроля над глобальными переменными

РЕАЛИЗАЦИЯ:

**1. КОНКРЕТНЫЙ СИНГЛТОН**: определяет статический метод getInstance(), который возвращает единственный экземпляр своего класса. Конструктор должен быть скрыт от клиентов - вызов метода getInstance() должен стать единственным способом получить объект этого класса. Переменная экземпляра. (Опционально) другой код класса. Например, класс подключения к базе данных.

ОДИНОЧКА VS ГЛОБАЛЬНАЯ ПЕРЕМЕННАЯ

Оба предоставляют глобальный доступ к экземпляру класса, но О также гарантирует существование одного экземпляра. О предоставляет возможность отложенной инициализации. Глобальные переменные засоряют пространство имен множеством глобальных ссылок на мелкие объекты.

ПРОБЛЕМЫ МНОГОПОТОЧНОСТИ

Если 2 потока обратятся к методу getInstance(), то есть вероятность, что для обоих еще не будет существовать экземпляра, поэтому будет создано 2 экземпляра. Решение:

1 - статическая инициализация переменной: лишает возможности отложенной инициализации, но и нет необходимости делать в методе проверку на существование экземпляра

2 - синхронизированный метод: весь метод становится synchronized. Работа метода замедляется в 100 раз. Не подходит для частого использования.

3 - синхронизированный блок в методе в месте создания экземпляра: синхронизирован только тот код, который производит создание экземпляра. Переменная должна быть volatile. Гарантирует, что параллельные потоки будут правильно работать с переменной при ее инициализации экземпляром. Не работает ниже версии 1.5

ПЛЮСЫ:

- гарантирует наличие единственного экземпляра класса

- предоставляет к нему глобальную точку доступа

- реализует отложенную инициализацию объекта-одиночки

МИНУСЫ:

- нарушает принцип единственной ответственности класса

- маскирует плохой дизайн

- проблемы мультипоточности

- требует постоянного создания Mock-объектов при юнит-тестировании