Паттерн Одиночка (Singleton)

Одиночка — это порождающий паттерн проектирования, который гарантирует, что у класса есть только **один экземпляр**, и предоставляет к нему **глобальную точку доступа**.

Применяется когда

1. В программе должен быть единственный экземпляр какого-то класса, доступный всем клиентам, например, общий доступ к базе данных из разных частей программы.

Одиночка скрывает от клиентов все способы создания нового объекта, кроме ***специального метода***. Этот метод либо ***создаёт объект***, либо ***отдаёт существующий объект***, если он уже был создан.

2. Необходимо иметь больше контроля над глобальными переменными.

В отличие от глобальных переменных, паттерн Одиночка гарантирует, что никакой другой код не заменит созданный экземпляр класса, поэтому вы всегда уверены в наличии лишь одного объекта-одиночки.

Паттерн Одиночка (Singleton)

Шаги реализации

1. Добавьте в класс приватное статическое поле, которое будет содержать одиночный объект.

2. Объявите статический создающий метод, который будет использоваться для получения одиночки.

3. Добавьте «ленивую инициализацию» (создание объекта при первом вызове метода) в создающий метод одиночки.

4. Сделайте конструктор класса приватным.

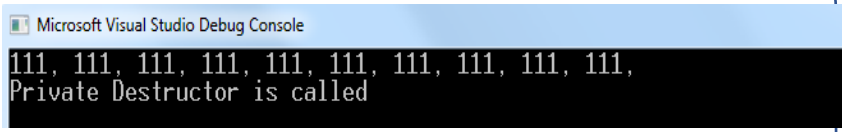
Паттерн Одиночка (Singleton) - Пример

```
class Singleton
{
public:
static Singleton& Instance()
{
// согласно стандарту, этот код ленивый и потокобезопасный
static Singleton s;
return s;
}
private:
Singleton() { }  // конструктор недоступен
~Singleton() {

} // и деструктор
// необходимо также запретить копирование
Singleton(Singleton const&); // реализация не нужна
Singleton& operator= (Singleton const&);  // и тут
};

int main(int argc, char** argv) {
//new Singleton(); // Won't work
Singleton& instance = Singleton::Instance();
return 0;
}
```

Паттерн Одиночка (Singleton)

```
class Singleton
{public:
 static Singleton& Instance()
 { static Singleton s;
   return s;
 }
 void SomeInformation()
 {
  for (int i = 0; i < 10; i++) std::cout << mas[i] << ", ";
  std::cout <<std::endl;
 }
 private:
  Singleton() {
  for (int i = 0; i < 10; i++){mas[i] = 111; }
 };
 ~Singleton() { std::cout << "Private Destructor is
called"<<std::endl; };
 Singleton(Singleton const&); // реализация не нужна
 Singleton& operator= (Singleton const&);
 protected:
 int mas[10];};
```

Паттерн Одиночка (Singleton)




```
int main()
{
        //new Singleton(); // Won't work
        Singleton& instance = Singleton::Instance();
        Singleton::Instance().SomeInformation();
}
```