Паттерн Одиночка

Наумов Д.А., доц. каф. КТ

Основы программной инженерии, 2019


Говорю тебе, другой такой НЕТ В ПРИРОДЕ. Посмотри на эти линии, на эти изгибы!
Ты с кем говоришь — со мной или с машиной? И когда отдашь мое кухонное полотенце?

Маленькое упражнение в стиле сократовских диалогов

Как создать один объект?

Маленькое упражнение в стиле сократовских диалогов

Как создать один объект?

```
MyObject := MyClass.Create;
```

А если другой объект захочет создать еще один экземпляр `MyClass`? Сможет ли он снова вызвать `MyClass.Create`;

Значит, если у нас есть класс, мы всегда можем создать один или несколько экземпляров этого класса?

Хмм, интересно.

А вы знаете, что можно сделать так?

```
type
  MyClass = class
    private
      constructor Create;
  end;
```

И что это значит?

Хоть КАКОЙ-НИБУДЬ объект может вызывать приватный конструктор?

Хмм, он может вызываться только из кода MyClass. Но какая от этого польза?

Почему?

Потому что для вызова нужно иметь экземпляр класса, а я не могу создать экземпляр, потому что он не может быть создан другим классом. Классическая проблема «курицы и яйца».

Понятно.

А что можно сказать об этом фрагменте?

```
type
  MyClass = class
    public
      class function GetInstance(): MyClass;
  end;

class function MyClass.GetInstance(): MyClass;
begin
end;
```

Почему вместо имени объекта используется MyClass?

Очень интересно.

А *теперь* я могу создать экземпляр MyClass?

```
class function MyClass.GetInstance(): MyClass;
begin
  Result := MyClass.Create;
end;
```

Как должны создаваться экземпляры в клиентском коде?

```
MyClass.getInstance();
```

Можете ли вы дописать код, чтобы он всегда создавал не более ОДНОГО экземпляра MyClass?

Классическая реализация паттерна Одиночка

```
type
  Singleton = class
    private
      UniqueInstance: Singleton; static;
      constructor Create;
    public
      class function GetInstance(): Singleton;
  end;
constructor Singleton.Create;
begin
end;
class function Singleton.GetInstance(): Singleton;
begin
  if UniqueInstance = nil then
    UniqueInstance := Singleton.Create;
  Result := Singleton.Create;
end;
```

Класс MyClass переименован в Singleton.

Статическая переменная для хранения единственного экземпляра.

Приватный конструктор; только Singleton может создавать экземпляры этого класса!

Метод getInstance() создает и возвращает экземпляр.

Код под увеличительным стеклом

uniqueInstance содержит ЕДИНСТВЕННЫЙ экземпляр; не забудьте, что это статическая переменная.

Если uniqueInstance содержит nil , значит, экземпляр еще не создан...

...тогда мы создаем экземпляр Singleton приватным конструктором и присваиваем его uniqueInstance.

```
if UniqueInstance = nil then
  UniqueInstance := Singleton.Create;
Result := Singleton.Create;
```

Если uniqueInstance уже содержит значение, сразу переходим к команде return.

К моменту выполнения этой команды экземпляр уже создан — возвращаем его.