Workshop

Os principais Design Patterns

Sobre

- Microsoft MVP – Visual Studio and Development Technologies
- Trabalha com desenvolvimento Web desde 2003
- Atuação em projetos nacionais e internacionais de grande porte
- Hoje realiza consultorias e treinamentos na área

Agenda

- Antes de Começar
  - Introdução
  - Cuidados e Quando Utilizar
  - Tipos de Design Patterns
- Alguns Patterns
  - Factory
  - Abstract Factory
  - Singleton
  - Adapter
  - Facade
  - MVC

Introdução

- Padrões de escrita de código
- Independentes de linguagem
- Design Patterns foi um conceito iniciado em 1994
- Criado por um grupo de 4 pessoas, conhecido como Gang of Four
- Em 1994, Erich Gamma, Richard Helm, Ralph Johnson e John Vissides publicaram um livro:
  - Design Patterns – Elements of Reusable Object-Oriented Software

Quais são?

- Factory Pattern
- Abstract Factory Pattern
- Singleton Pattern
- Builder Pattern
- Prototype Pattern
- Adapter Pattern
- Bridge Pattern
- Filter Pattern
- Composite Pattern
- Decorator Pattern
- Facade Pattern
- Flyweight Pattern
- Proxy Pattern
- Chain of Responsibility Pattern
- Command Pattern
- Interpreter Pattern
- Iterator Pattern
- Mediator Pattern
- Memento Pattern
- Observer Pattern
- State Pattern
- Null Object Pattern
- Strategy Pattern
- Template Pattern
- Visitor Pattern
- MVC Pattern
- Business Delegate Pattern
- Composite Entity Pattern
- Data Access Object Pattern
- Front Controller Pattern
- Intercepting Filter Pattern
- Service Locator Pattern
- Transfer Object Pattern

Além dos Design Patterns

- A importância de um código limpo
- Um código bem escrito já é documentado
- Identação, nomeação de variáveis, linguagem ubíqua, complexidade de código, divisão de responsabilidades, separação de conceitos, etc

Cuidados

- Não utilize o que você não sabe
- Não é por que você sabe que tem que utilizer
- Não é uma competição “Quem usa mais DP”

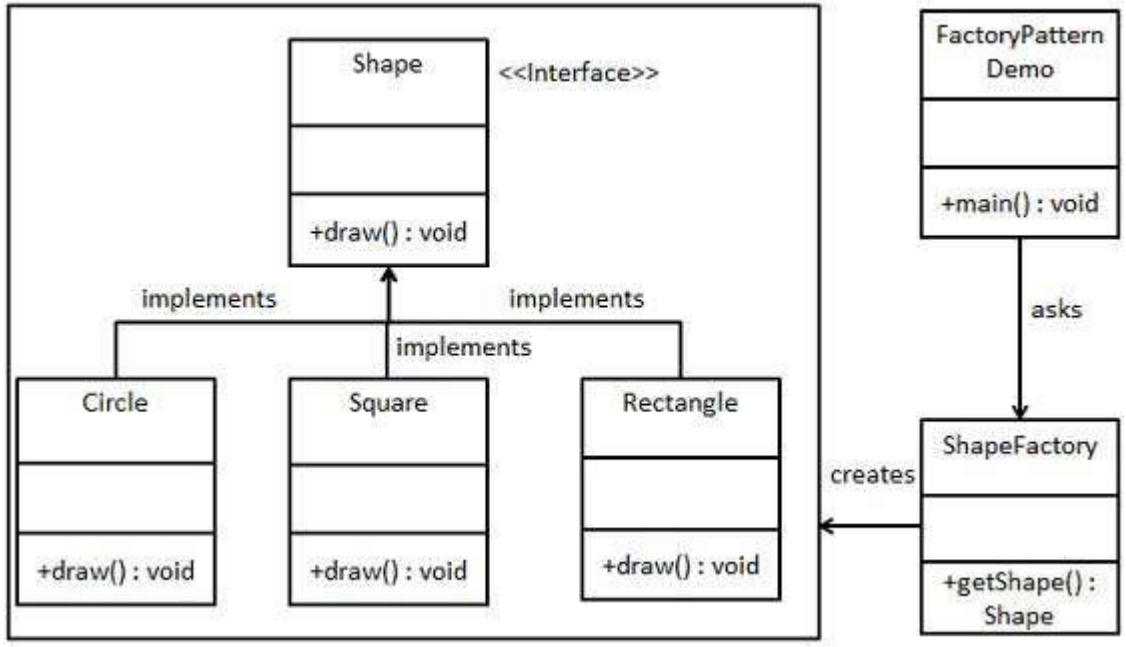
Tipos de Design Patterns

- Creational Patterns (Criacional)
  - Fornecem uma maneira de criar objetos sem a necessidade do uso de uma nova instância do mesmo (Baseado em abstração por exemplo).
  - Fornecem maior flexibilidade na decisão de qual objeto será criado.
- Structural Patterns (Estrutural)
  - Focam nas classes e composição de objetos.
  - Utilizam o conceito de herança para compor interfaces e então definir objetos obtendo assim novas funcionalidades.
- Behavioral Patterns (Comportamental)
  - Focam na comunicação entre objetos.

Factory

- É um dos Patterns mais utilizados
- Fornece um meio de criar um objeto sem expor sua lógica de criação
- Durante o consumo deste objeto, ele será referenciado como uma interface.

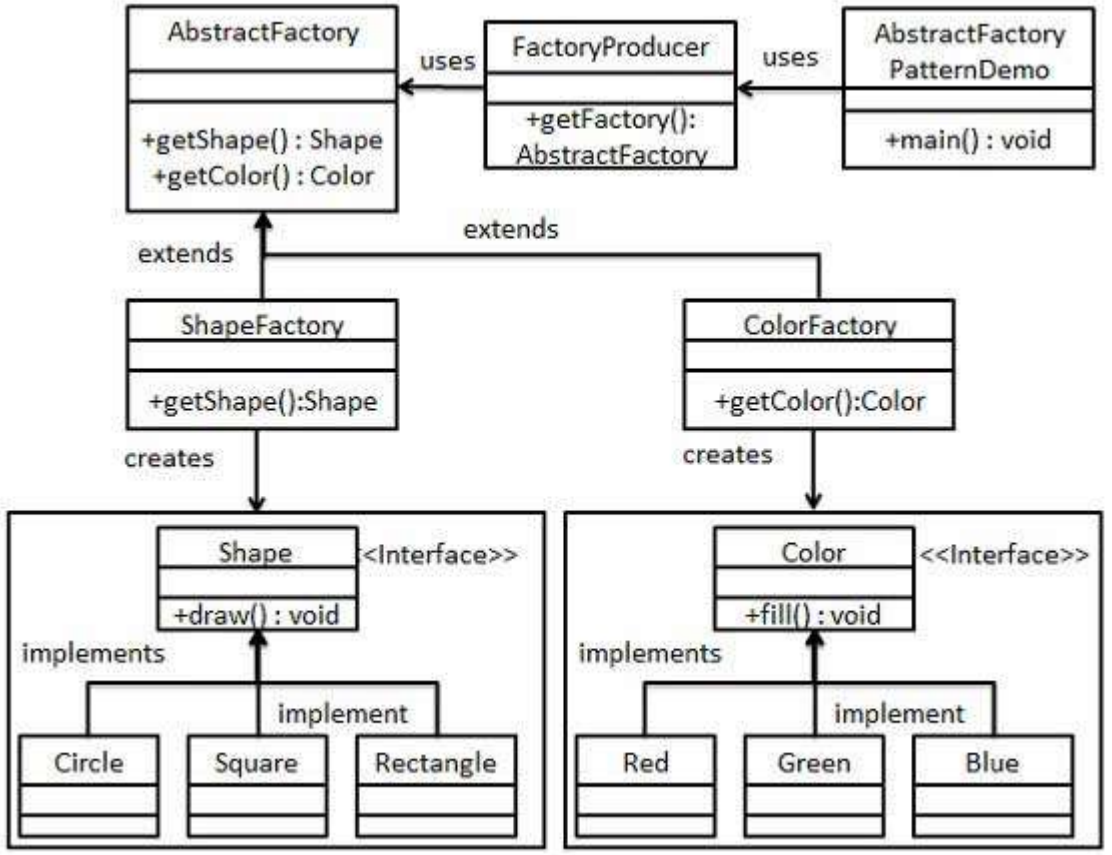
Factory


Shape
<<Interface>>
+draw() : void
implements
implements
implements
Circle
+draw() : void
Square
+draw() : void
Rectangle
+draw() : void
FactoryPattern Demo
+main() : void
asks
creates
ShapeFactory
+getShape() : Shape

Abstract Factory

- Atua como um super-factory, que cria outras factories
- Também pode ser chamado de “Factory of Factories”
- Neste padrão, uma interface é responsável por cirar uma Factory de um objeto relacionado, sem expor suas classes

Abstract Factory

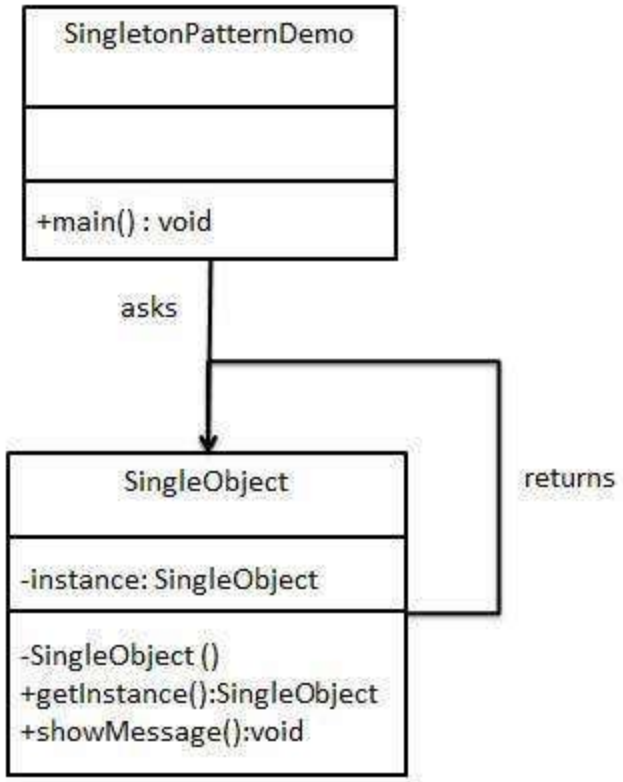


Singleton

- Este padrão envolve uma classe na qual a responsabilidade é criar um objeto e ter certeza que apenas um deste objeto será criado.

- Esta classe fornece uma maneira de acessar este objeto, na qual pode ser chamada diretamente, sem a necessidade de uma nova instância.

Singleton

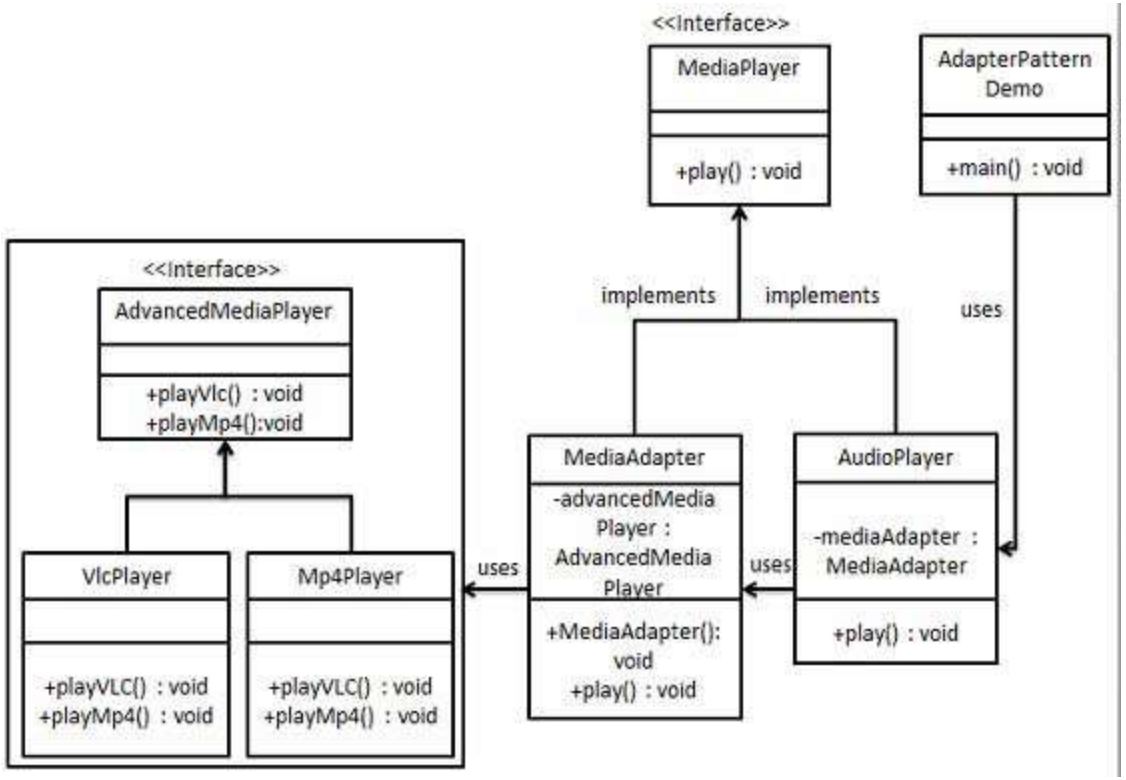

SingletonPatternDemo
+main() : void
asks
SingleObject
-instance: SingleObject
-SingleObject ()
+getInstance():SingleObject
+showMessage():void
returns

Adapter

- Este padrão funciona como uma ponte entre duas interfaces incompatíveis.

- Ele é na verdade uma única classe cuja responsabilidade é unir funcionalidades de interfaces independentes e imcompatíveis.

Adapter

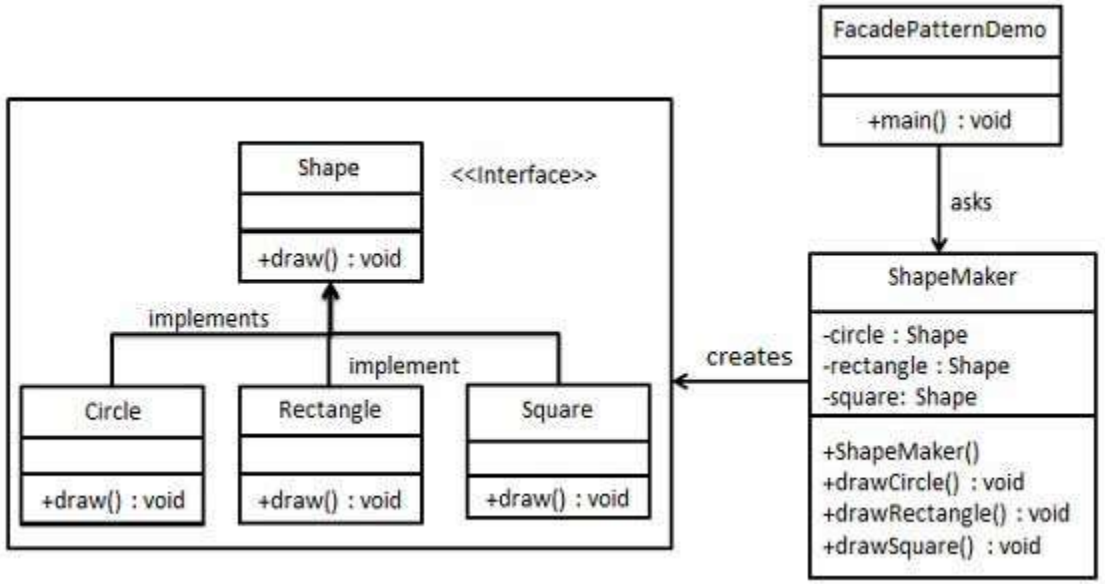

<<Interface>>
MediaPlayer
+play() : void
AdapterPattern Demo
+main() : void
implements
implements
uses
<<Interface>>
AdvancedMediaPlayer
+playVlc() : void
+playMp4():void
MediaAdapter
-advancedMedia Player : AdvancedMedia Player
+MediaAdapter(): void
+play() : void
AudioPlayer
-mediaAdapter : MediaAdapter
+play() : void
uses
uses
VlcPlayer
+playVLC() : void
+playMp4() : void
Mp4Player
+playVLC() : void
+playMp4() : void

FACADE

- Este padrão esconde a complexidade do sistema provisionando uma interface ao cliente.

- Esta interface prove métodos simplificados ao cliente, tornando a vida dele mais fácil.

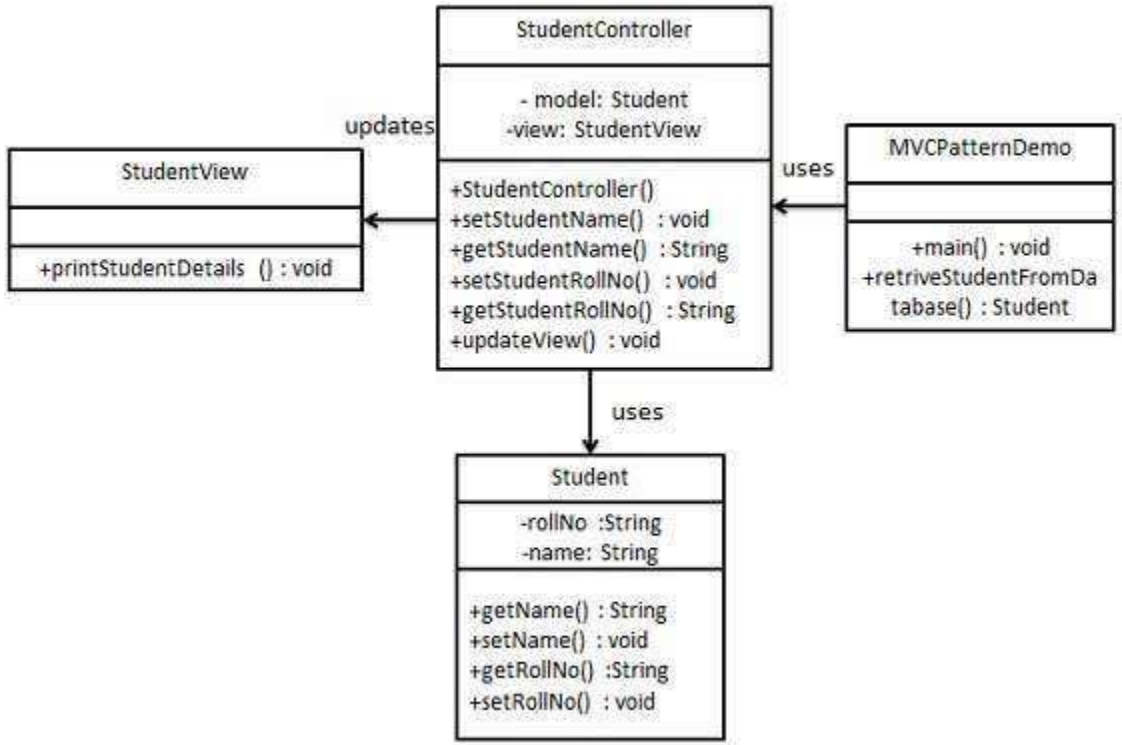
FACADE


FacadePatternDemo
+main() : void
asks
ShapeMaker
-circle : Shape
-rectangle : Shape
-square: Shape
+ShapeMaker()
+drawCircle() : void
+drawRectangle() : void
+drawSquare() : void
creates
Shape
<<Interface>>
+draw() : void
implements
implement
Circle
+draw() : void
Rectangle
+draw() : void
Square
+draw() : void

MVC

- Sigla para Model-View-Controller
- É um padrão de separação de responsabilidades
- Model – Representa seu domínio, o core, os objetos do seu Sistema
- View – Representa a tela, a interface com o usuário
- Controller – Faz a ligação entre o modelo e a visão. Recebe os dados da tela, trabalha no model e retorna um resultado para tela.

MVC


StudentController
- model: Student
-view: StudentView
+StudentController()
+setStudentName() : void
+getStudentName() : String
+setStudentRollNo() : void
+getStudentRollNo() : String
+updateView() : void
updates
StudentView
+printStudentDetails () : void
uses
MVCPatternDemo
+main() : void
+retriveStudentFromDatabase() : Student
uses
Student
-rollNo :String
-name: String
+getName() : String
+setName() : void
+getRollNo() :String
+setRollNo() : void

Fonte

- Design Patterns in Java
  - http://www.tutorialspoint.com/design_pattern/factory_pattern.htm

TREINAMENTO

Aplicando Design Patterns em Aplicações Corporativas

http://bit.ly/abt-5508

Microsoft®
Most Valiable
Professional

andrebaltieri.net
contato@andrebaltieri.net

OBRI

CADO

Microsoft®
Most Valiable
Professional

andrebaltieri.net
contato@andrebaltieri.net

Fique por dentro de todas as **NOVIDADES!**

ASSINE O NEWSLETTER

http://bit.ly/abt-news

Curta a Fanpage

/andre.baltieri

Se inscreva no canal

/andrebaltieri

ANDRÉ BALTIERI

treinamentos e consultorias