Padrão mais utilizado

Codigo exemplo => Client ( AutoSocorro )

using System;

namespace DesignPatterns.AbstractFactory

{

// Client Class

public class AutoSocorro

{

private readonly Veiculo \_veiculo;

private readonly Guincho \_guincho;

public AutoSocorro(AutoSocorroFactory factory, Veiculo veiculo)

{

\_veiculo = factory.CriarVeiculo(veiculo.Modelo, veiculo.Porte);

\_guincho = factory.CriarGuincho();

}

public void RealizarAtendimento()

{

\_guincho.Socorrer(\_veiculo);

}

public static AutoSocorro CriarAutoSocorro(Veiculo veiculo)

{

switch (veiculo.Porte)

{

case Porte.Pequeno:

return new AutoSocorro(new SocorroVeiculoPequenoFactory(), veiculo);

case Porte.Medio:

return new AutoSocorro(new SocorroVeiculoMedioFactory(), veiculo);

case Porte.Grande:

return new AutoSocorro(new SocorroVeiculoGrandeFactory(), veiculo);

default:

throw new ApplicationException("Não foi possível identificar o veículo");

}

}

}

}

Codigo exemplo => AbstractFactory ( AutoSocorroFactory )

namespace DesignPatterns.AbstractFactory

{

// Abstract Factory

public abstract class AutoSocorroFactory

{

public abstract Guincho CriarGuincho();

public abstract Veiculo CriarVeiculo(string modelo, Porte porte);

}

}

Codigo exemplo => ConcreterFactory1 (SocorroVeiculoGrandeFactory )

namespace DesignPatterns.AbstractFactory

{

// Concrete Factory

public class SocorroVeiculoGrandeFactory : AutoSocorroFactory

{

public override Guincho CriarGuincho()

{

return GuinchoCreator.Criar(Porte.Grande);

}

public override Veiculo CriarVeiculo(string modelo, Porte porte)

{

return VeiculoCreator.Criar(modelo, porte);

}

}

}

Codigo exemplo => ConcreterFactory2 (SocorroVeiculoMedioFactory )

namespace DesignPatterns.AbstractFactory

{

// Concrete Factory

public class SocorroVeiculoMedioFactory : AutoSocorroFactory

{

public override Guincho CriarGuincho()

{

return GuinchoCreator.Criar(Porte.Medio);

}

public override Veiculo CriarVeiculo(string modelo, Porte porte)

{

return VeiculoCreator.Criar(modelo, porte);

}

}

}

Codigo exemplo => ConcreterFactory3 (SocorroVeiculoPequenoFactory )

namespace DesignPatterns.AbstractFactory

{

// Concrete Factory

public class SocorroVeiculoPequenoFactory : AutoSocorroFactory

{

public override Guincho CriarGuincho()

{

return GuinchoCreator.Criar(Porte.Pequeno);

}

public override Veiculo CriarVeiculo(string modelo, Porte porte)

{

return VeiculoCreator.Criar(modelo, porte);

}

}

}

Codigo exemplo => AbstractFactory (ExecucaoAbstractFactory)

using System.Collections.Generic;

namespace DesignPatterns.AbstractFactory

{

public class ExecucaoAbstractFactory

{

public static void Executar()

{

var veiculosSocorro = new List<Veiculo>

{

VeiculoCreator.Criar("Celta", Porte.Pequeno),

VeiculoCreator.Criar("Jetta", Porte.Medio),

VeiculoCreator.Criar("BMW X6", Porte.Grande)

};

veiculosSocorro.ForEach(v => AutoSocorro.CriarAutoSocorro(v).RealizarAtendimento());

}

}

}

Codigo exemplo => AbstractProdutoA (Guincho)

using System;

namespace DesignPatterns.AbstractFactory

{

// AbstractProduct

public abstract class Guincho

{

protected Guincho(Porte porte)

{

Porte = porte;

}

public abstract void Socorrer(Veiculo veiculo);

public Porte Porte { get; set; }

}

// Produto Concreto

public class GuinchoPequeno : Guincho

{

public GuinchoPequeno(Porte porte) : base(porte) { }

public override void Socorrer(Veiculo veiculo)

{

// Processo de socorro

Console.WriteLine("Socorrendo Carro Pequeno - Modelo " + veiculo.Modelo);

}

}

// Produto Concreto

public class GuinchoMedio : Guincho

{

public GuinchoMedio(Porte porte) : base(porte) { }

public override void Socorrer(Veiculo veiculo)

{

// Processo de socorro

Console.WriteLine("Socorrendo Carro Medio - Modelo " + veiculo.Modelo);

}

}

// Produto Concreto

public class GuinchoGrande : Guincho

{

public GuinchoGrande(Porte porte) : base(porte) { }

public override void Socorrer(Veiculo veiculo)

{

// Processo de socorro

Console.WriteLine("Socorrendo Carro Grande - Modelo " + veiculo.Modelo);

}

}

public class GuinchoCreator

{

public static Guincho Criar(Porte porte)

{

switch (porte)

{

case Porte.Pequeno:

return new GuinchoPequeno(porte);

case Porte.Medio:

return new GuinchoMedio(porte);

case Porte.Grande:

return new GuinchoGrande(porte);

default:

throw new ApplicationException("Porte de Guincho desconhecido.");

}

}

}

}

Codigo exemplo => AbstractProdutoB (Veiculo)

using System;

namespace DesignPatterns.AbstractFactory

{

// AbstractProduct

public abstract class Veiculo

{

protected Veiculo(string modelo, Porte porte)

{

Modelo = modelo;

Porte = porte;

}

public string Modelo { get; set; }

public Porte Porte { get; set; }

}

public enum Porte

{

Pequeno,

Medio,

Grande

}

// Produto Concreto

public class VeiculoPequeno : Veiculo

{

public VeiculoPequeno(string modelo, Porte porte) : base(modelo, porte)

{

}

}

// Produto Concreto

public class VeiculoMedio : Veiculo

{

public VeiculoMedio(string modelo, Porte porte) : base(modelo, porte)

{

}

}

// Produto Concreto

public class VeiculoGrande : Veiculo

{

public VeiculoGrande(string modelo, Porte porte) : base(modelo, porte)

{

}

}

public class VeiculoCreator

{

public static Veiculo Criar(string modelo, Porte porte)

{

switch (porte)

{

case Porte.Pequeno:

return new VeiculoPequeno(modelo, porte);

case Porte.Medio:

return new VeiculoMedio(modelo, porte);

case Porte.Grande:

return new VeiculoGrande(modelo, porte);

default:

throw new ApplicationException("Porte de veiculo desconhecido.");

}

}

}

}





