Este padrão converte a interface de uma classe em outra interface, que seja esperada pelo cliente. O padrão Adapter permite que interfaces incompatíveis possam trabalhar juntas, o que seria bem impossível de outra forma. Quando buscamos resolver incompatibilidades em um projeto, mas especificamente entre suas classes, quando são de interfaces diferentes, e precisam ser compatíveis entre si, do mesmo tipo.

Estrutura e UML do Adaptador de classes, um exemplo usando a linguagem C#:

namespace PadraoAdapter

{

class EntradaPS2

{

*//Solicitação Especifica*

public void conectarEntradaPS2()

{

Console.Write("Conectado na entrada PS2");

}

}

*//Classe alvo(Target)*

class EntradaUSB

{

*//Solicitação*

public void conectarEntradaUSB()

{

Console.Write("Conectado na entrada USB");

}

}

*//Classe adaptador(Adapter)*

class AdapterEntrada : EntradaUSB

{

**private** EntradaPS2 entradaPS2;

**public** AdapterEntrada(EntradaPS2 entradaPs2)

{

**this**.entradaPS2 = entradaPs2;

}

*//Solicitação*

**public** **void** conectarEntradaUSB()

{

entradaPS2.conectarEntradaPS2();

}

}

**class** **Teclado**

{

**static** **void** Main(string[] args)

{

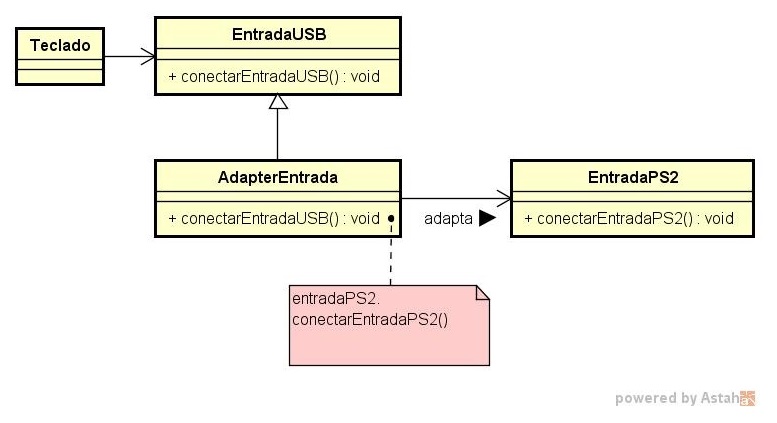
EntradaPS2 ps2 = **new** EntradaPS2();

AdapterEntrada adaptador = **new** AdapterEntrada(ps2);

adaptador.conectarEntradaUSB();

}

}

} 

Abordagem Adapter de Classes: Usando também a linguagem C#

**namespace** **PadraoAdapter**

{

*//Adaptada(adaptee)*

**class** **ClasseExistente**

{

**public** **void** exibir()

{

Console.Write("Metodo exibir()");

}

}

*//Adaptada(adaptee)*

**interface** InterfaceCliente

{

**void** mostrar();

}

*//Adaptador(adapter)*

**class** **Adaptador** : ClasseExistente, InterfaceCliente

{

**public** **void** mostrar()

{

exibir();

}

}

**class** **Cliente**

{

**static** **void** Main(string[] args)

{

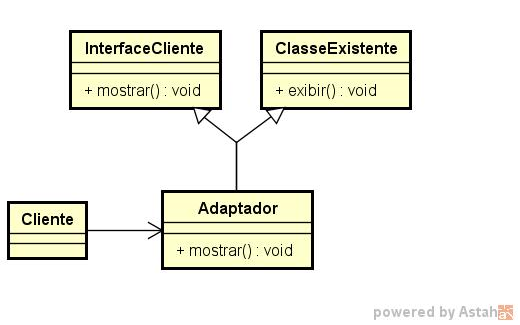
InterfaceCliente existente = **new** Adaptador();

existente.mostrar();

}

}

}



Prefiro Object Adapter. Porque pelo que vi, não precisaríamos modificar comportamentos, digo que teria mais flexibilidade no designer, e só seria preciso apenas um objeto Adapter para trabalhar com quaisquer adaptados, ao contrário do Class Adapter que nos obriga sem dúvidas a redefinir comportamentos, o que poderia causar desvantagens, em resumo, apenas um objeto do Adapter(Object Adapter), teríamos mais flexibilidade e código mais conciso e curto, e fora que demonstra ter mais funcionalidades que o Class Adapter, visto que este é limitado a adaptar subclasses e a superclasse.