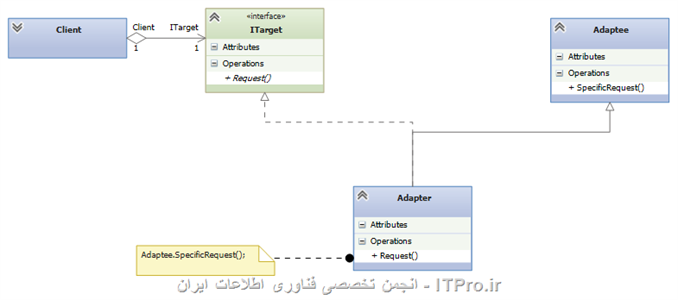
الگوی Adapter

همان گونه که از اسم این الگو مشخص است، هنگامی که دو کلاس واسط(Interface) های غیرمرتبط با یکدیگر داشته باشند این الگو واسط یکی را به دیگری تبدیل می کند که بتوانند با یکدیگر ارتباط برقرار کند.   
از این الگو که یک الگوی ساختاری است زمانی استفاده می شود که در یک برنامه بخواهیم که دو کلاس غیرمرتبط با یکدیگر کار کنند. این الگو در برنامه هایی که از کلاس های آماده استفاده می شود و یا از کلاس هایی استفاده می شود که قبلا نوشته شده اند و به گونه ای باشد که طراحان نرم افزار اجازه تغییر در این کلاس ها را نداشته باشند. بسیاری از مثال های الگوی Adapter درگیر ورودی و خروجی هستند زیرا این دامنه ای است که همیشه تغییر می کند. برای مثال برنامه هایی که در دهه 80 میلادی نوشته شده اند دارای UI بسیار ضعیف تری نسبت به برنامه های نوشته شده در هزاره سوم میلادی هستند. حال اگر دوباره نویسی همان برنامه های قبلی به صرفه نباشد و بخواهیم این برنامه ها با سخت افزارهای جدید همخوانی و سازگاری داشته باشند باید برنامه ای طراحی کنیم که بین این دو برنامه یعنی برنامه قدیمی و راه اندازهای سخت افزارهای جدید قرار بگیرد و بین این دو برنامه ارتباط برقرار کند. به برنامه ای که بین این دو برنامه قرار می گیرد Adapter گویند. حال شاید به نظر برسد که استفاده از Adapter اصلاً نیاز نباشد و می توان واسط یکی از کلاس ها را تغییر داد تا بتواند کار کند، این کار زمانی ممکن است که بتوان به کد کلاس ها دسترسی داشت و تغییر کلاس ها باعث به وجود آمدن مشکل در برنامه نباشد و پیچیدگی برنامه را افزایش ندهد که در اکثر مواقع این عمل به صرفه نیست.   
ساختار این الگو به این شکل است:



همانطور که در شکل دیده می شود کلاس Adapter از اینترفیس ITarget مشتق شده است و در یکی از متدهایی که در این کلاس وجود دارد (Request) متد مورد نظر را از کلاس Adaptee فراخوانی می کند. کلاس Adapter متدهای Adaptee را با متدها و خصوصیای Target وفق می دهد. به طور کلی اینترفیس ITarget همان اینترفیسی است که کاربر آن را پیاده سازی می کند و کلاس Adaptee همان کلاسی است که باید با برنامه وفق داده شود. کاربران برای استفاده فقط از کلاس Adapter نمونه سازی می کنند و نیازی به درگیری با ارتباط Target و Adaptee نخواهند بود.   
کد این الگو به شرح زیر است. 

using System;

public class Adaptee

{

public double SpecificRequest(double a, double b)

{

return a/b;

}

}

public interface ITarget

{

string Request(int a);

}

public class Adapter : Adaptee, ITarget

{

public string Request(int a)

{

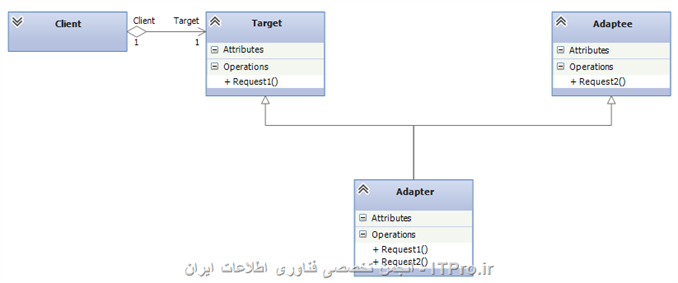
return "the result is" + (int) Math.Round(SpecificRequest(a, 5));

}

}

Adapter دوطرفه

ساده ترین شکل الگوی Adapter شکل یک طرفه آن است که یک کلاس یا اینترفیس Target بوده و کلاس دیگر نقش Adaptee را داشت که تا این قسمت شرح داده شد. حال فرض کنید که می خواهیم هر دو طرف هم نقش Adaptee و هم نقش Target را داشته باشند. به گونه ای که از کلاس Adapter بتوان همه به شکلی استفاده کرد که متدهای هر دو طرف را داشته باشند. در این صورت کلاس Adapter دارای متدها و خصوصیات هر دو کلاس می باشد. یک Adapter دو طرفه می تواند به جای هر دو کلاس Target و Adaptee مورد استفاده قرار بگیرد. برای این کار در زبانهایی مانند ++C که وراثت از چندین کلاس را پشتیبانی می کنند کار آسانی است. به این شکل که کلاس Adapter از دو کلاس دیگر ارث بری داشته باشد همانند ساختار زیر



اما در بسیاری از زبان های [برنامه نویسی](http://itpro.ir/gotorel/technichaltext/32786) رایج مانند #C و Java ارث بری از چندکلاس پشتیبانی نمی شود. بنابراین این راه حل در این زبان ها پاسخگو نیست. برای این که در این زبان ها از Adapter دو طرفه استفاده کنیم به شکل زیر عمل می کنیم. 

public class Adaptee

{

public double SpecificRequest(double a, double b)

{

return a/b;

}

}

public interface ITarget

{

string Request(int a);

}

public class Adapter : Adaptee, ITarget

{

public string Request(int a)

{

return "the result is" + (int) Math.Round(SpecificRequest(a, 5));

}

}

////////////////////////////////////////////////////////

internal class Program

{

private static void Main(string[] args)

{

ITarget aa = new Adapter();

Console.WriteLine(aa.Request(5));

Console.WriteLine((aa as Adaptee).SpecificRequest(8, 3));

}

}

در کد بالا یکی از اینترفیس ها (IAircraft ) که همان Target است را در کلاس Adapter پیاده سازی می کنیم و از کلاس Adaptee ارث بری می نماییم. ساختار این عمل شبیه به همان استفاده از الگوی Adapter به شکل یک طرفه آن است ولی در استفاده به این شکل عمل می کنیم. همانطور که مشاهده می شود می توان از کلاس Adapter هم به عنوان یک نمونه از Target و هم به عنوان یک نمونه از Adaptee استفاده نمود. مشکلی که در زبان هایی مانند C# یا Java وجود دارد این است که باید Target را به عنوان اینترفیس معرفی کرده و در کلاس Adapter آن را پیاده سازی نمود ولی در زبانی مانند C++ چنین محدودیتی وجود ندارد.