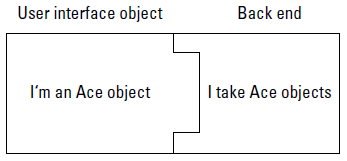
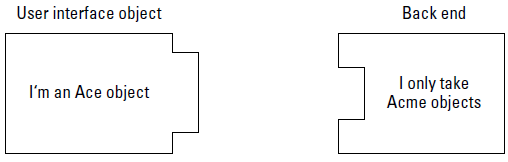
**ADAPTER PATTERN**

Để có thể hiểu được Adapter pattern ta coi một ví dụ được mô tả bằng hình ảnh sau.

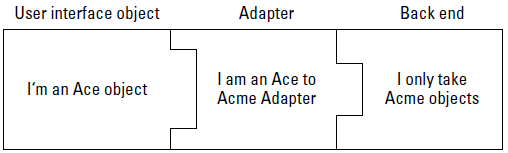
Ban đầu gói phần mềm của chúng ta gồm 2 thành phần phù hợp với nhau



Tuy nhiên do nhu cầu của người dùng mún sử dụng một object khác (Acme) chứ không phải là Ace ta phải thay đổi Back end. Vấn đề xảy ra khi User interface không phù hợp, sẽ là rất tốn kém nếu chúng ta xây dựng lại interface cho Acme object



Giải pháp được đưa ra là sử dụng adapter pattern, lúc này ta sẽ biến ace object thành 1 adapter.

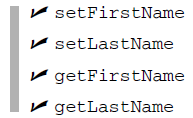


Adapter pattern giúp chúng ta fix interface giữa object và class mà không can thiệp trực tiếp vào object hoặc class. Chúng ta hãy cùng phân tích ví dụ trên (các hình) để tìm hiểu Adapter pattern làm việc như thế nào.

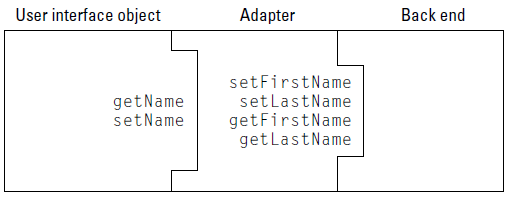
Khi chúng ta làm việc với đối tượng, chúng ta thường không thể can thiệp sâu bên trong để làm cho nó đa dạng hơn hay phù hợp với ứng dụng khác. Chúng ta quay lại với ví dụ ở trên, bây giờ ta thấy đầu tiên có class Ace, class này có 2 phương thức



Nhưng vấn đề bây giờ là ta muốn sử dụng gói Acme. Acme hoàn toàn khác Ace với 4 phướng thức



Do đó chúng ta cần adapter để có sự chuyển đổi thích hợp cho interface Ace đối với object Acme.



Đó là ý tưởng mà chúng ta cần thực hiện. Chúng ta có thể nói Adapter pattern như sau “Convert the interface of a class into another interface the client expects.”

Đầu tiên ta có interface Ace

public interface AceInterface

{

public void setName(String n);

public String getName();

}

Sau đó có đối tượng Ace

public class AceClass implements AceInterface

{

**String name;**

**public void setName(String n)**

**{**

**name = n;**

**}**

**public String getName()**

**{**

**return name;**

**}**

}

Đó là tất cả những gì về Ace. Bây giờ chúng ta coi tiếp tới Acme. Đầu tiên là interface của nó.

public interface AcmeInterface

{

public void setFirstName(String f);

public void setLastName(String l);

public String getFirstName();

public String getLastName();

}

Tiếp theo là Acme class

public class AcmeClass implements AcmeInterface

{

**String firstName;**

**String lastName;**

**public void setFirstName(String f)**

**{**

**firstName = f;**

**}**

**public void setLastName(String l)**

**{**

**lastName = l;**

**}**

**public String getFirstName()**

**{**

**return firstName;**

**}**

**public String getLastName()**

**{**

**return lastName;**

**}**

}

Công việc tiếp theo của chúng ta là xây dựng adapter Ace-to-Acme. Adapter có tác dụng giúp cho Acme object có thể làm việc với 1 Ace object, do đó chúng ta sẽ tạo ra object adapter. Và vì object adapter giống như Acme object nên ta sử dụng interface của Acme.

public class AceToAcmeAdapter implements AcmeInterface

{

**AceClass aceObject;**

**public AceToAcmeAdapter(AceClass a)**

**{**

**aceObject = a;**

**}**

.

.

.

}

Sự khác biệt giữa Ace và Acme là Ace lưu trữ tên khách hàng bằng 1 string trong khi Acme lưu trữ tách rời phần tên và phần họ. Đề xây dựng adapter giữa Ace và Acme, chúng ta chia tên lưu trong Ace object thành phần họ và tên rõ ràng, đồng thời thực thi các hàm trong interface Acme.

public class AceToAcmeAdapter implements AcmeInterface

{

AceClass aceObject;

String firstName;

String lastName;

public AceToAcmeAdapter(AceClass a)

{

aceObject = a;

firstName = aceObject.getName().split(“ “)[0];

lastName = aceObject.getName().split(“ “)[1];

}

**public void setFirstName(String f)**

**{**

**firstName = f;**

**}**

**public void setLastName(String l)**

**{**

**lastName = l;**

**}**

**public String getFirstName()**

**{**

**return firstName;**

**}**

**public String getLastName()**

**{**

**return lastName;**

**}**

}

Như vậy là chúng ta đã có adapter. Bây giờ chúng ta làm 1 bài test thử sử dụng nó. Ban đầu ta có thể tạo đối tượng Ace bằng class Ace, sau đó dùng adapter để có thể dùng các thuộc tính của Acme.

public class TestAdapter

{

public static void main(String args[])

{

AceClass aceObject = new AceClass();

aceObject.setName(“Cary Grant”);

AceToAcmeAdapter adapter = new AceToAcmeAdapter(aceObject);

**System.out.println(“Customer’s first name: “ +**

**adapter.getFirstName());**

**System.out.println(“Customer’s last name: “ +**

**adapter.getLastName());**

}

}

Kết quả của đoạn code trên là

Customer’s first name: Cary

Customer’s last name: Grant

Có một cách khác để tạo ra adapter đó là xây dựng 1 class kế thừa từ 2 class Ace và Acme và override lại các phương thức, tuy nhiên việc này không thể làm trong java hay C#, vì java không cho thừa kế từ 2 class, ta có thể thực hiện bằng C++. Sơ đồ cho cách thực hiện như sau

