

CIENCIA DE LA COMPUTACION B

Leonardo Daniel Valdivia Ramos

0.1 DEFINICION:

El patron adapter sirve para integrar funcionalidades que no se pueden implementar por herencia. Esta permite establecer un puente para la clase incompatible y de esta manera se adapta la clase incompatible al resto.

0.2 DESCRIPCION

En una clase abstracta que tiene métodos sus clases heredadas siguen el mismo diseño, en el caso que se quiera adaptar a la clase base una nueva clase heredada se deberá crear una clase donde se adapten los métodos de la clase incompatible a los métodos de la clase base.

0.3 EJEMPLO

Creacion de la clase base Motor en esta clase abstracta crearemos tres métodos encender, acelerar, apagar.

```
1 #include <iostream>
2 using namespace std;
3 class Motor{
4     public:
5         Motor() {}
6         virtual void encender()=0;
7         virtual void acelerar()=0;
8         virtual void apagar()=0;
9 };
```

Clase heredada Motorcomun con la implementación de los métodos encender, acelerar y apagar.

```
1 class Motorcomun : public Motor{
2     public:
```

```

3| Motorcomun() {
4|     cout<<"creando motor comun"<<endl;
5| }
6| void encender() {
7|     cout<<"encendiendo motor comun"<<endl;
8| }
9| void acelerar() {
10|     cout<<"acelerando motor comun"<<endl;
11| }
12| void apagar() {
13|     cout<<"apagando motor comun"<<endl;
14| }
15| };

```

Clase heredada MotorEconomico con la implementación de los métodos encender,acelerar y apagar.

```

1| class Motoreconomico : public Motor{
2|     public:
3|     Motoreconomico() {
4|         cout<<"creando motor economico"<<endl;
5|     }
6|     void encender() {
7|         cout<<"encendiendo motor economico"<<endl;
8|     }
9|     void acelerar() {
10|         cout<<"acelerando motor economico"<<endl;
11|     }
12|     void apagar() {
13|         cout<<"apagando motor economico"<<endl;
14|     }
15| };

```

Clase Motorelectrico es la clase incompatible a la clase base y por ende para adaptarla a nuestra clase base tendremos que crear la clase Motorelectricoadapter que será la compatible.

```
1 class Motorelectrico{
2     private:
3         bool conectado;
4     public:
5         Motorelectrico(){
6             cout<<"creando motor electrico"<<endl;
7             conectado=false;
8         }
9         void conectar() {
10             cout<<"Conectando motor electrico"<<endl;
11             this->conectado = true;
12         }
13
14         void activar() {
15             if (conectado==false) {
16                 cout<<"nose puede activar porque no esta conectado el
17                 motor electrico"<<endl;
18             } else
19                 cout<<"Esta conectado , activando motor electrico"<<
20                 endl;
21         }
22
23         void moverMasRapido() {
24             if (conectado==false) {
25                 cout<<"el motor electrico porque no esta conectado..."
26                 "<<endl;
27             } else {
28                 cout<<"Moviendo mas rapido ...aumentando voltaje"<<
29                 endl;
30             }
31         }
32
33         void detener() {
34             if (conectado==false) {
35                 cout<<"No se puede detener motor electrico porque no
36                 esta conectado"<<endl;
37             } else
38                 cout<<"Deteniendo motor electrico"<<endl;
39         }
40
41         void desconectar() {
```

```

37         cout<<"Desconectando motor electrico"<<endl;
38         this->conectado = false;
39     }
40 };

```

Clase motorelectricoadapter heredada de la clase base y en esta implementaremos los métodos encender, acelerar y apagar haciendo uso también de una instancia de Motorelectrico e adaptaremos los métodos de este a los métodos de la clase base.

```

1  class MotorelectricoAdapter : public Motor{
2      private:
3          Motorelectrico motorelectrico;
4      public:
5          void encender() {
6              motorelectrico.conectar();
7              motorelectrico.activar();
8          }
9          void acelerar() {
10             motorelectrico.moverMasRapido();
11         }
12         void apagar() {
13             motorelectrico.detener();
14             motorelectrico.desconectar();
15         }
16 };

```

Creación de una instancia de motorcomun con los métodos de la clase Motor y creación de una instancia de MotorelectricoAdapter que hace uso de los métodos de la clase base Motor.

```

1  int main() {
2      Motorcomun motorcomun;
3      motorcomun.encender();
4      motorcomun.acelerar();
5      motorcomun.apagar();
6
7      MotorelectricoAdapter electrico;
8      electrico.encender();
9      electrico.acelerar();
10     electrico.apagar();
11 }

```