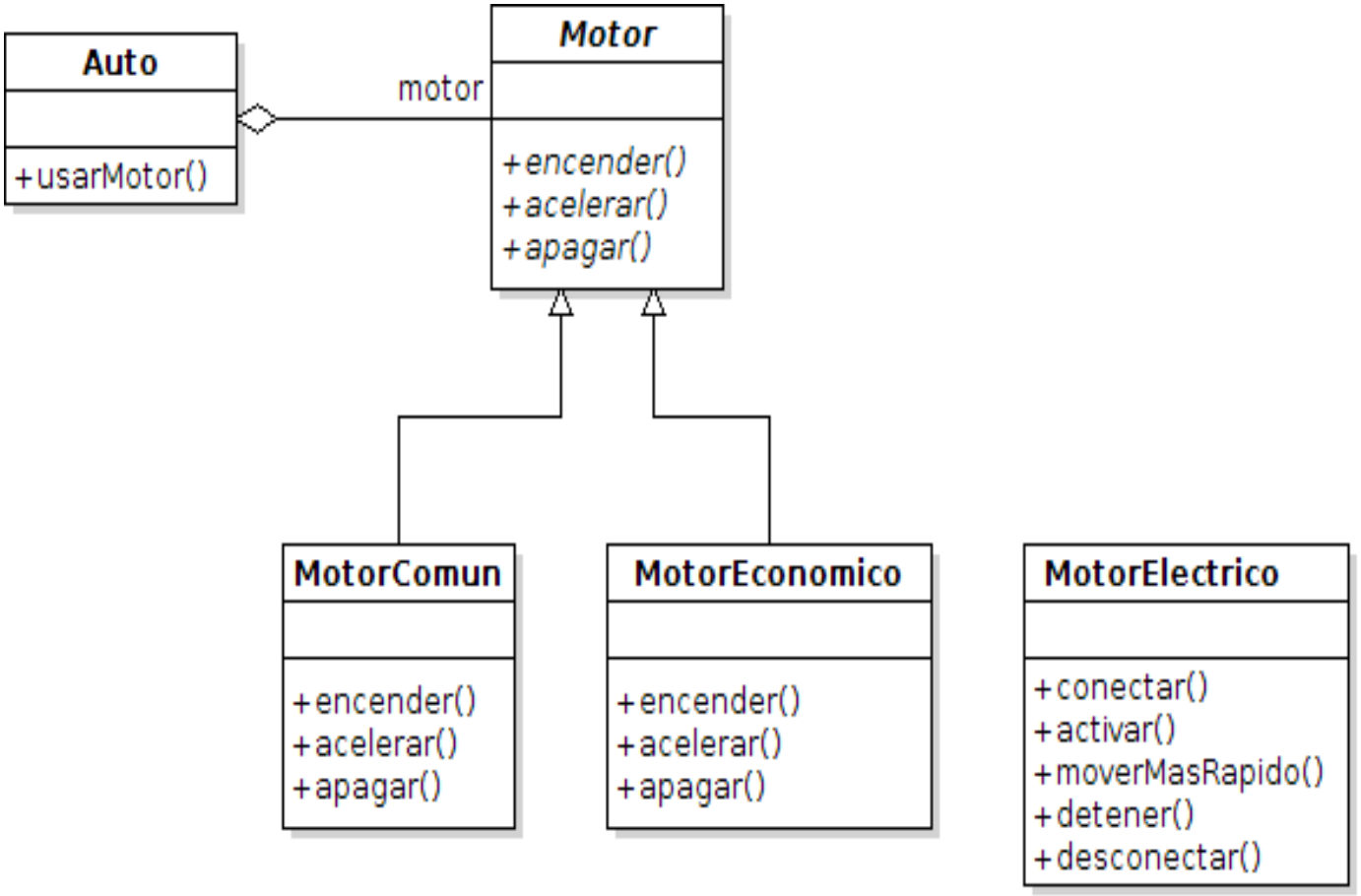
UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

Problema



UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

Patrón Adapter

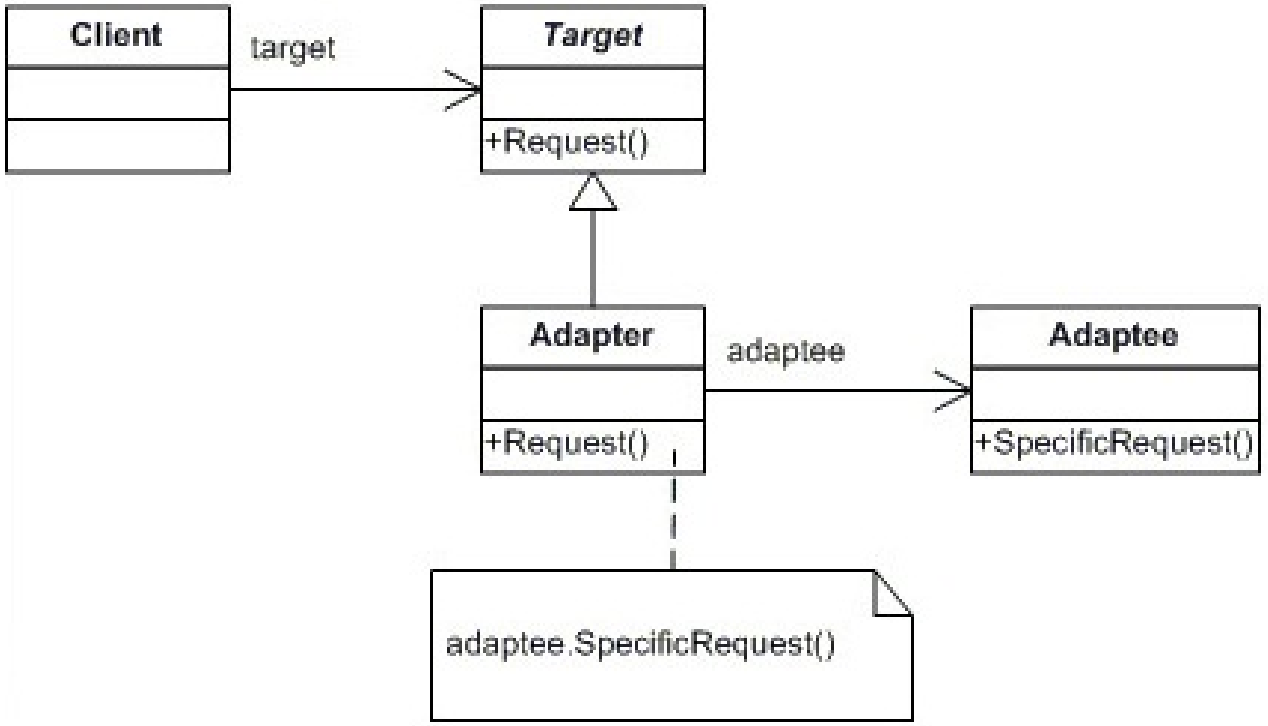
Objetivo:
Convertir el protocolo de una clase en otra, que es la que el objeto cliente espera.

Problema:
Muchas veces, las clases que fueron pensadas para ser reutilizadas no pueden aprovecharse porque su protocolo no es compatible con el protocolo específico del dominio de aplicación con el que se está trabajando.

- Deseamos utilizar una clase existente, cuyo protocolo no encaja con la que necesitamos.
- Deseamos crear una clase que puede llegar a cooperar con otros objetos cuyo protocolo no podemos predecir de antemano.

Patrón Adapter

Solución:
Crear una clase que se encargue de “transformar” los nombres de los mensajes.



UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

Patrón Adapter

Participantes:

Objetivo(target)

- define la interfaz específica del dominio que usa el cliente

Cliente(Client)

- colabora con objetos que se ajustan a la interfaz Objetivo.

Adaptable (Adaptee)

- define una interfaz existente que necesita ser adaptada.

Adaptador (Adapter)

- adapta la interfaz de Adaptable a la interfaz Objetivo.

Colaboraciones

Los clientes llaman a operaciones de una instancia de Adaptador. A su vez, el adaptador llama a operaciones de Adaptable, que son las que satisfacen la petición.

UNTDF
Universidad Nacional de Tierra del Fuego,
Antártida e Islas del Atlántico Sur

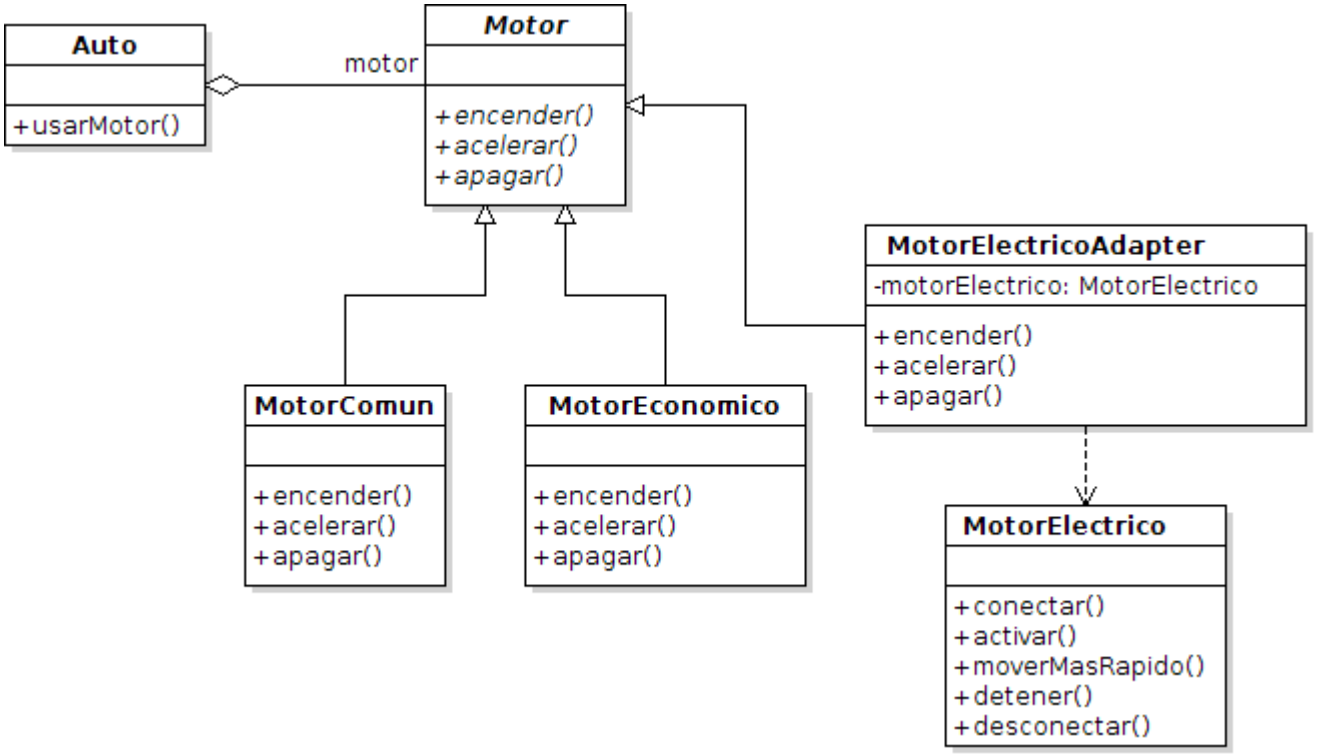
Patrón Adapter

Consecuencias:

- Permite que Adaptador redefina parte del comportamiento de Adaptable.
- Permite que un mismo adaptador funcione con muchos adaptables (con el adaptable en si y todas sus subclases, en caso que las tenga). El adaptador puede también añadir funcionalidad a todos los adaptables a la vez.

Patrón Adapter

Solución:



Patrón Adapter

```
public abstract class Motor {
   abstract public void encender();
   abstract public void acelerar();
   abstract public void apagar();
}

public class MotorComun extends Motor {
     public MotorComun(){
        super();
        System.out.println("Creando el motor comun");
     }
     @Override
     public void encender() {
        System.out.println("encendiendo motor comun");
     }
     @Override
     public void acelerar() {
        System.out.println("acelerando el motor comun");
     }
     @Override
     public void apagar() {
        System.out.println("Apagando motor comun");
     }
}
```

Patrón Adapter

```
public class MotorElectricoAdapter extends Motor{
    private MotorElectrico motorElectrico;
     public MotorElectricoAdapter(){
        this.motorElectrico = new MotorElectrico();
        System.out.println("Creando motor Electrico adapter");
    }
    @Override
    public void encender() {
        System.out.println("Encendiendo motorElectricoAdapter");
        this.motorElectrico.conectar();
        this.motorElectrico.activar();
    }
     @Override
    public void acelerar() {
        System.out.println("Acelerando motor electrico...");
        this.motorElectrico.moverMasRapido();
    }
     @Override
    public void apagar() {
        System.out.println("Apagando motor electrico");
        this.motorElectrico.detener();
        this.motorElectrico.desconectar();
    }
}
```

Patrón Adapter

```
public class MotorElectrico {

    private boolean conectado = false;

    public MotorElectrico() {
        System.out.println("Creando motor electrico");
        this.conectado = false;
    }

    public void conectar() {
        System.out.println("Conectando motor electrico");
        this.conectado = true;
    }

    public void activar() {
        if (!this.conectado) {
            System.out.println("No se puede activar porque no " +
              "esta conectado el motor electrico");
        } else {
            System.out.println("Esta conectado, activando motor" +
              " electrico....");
        }
    }
```

Patrón Adapter

```
    public void moverMasRapido() {
        if (!this.conectado) {
            System.out.println("No se puede mover rapido el motor " +
              "electrico porque no esta conectado...");
        } else {
            System.out.println("Moviendo mas rapido...aumentando voltaje");
        }
    }

    public void detener() {
        if (!this.conectado) {
            System.out.println("No se puede detener motor electrico" +
              " porque no esta conectado");
        } else {
            System.out.println("Deteniendo motor electrico");
        }
    }

    public void desconectar() {
        System.out.println("Desconectando motor electrico...");
        this.conectado = false;
    }
}
```

Patrón Adapter

```
public class Auto{
private Motor motor;
public void usarMotor()
{
motor.encender();
motor.acelerar();
motor.apagar();
}

public void setMotor(Motor m)
{
  motor=m;
}

public static void main(String args[])
{ Auto auto=new Auto();
   auto.setMotor(new MotorElectricoAdapter());
   Auto.usarMotor();
 }
}
```