Práctico de Programación 4 – clase 10

Patrones de diseño

En esta clase veremos dos patrones de diseño que no se ven en el Teórico: Proxy y Adapter.

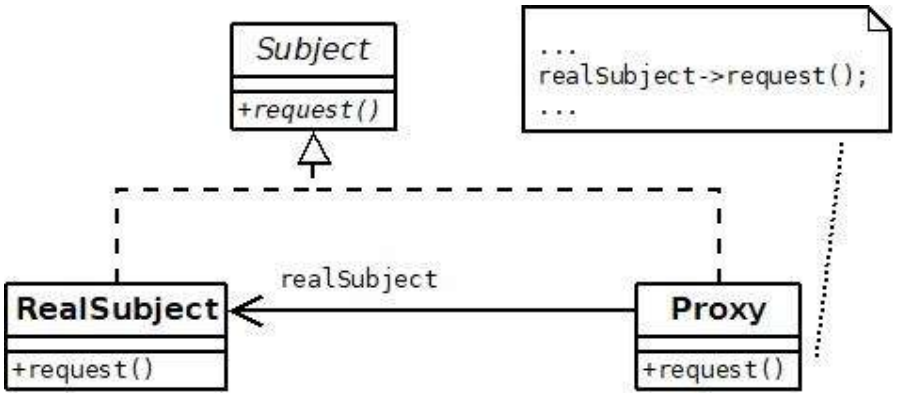
Proxy

Propósito (problema al que aplica): Proporcionar un sustituto para otro objeto, para controlar el acceso a él.

Participantes:

- RealSubject: Define el objeto real al que el proxy representa.
- Subject: Define una interfaz común para RealSubject y Proxy, de tal modo que Proxy pueda ser usado desde cualquier lugar donde se espera una instancia de RealSubject.
- Proxy: Mantiene una referencia que permite acceder a RealSubject. Proporciona una interfaz idéntica a la de Subject, de modo que Proxy pueda ser sustituido por RealSubject.

Estructura y comportamiento:



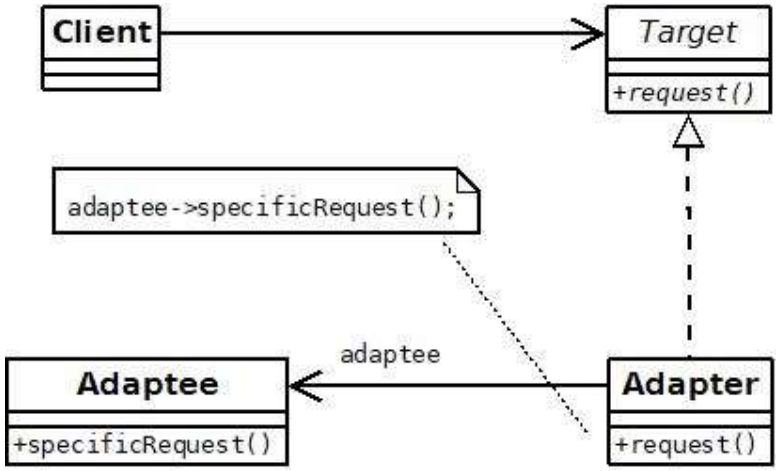
Adapter

Propósito (problema al que aplica): Convertir la interfaz de una clase en otra interfaz que espera el cliente. Permite que dos clases trabajen juntas, las cuales de otro modo no podrían hacerlo por tener interfaces incompatibles.

Participantes:

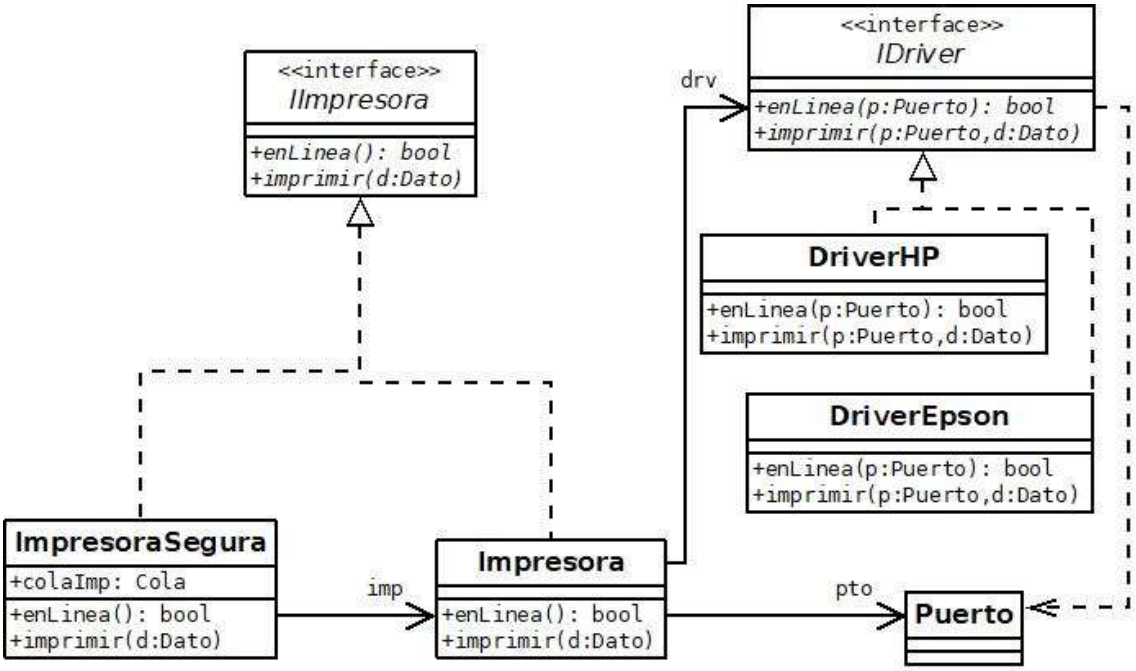
- Target: Define la interfaz que usa Client.
- Client: Interactúa con objectos que cumplan con la interfaz de Target.
- Adaptado: Define una interfaz existente que necesita ser adaptada.
- Adapter: Adapta la interfaz de Adaptado a la interfaz de Target.

Estructura y comportamiento:



Práctico 5, Ejercicio 9:

a)



Los comportamientos de las operaciones son los siguientes (por simplicidad, aquí se presentan como seudocódigos estilo C++, pero lo habitual es presentarlos como notas en el Diagrama de Clases de Diseño o como Diagramas de Comunicación):

```
bool Impresora::enLinea() {

      return drv->enLinea(pto); // Pregunta al driver si está en línea para
                                // imprimir en el puerto
}
```

```
void Impresora::imprimir(Dato d) {
      drv->imprimir(pto, d); // Imprime el dato en el puerto a través del
                             // driver
}

bool ImpresoraSegura::enLinea() {
      return true; // Simula estar siempre en línea
}

void ImpresoraSegura::imprimir(Dato d) {
      colaImp.agregar(d); // Agrega el dato a una cola de impresión
      if (imp->enLinea()) { // Pregunta si la impresora está en línea …
            foreach e in colaImp do
                  imp->imprimir(e);
                  // … en ese caso imprime todo lo almacenado en la cola …
                  // … utilizando la impresora
      }
      colaImp.vaciar(); // Vacía la cola de impresión
}
```

b)

Se utiliza el patrón Proxy con los siguientes roles:

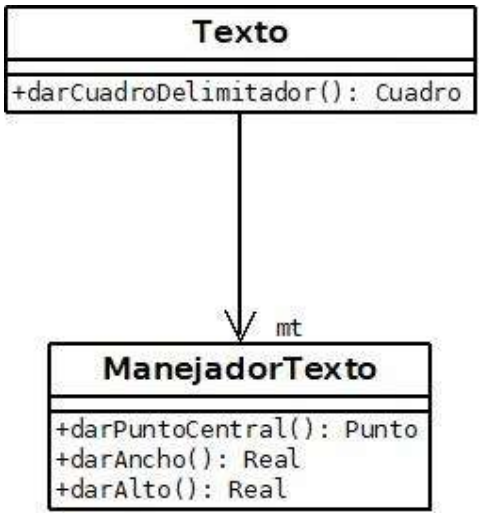
- IImpresora es el Subject. Define la interfaz a la cual invocará un cliente que necesita imprimir.
- Impresora es el RealSubject. Implementa la interfaz de IImpresora y mantiene las referencias al driver y al puerto, objetos necesarios para imprimir.
- ImpresoraSegura. También implementa la interfaz de IImpresora. Necesita de la Impresora para imprimir, pero le agrega un comportamiento seguro.

Práctico 5, Ejercicio 10:

Se utiliza el patrón Adapter con los siguientes roles:

- Editor es el Client. Invoca operaciones de la interfaz Forma.
- Forma es el Target. Implementa la interfaz que utiliza Client.
- Texto es el Adapter. Para devolver el cuadro delimitador necesita de las funcionalidades implementadas en ManejadorTexto, que no cumplen con la interfaz de Forma.
- ManejadorTexto es el Adaptee.
- La operación darCuadroDelimitador es el request.

Se muestra la parte que se modifica del Diagrama de Clases de Diseño.




```
Cuadro Texto::darCuadroDelimitador() {
      mt->setDatos(getDatos());
      Punto p = mt->darPuntoCentral();
      float an = mt->darAncho();
      float al = mt->darAlto();
      Punto pIzq(p.getX()-an/2, p.getY()-al/2);
      Punto pDer(p.getX()+an/2, p.getY()+an/2);
      return Cuadro(pIzq, pDer);
}
```

Para el seudocódigo de la operación (en estilo C++, misma observación que en Ejercicio 9), se asume que:

- ManejadorTexto tiene una operación para setearle la información del texto y su contexto en el dibujo.
- Texto tiene una operación para obtener los datos mencionados anteriormente.
- Cuadro tiene un constructor a partir de su punto inferior izquierdo y su punto superior derecho.
- Punto tiene un constructor a partir de sus coordenadas X e Y.