TP4

Patrones de Estructurales

1. Provea una clase ping que luego de creada al ser invocada con un método "execute(string)" realice 10 intentos de ping a la dirección IP contenida en "string" (argumento pasado), la clase solo debe funcionar si la dirección IP provista comienza con "192.". Provea un método executefree(string) que haga lo mismo pero sin el control de dirección. Ahora provea una clase pingproxy cuyo método execute(string) si la dirección es "192.168.0.254" realice un ping a www.google.com usando el método executefree de ping y re-envie a execute de la clase ping en cualquier otro caso. (Modele la solución como un patrón proxy).

2. Para un producto láminas de acero de 0.5" de espesor y 1,5 metros de ancho dispone de dos trenes laminadores, uno que genera planchas de 5 mts y otro de 10 mts. Genere una clase que represente a las láminas en forma genérica al cual se le pueda indicar que a que tren laminador se enviará a producir. (Use el patrón *bridge* en la solución).

3. Represente la lista de piezas componentes de un ensamblado con sus relaciones jerárquicas. Empiece con un producto principal formado por tres sub-conjuntos los que a su vez tendrán cuatro piezas cada uno. Genere clases que representen esa configuración y la muestren. Luego agregue un sub-conjunto opcional adicional también formado por cuatro piezas. (Use el patrón *composite*).

4. Implemente una clase que permita a un número cualquiera imprimir su valor, luego agregarle sucesivamente.

    a. Sumarle 2.

    b. Multiplicarle por 2.

    c. Dividirlo por 3.

    Mostrar los resultados de la clase sin agregados y con la invocación anidada a las clases con las diferentes operaciones. Use un patrón *decorator* para implementar.

5. Imagine una situación donde pueda ser de utilidad el patrón "*flyweight*".