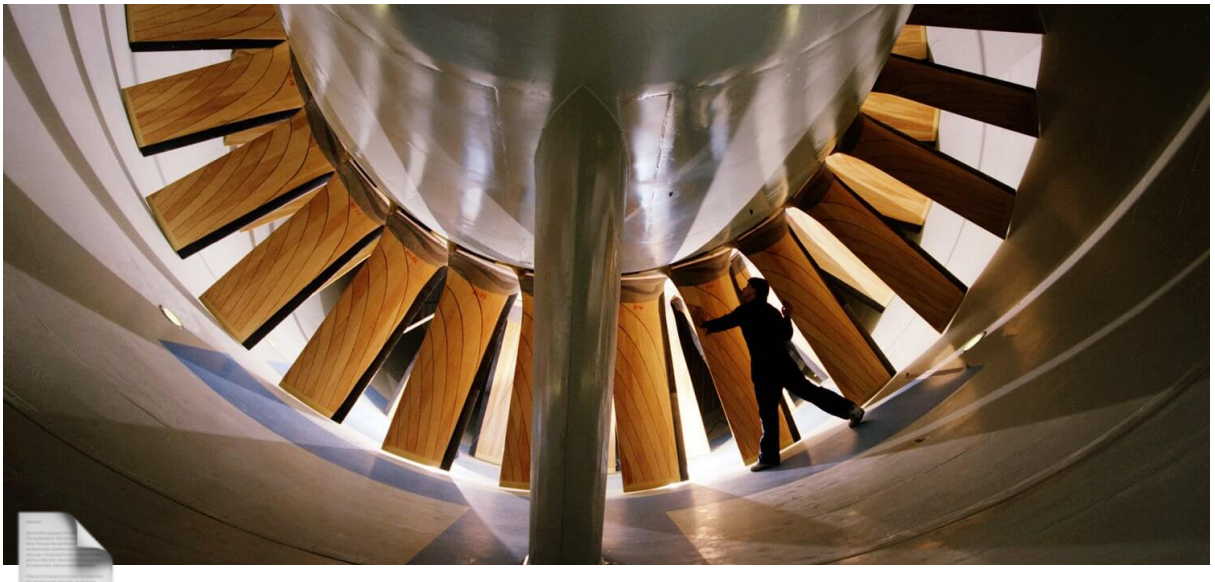

PROG/CREATOR

PROBLEMA:

> Todos los objetos deben ser creados para poder enviarles mensajes que hagan uso de sus responsabilidades de hacer y de conocer.
>
> ¿Quién debería ser responsable de crear una nueva instancia de cierta clase?

SOLUCION:

> Asignar a la clase B la responsabilidad de crear una instancia de la clase A si una de las siguientes condiciones es verdadera:
>
> - B agrega objetos A
> - B contiene objetos A
> - B guarda instancias de objetos A
> - B usa de forma muy cercana objetos A

> - B tiene los datos que serán provistos al constructor para inicializar instancias de A por lo que B es un experto con respecto a crear A.
>
> Entonces B es un creador de objetos A. Cuando se pueda aplicar más de una alternativa, prefieran la clase B que agrega o contiene instancias de A.
>
> En cualquiera de las opciones anteriores la clase B tiene una variable de instancia de A para referenciar objetos de esa clase, o tiene un contener capaz de almacenar instancias de A.

EJEMPLO:

En el ejemplo del patrón Expert, la clase `Sale` maneja su estado a través de las líneas de venta (`SalesLineItem`), las cuales están encapsuladas en el atributo privado `lineItem`. Para agregar o quitar líneas de venta, se usan los métodos `AddLineItem` y `RemoveLineItem`. Sin embargo, la responsabilidad de crear instancias de `SalesLineItem` recae actualmente en la clase `Program`.

```
namespace Creator
{
public class Program
{
private static ArrayList productCatalog = new ArrayList();
```

```
    public static void Main(string[] args)
    {
        PopulateCatalog();

        Sale sale = new Sale();
        sale.DateTime = DateTime.Now;
        sale.AddLineItem(new SalesLineItem(1, ProductAt
(0)));
        sale.AddLineItem(new SalesLineItem(2, ProductAt
(1)));
```

```
        sale.AddLineItem(new SalesLineItem(3, ProductAt
(2)));
        ConsolePrinter.PrintTicket(sale);
    }

    private static void PopulateCatalog()
    {
        AddProductToCatalog("Product 1", 100.00);
        AddProductToCatalog("Product 2", 200.00);
        AddProductToCatalog("Product 3", 300.00);
    }
```

Según la guía Creator, no es correcto que la clase `Program` cree las instancias de `SalesLineItem`. La responsabilidad debería recaer en la clase `Sale`, ya que es quien contiene esas instancias. Por ello, se modifica la clase `Sale` para que sea ella quien cree las instancias de `SalesLineItem`.

```
public class Sale
{
    private List<SalesLineItem> lineItems = new List<SalesLin
    …
    public SalesLineItem AddLineItem(double quantity, Product
    {
        SalesLineItem item = new SalesLineItem(quantity, prod
        this.lineItems.Add(item);
        return item;
    }
    …
}
```

El método `AddLineItem` de la clase `Sale` ahora recibe los datos necesarios para crear instancias de `SalesLineItem`, lo que garantiza que cada instancia creada pertenezca a una `Sale`. Así, la clase `Program` ya no crea estas instancias, sino que solicita a `Sale` que lo haga.

```
public class Program
{
    …
```

```
    public static void Main(string[] args)
    {
        …
        sale.AddLineItem(1, ProductAt(0));
        sale.AddLineItem(2, ProductAt(1));
        sale.AddLineItem(3, ProductAt(2));
        …
    }
    …
```

¿Cuál es el principal problema que se aborda al aplicar la guía Creator?

c) La dependencia entre clases

💬 PROG/DIP