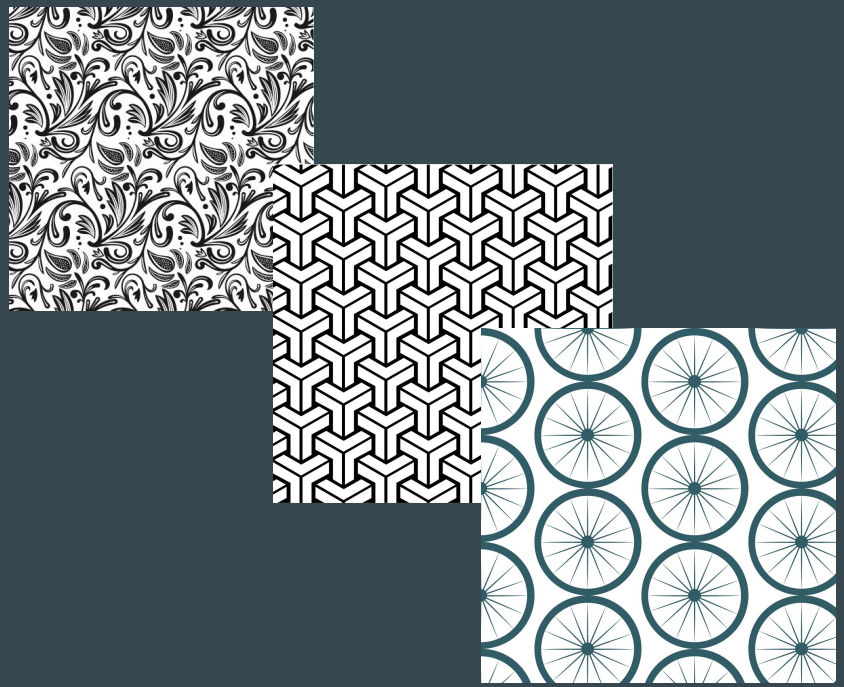
Patrones de Diseño

75.10 - Técnicas de Diseño



Problema

Necesitamos procesar distintos tipos de archivos de Excel.

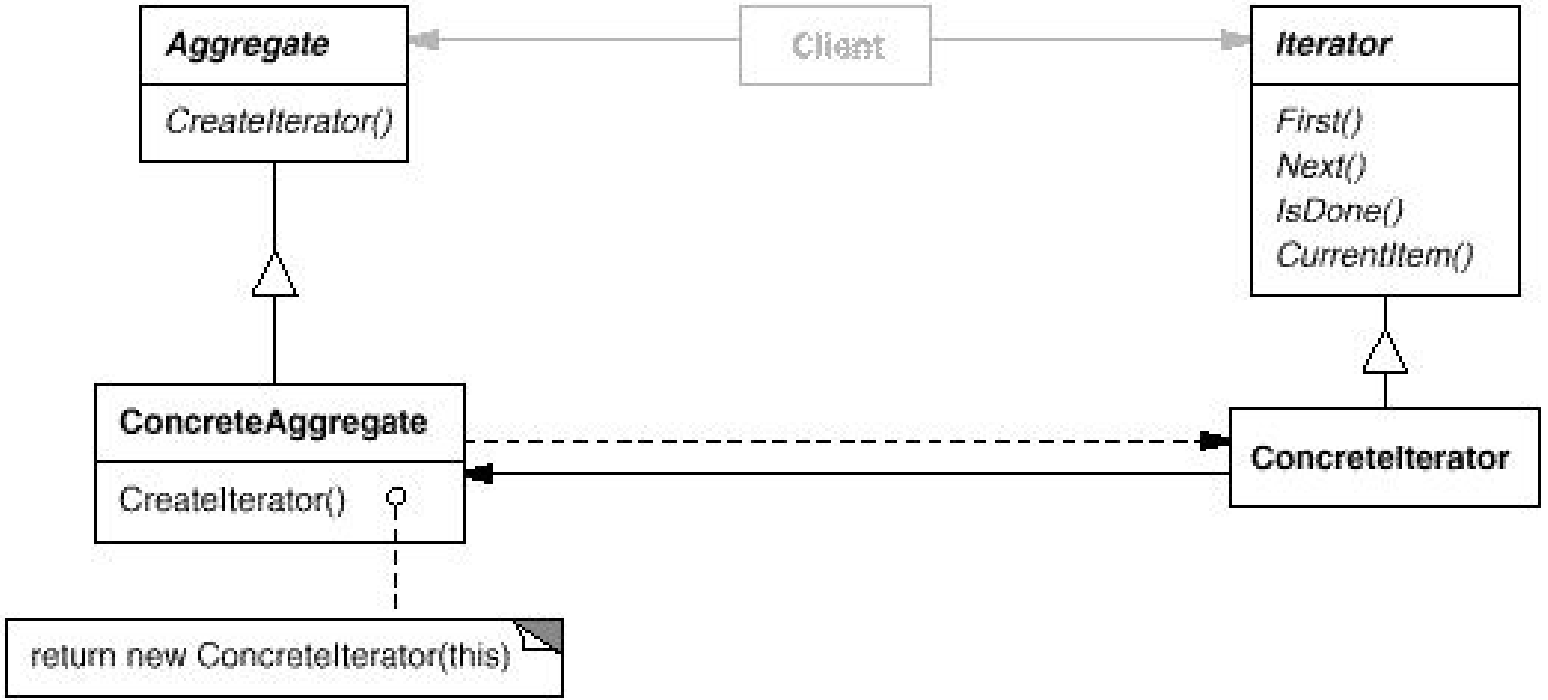
Cada tipo de archivo tiene la información distribuida de distinta manera.

Intención

Mover la responsabilidad del acceso a los ítems y su recorrido del objeto agregado al **iterador**.

Proveer distintas alternativas sin llenar de métodos la interface del objeto agregado.

Diagrama


Aggregate
CreateIterator()
Client
Iterator
First()
Next()
IsDone()
CurrentItem()
ConcreteAggregate
CreateIterator()
ConcreteIterator
return new ConcreteIterator(this)

Java

- interface **Iterable**
  - iterator<E> iterator()

- interface **iterator<E>**
  - bool hasNext()
  - E next()
  - Void remove()

- interface **Enumerable<E>**
  - bool hasMoreElements()
  - E nextElement()

```
for (Iterator<TimerTask> i = c.iterator(); i.hasNext(); )
        i.next().cancel();
```

```
for (TimerTask t : c)
        t.cancel();
```

C#

- interface **IEnumerable<T>**
  - IEnumerator<T> GetEnumerator()

- interface **IEnumerator<T>**
  - T Current
  - bool MoveNext()
  - void Reset()

```
for (int i = 0; i < fibarray.Length; i++)
{
    System.Console.WriteLine(fibarray[i]);
}

foreach (var element in fibarray)
{
    System.Console.WriteLine(element);
}
```

```
public static IEnumerable<int> Power(int number, int exponent)
{
    int result = 1;
    for (int i = 0; i < exponent; i++)
    {
        result = result * number;
        yield return result;
    }
}

foreach (int i in Power(2, 8))
{
    Console.Write("{0} ", i); // Output: 2 4 8 16 32 64 128 256
}
```

Null Iterator Trees