

# Iterator

# Patrones de diseño

---

- Problema, se implementar un método que sea capaz de recorrer un contenedor de objetos independientemente del cómo estén almacenados los objetos



```
public class ClaseX {  
  
    public void imprimirEstructura(??? estructura){  
        estructura.irAlPrimero();  
        while (estructura.quedanMas())  
            system.out.println(estructura.obtenerSiguiete());  
    }  
}
```

# Patrones de diseño



- Todos los objetos tienen que implementar

- irAlPrimero()
- quedanMas()
- obtenerSiguiente()

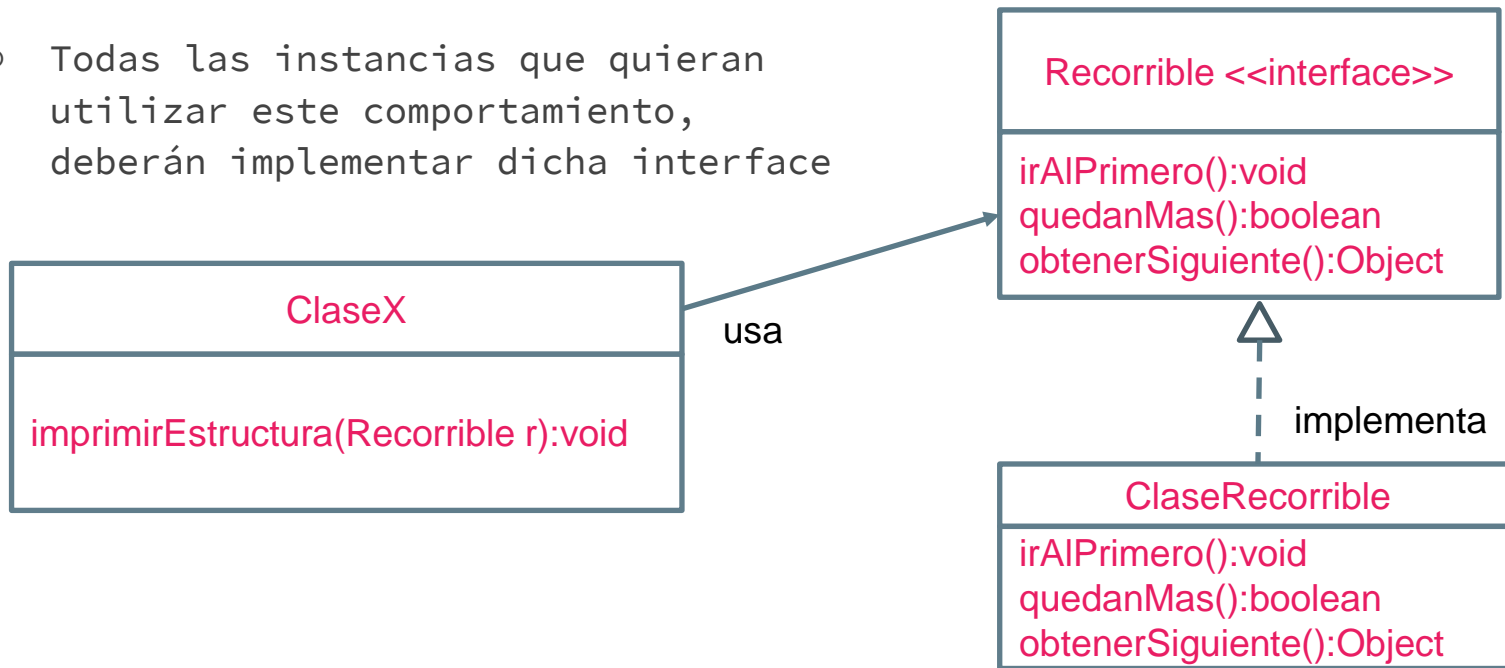
```
public class ClaseX {  
  
    public void imprimirEstructura(??? estructura){  
        estructura.irAlPrimero();  
        while (estructura.quedanMas())  
            system.out.println(estructura.obtenerSiguiente());  
    }  
}
```

- Pero... de qué tipo son los objetos que recibe el método?

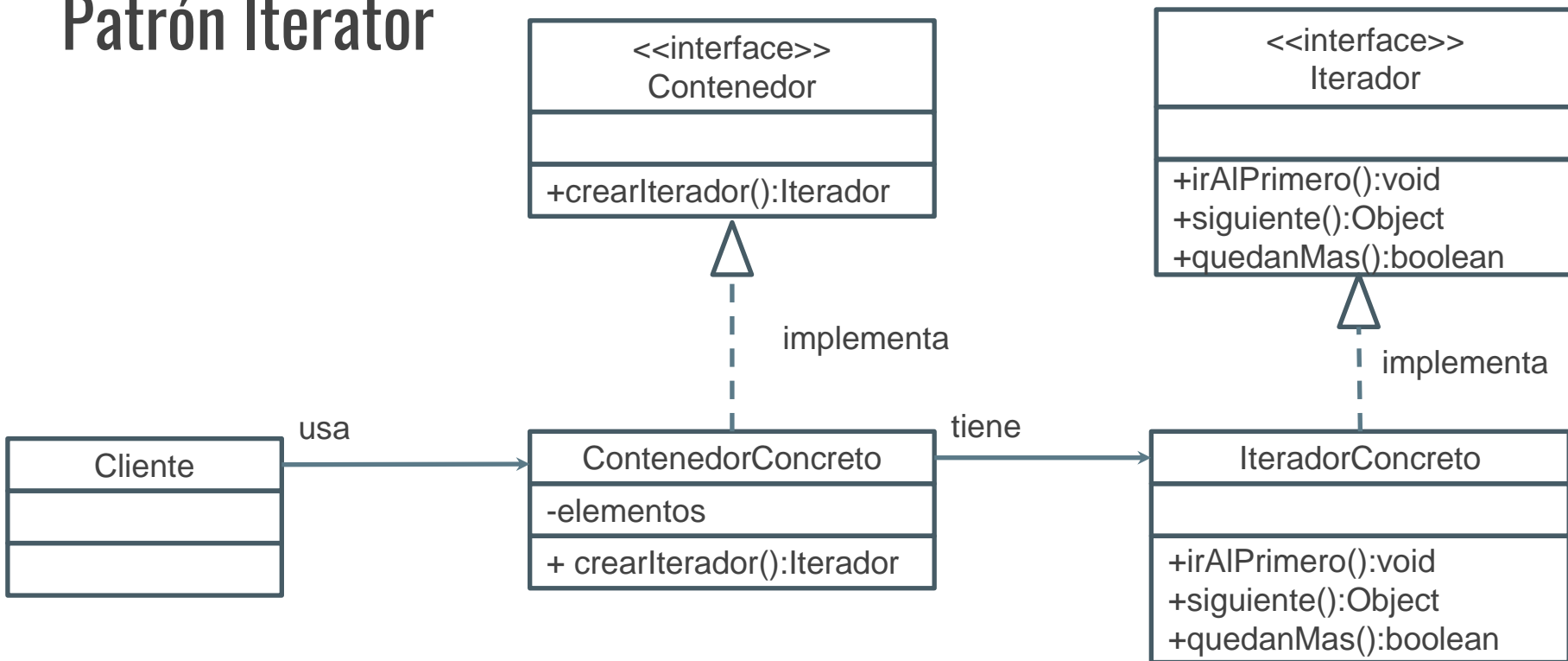
# Patrones de diseño

---

- Solución: Definir los parámetros de un tipo Interface
  - Todas las instancias que quieran utilizar este comportamiento, deberán implementar dicha interface



# Patrón Iterator

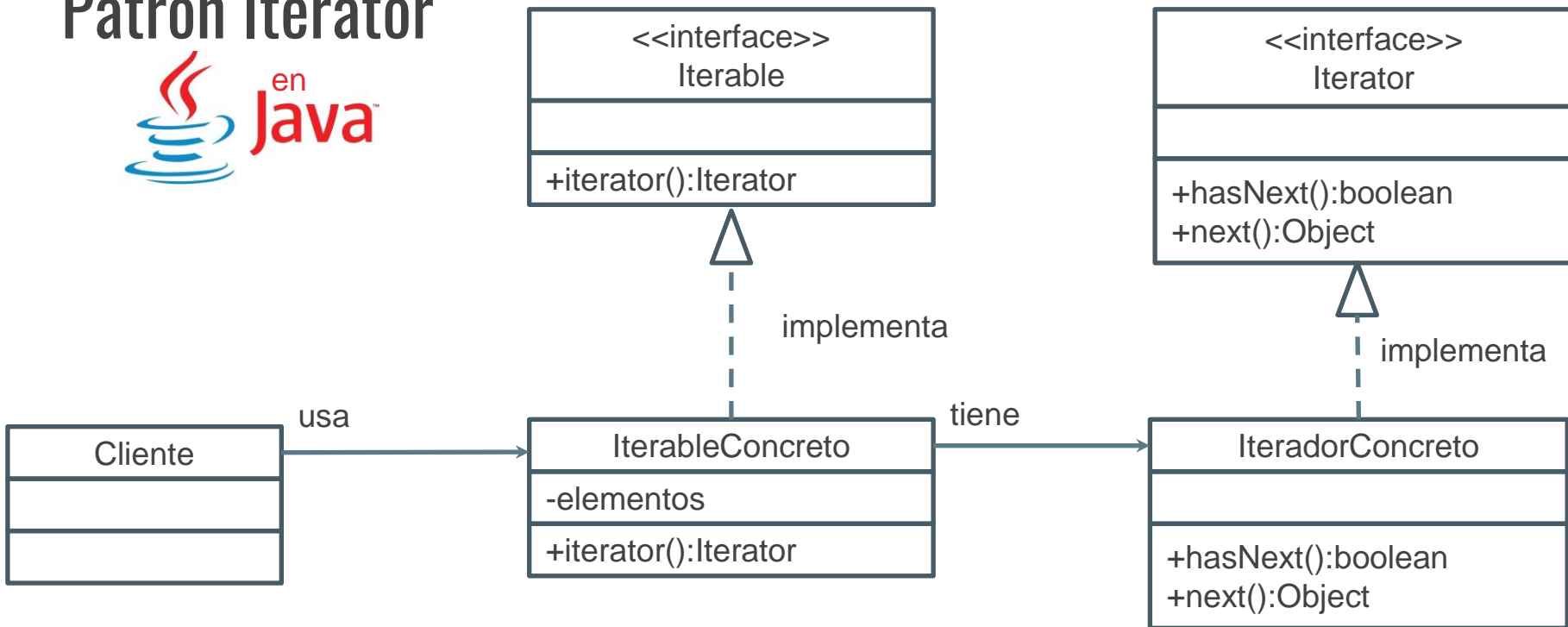


# Patrón Iterator

---

- La interfaz **Iterator** define los métodos necesarios para acceder a los elementos del contenedor y recorrerlos
- El **contenedor** define la interfaz para crear un objeto iterador.
- El **iterador concreto** implementa la interfaz del iterador y se encarga de mantener la posición actual del recorrido
- El **contenedor concreto** implementa la interfaz de creación del iterador para devolver una instancia apropiada del iterador concreto
- El **cliente** usa el contenedor concreto y le solicita el iterador cuando requiera recorrer la colección.

# Patrón Iterator



# Patrón Iterator

---

- Intención:

- proporcionar una forma de acceder a los elementos de una colección de objetos de manera secuencial, sin revelar su representación interna.
- Define una interfaz que declara métodos para acceder secuencialmente a la colección.

- Ventajas:

- acceder al contenido de una colección de elementos sin exponer su representación interna
- permitir varios recorridos sobre una colección de elementos
- proporcionar una interfaz uniforme para recorrer distintos tipos de colecciones (esto es, permitir iteración polimórfica)



# Patrón Iterator

---

- **Ejercicio:** implementar una lista vinculada de valores enteros. Una lista vinculada consiste de un nodo que contiene un valor, y una referencia al siguiente nodo de la lista. Implementar un iterador para esta lista.
- ¿Cómo cambia la implementación si la lista se debe mantener siempre ordenada ascendentemente, ante inserciones o eliminación de elementos?

# Patrón Iterator

---

- **Ejercicio:** implementar un iterador que permita ir recorriendo la sucesión de Fibonacci, durante un número N de iteraciones.  
En matemáticas, la sucesión o serie de Fibonacci es la siguiente sucesión infinita de números naturales:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597...

