**Universidad Autónoma Gabriel René Moreno**

Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones

Universidad Autónoma Gabriel René Moreno

Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones

**Patrón de diseño**

**Iterator**



**ESTUDIANTE**

Urquiza Contreras Jorge Luis…………………..214114260

**DOCENTE**

Ing. Josué Obed Veizaga Gonzales

**ASIGNATURA**

INF552 – Arquitectura de Software

**FECHA**

25/11/2019

Santa Cruz – Bolivia

# Concepto

El patrón iterador nos ofrece un mecanismo estándar, nos permite acceder de manera secuencial a los elementos de una colección. El patrón Iterator es uno de los de mayor uso. Existen muchas estructuras de datos de colección que usan este patrón como ser: Colas, pilas, listas, vectores etc...

Cada colección de datos nos debe proporcionar un iterador de datos que nos permita recorrer sus objetos y nosotros debemos asegurarnos que al hacer este patrón no se exponga su implementación.

Es decir este patrón nos permite diseñar un iterador de una colección de datos de manera que podamos acceder a los elementos de dicha colección sin exponer su estructura interna de los elementos o colección en si.

Este patrón tiene muchas variantes por lo general solo se usan los métodos hasNext y next, pero también se puede agregar variantes como los métodos first, last , currentItem etc… ya que normalmente se recorre una collecion de datos de manera secuencial pero también podríamos hacerlo desde el ultimo elemento hacia el primero o empezar a recorrer los elementos desde los extremos hacia adentro. Recordando que en las estructuras de datos hay muchas técnicas de recorridos y estas las podemos aplicar usando un iterator.

# Participantes

Los participantes del patrón son los siguientes:

* **LIST**.- Es la interfaz para crear un Iterador.
* **ITERATOR**.- Es la interfaz para acceder y recorrer de un aggregate “LIST”.
* **CARDITERATOR**. – Es la que implementa la interfaz Iterador y es la que guarda la posición actual del recorrido en cada momento para saber dónde está el recorrido.
* **CARD. -** Es la clase encargada de almacenar el tipo de la tarjeta y además el número correspondiente.
* **CARDLIST**. - implementa la interfaz de creación de los iteradores retornando una instancia del iterador concreto apropiado.

# Problema

Se desea recorrer los datos de una colección de tarjetas de todo tipo con sus respectivos números, para esto debemos usar el Iterador que hemos construido para cada colección de datos.

# Diagrama General

