|  |  |
| --- | --- |
| **Iterator** | |
| **Line** | **Code** |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18**  **19**  **20**  **21**  **22**  **23**  **24**  **25**  **26**  **27**  **28**  **29**  **30**  **31**  **32**  **33**  **34**  **35**  **36**  **37**  **38**  **39**  **40**  **41**  **42**  **43**  **44**  **45**  **46**  **47**  **48**  **49**  **50**  **51**  **52**  **53**  **54**  **55**  **56**  **57**  **58**  **59**  **60**  **61**  **62**  **63**  **64**  **65**  **66**  **67**  **68**  **69**  **70**  **71**  **72**  **73**  **74**  **75**  **76**  **77**  **78**  **79**  **80**  **81**  **82**  **83**  **84**  **85**  **86**  **87**  **88** | /\*------------------------------------------------------------------------  Provides a way to access the elements of an aggregate object sequentially  without exposing its underlying representation.  This structural code demonstrates the Iterator pattern which provides for  a way to traverse (iterate) over a collection of items without detailing  the underlying structure of the collection.  -------------------------------------------------------------------------\*/  **using** System;  **using** System.Collections.Generic;  **namespace** Iterator {  **class** Program {  **static void** Main() {  ConcreteAggregate a = **new** ConcreteAggregate();  a[0] = "Item A";  a[1] = "Item B";  a[2] = "Item C";  a[3] = "Item D";  //Create Iterator and provide aggregate  Iterator i = a.CreateIterator();//new ConcreteIterator(a);  Console.WriteLine("Iterating over collection:");  **object** item = i.First();  Console.WriteLine("isDone={0}", i.IsDone());  **while** (item != **null**) {  Console.WriteLine(item);  item = i.Next();  }  Console.WriteLine("isDone={0}", i.IsDone());  Console.ReadKey();  }  }  **abstract class** Aggregate {  **public abstract** Iterator CreateIterator();  }  **class** ConcreteAggregate : Aggregate {  **private** List<**string**> items = **new** List<**string**>();  **public override** Iterator CreateIterator() {  **return new** ConcreteIterator(**this**);  }  **public int** Count {  **get** { return items.Count; }  }  **public string this**[**int** index] {  **get** { return items[index]; }  **set** { items.Insert(index, value); }  }  }  **abstract class** Iterator {  **public abstract string** First();  **public abstract string** Next();  **public abstract bool** IsDone();  **public abstract string** CurrentItem();  }  **class** ConcreteIterator : Iterator {  **private** ConcreteAggregate aggregate;  **private int** current = 0;  **public** ConcreteIterator(ConcreteAggregate aggregate) {  **this**.aggregate = aggregate;  }  **public override string** First() {  **if** (current != 0) **throw new** Exception("Invalid First");  **return** aggregate[current++];  }  **public override string** Next() {  **return** (current < aggregate.Count) ? aggregate[current++] : **null**;  }  **public override string** CurrentItem() {  **return** aggregate[current];  }  **public override bool** IsDone() {  **return** (current >= aggregate.Count);  }  }  }  /\*=================== OUTPUT ===================  Iterating over collection:  isDone=False  Item A  Item B  Item C  Item D  isDone=True  =================================================\*/ |