|  |  |
| --- | --- |
| **IteratorExPrime** | |
| **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**  **89**  **90**  **91**  **92**  **93**  **94**  **95**  **96**  **97** | **using** System;  **using** System.Collections.Generic;  **namespace** IteratorPrimes {  **class** Program {  **static void** Main() {  //Aggregate ps = new Primes(1000);  Aggregate ps = **new** Primes2(1000);  Iterator i = ps.CreateIterator();  **uint**? p = i.First();  **while** (p != **null**) {  Console.WriteLine(p.Value);  p = i.Next();  }  Console.ReadKey();  }  }  /\*  \* In this example, the Primes does not keep the prime numbers. This is  \* a special case of Iterator called "Generator"  \*/  **abstract public class** Aggregate {  **public abstract** Iterator CreateIterator();  }  **public class** Primes : Aggregate {  **public override** Iterator CreateIterator() {  **return new** ConcreteIterator(**this**);  }  **internal uint** Max;  **public** Primes(**uint** max) {  Max = max;  }  }  **public class** Primes2 : Aggregate {  **public override** Iterator CreateIterator() {  **return new** ConcreteIterator2(**this**);  }  **static bool** isPrime(**uint** n) {  **uint** i;  **for** (i = 2; i < n; i++) **if** ((n % i) == 0) **break**;  **return** (i == n);  }  **internal** List<**uint**> primes = **new** List<**uint**>();  **public** Primes2(**uint** max) {  **for**(**uint** p = 2; p <= max; p++) {  **if** (isPrime(p)) primes.Add(p);  }  }  }  **abstract public class** Iterator {  **public abstract uint**? First();  **public abstract uint**? Next();  }  **class** ConcreteIterator : Iterator {  **static bool** isPrime(**uint** n) {  **uint** i;  **for** (i = 2; i < n; i++) **if** ((n % i) == 0) **break**;  **return** (i == n);  }  **private** Primes aggregate;  **private uint** current = 0;  **public** ConcreteIterator(Aggregate aggregate) {  **this**.aggregate = (Primes)aggregate;  }  **public override uint**? First() {  **return** Next();  }  **public override uint**? Next() {  **while** (current<aggregate.Max) {  current++;  **if** (isPrime(current)) **return** current;  }  **return null**;  }  }  **class** ConcreteIterator2 : Iterator {  **private** Primes2 aggregate;  **private int** current = 0;  **public** ConcreteIterator2(Aggregate aggregate) {  **this**.aggregate = (Primes2)aggregate;  }  **public override uint**? First() {  **if** (current != 0) **throw new** Exception("Cannot call First() now");  **if** (aggregate.primes.Count > 0) {  **return** aggregate.primes[current++];  }  **return null**;  }  **public override uint**? Next() {  **if** (current == aggregate.primes.Count) **return null**;  **return** aggregate.primes[current++];  }  }  } |