Dylan T Carlson

1161653

Lab107

10/11/17

**Client:**

/\*\*

\* This client class makes a list of ten people with random lucky numbers.

\* It uses three Iterators to traverse the list and print them out

\* in their respective categories.

\* @author dylca

\*/

public class Client {

public static void main(String[] args){

LuckyNumberList L = new LuckyNumberList();

LuckyNumber p1 = new LuckyNumber("Bob");

LuckyNumber p2 = new LuckyNumber("Dylan");

LuckyNumber p3 = new LuckyNumber("Tom");

LuckyNumber p4 = new LuckyNumber("Amie");

LuckyNumber p5 = new LuckyNumber("Joel");

LuckyNumber p6 = new LuckyNumber("Eric");

LuckyNumber p7 = new LuckyNumber("Jim");

LuckyNumber p8 = new LuckyNumber("Ron");

LuckyNumber p9 = new LuckyNumber("Joe");

LuckyNumber p10 = new LuckyNumber("Dustin");

L.addLuckyNumber(p1);

L.addLuckyNumber(p2);

L.addLuckyNumber(p3);

L.addLuckyNumber(p4);

L.addLuckyNumber(p5);

L.addLuckyNumber(p6);

L.addLuckyNumber(p7);

L.addLuckyNumber(p8);

L.addLuckyNumber(p9);

L.addLuckyNumber(p10);

/\*\*

\* The following code is copied from the alphabet client class

\* but modified for this client and classes.

\*/

Iterator<Position<LuckyNumber>> NumberIterator = L.positions().iterator();

System.out.print("=====Using NumberIterator===== \n");

while ( NumberIterator.hasNext() ){

LuckyNumber element = NumberIterator.next().getElement();

System.out.printf( "%-10s %-2d %-8s %-10s ", element.getName(), element.getLuckyNumber(), isEven(element), isPrime(element) );

System.out.print( "\n" );

}

System.out.print( "\n\n" );

Iterator<Position<LuckyNumber>> EvenNumberIterator = L.evenPositions().iterator();

System.out.print("=====Using EvenNumberIterator===== \n");

while ( EvenNumberIterator.hasNext() ){

LuckyNumber element = EvenNumberIterator.next().getElement();

System.out.printf( "%-10s %-2d %-8s %-10s ", element.getName(), element.getLuckyNumber(), isEven(element), isPrime(element) );

System.out.print( "\n" );

}

System.out.print( "\n\n" );

Iterator<Position<LuckyNumber>> PrimeNumberIterator = L.primePositions().iterator();

System.out.print("=====Using PrimeNumberIterator===== \n");

while ( PrimeNumberIterator.hasNext() ){

LuckyNumber element = PrimeNumberIterator.next().getElement();

System.out.printf( "%-10s %-2d %-8s %-10s ", element.getName(), element.getLuckyNumber(), isEven(element), isPrime(element) );

System.out.print( "\n" );

}

}

/\*\*

\* isEven is passed a LuckyNumber element and returns whether

\* it is even or odd.

\* @param e

\* @return

\*/

public static String isEven(LuckyNumber e){

int n;

n = e.getLuckyNumber();

if(n%2 == 0)

return "Even";

else

return "Odd";

}

/\*\*

\* isPrime is passed a LuckyNumber element and returns whether

\* it is prime or not prime.

\* @param e

\* @return

\*/

public static String isPrime(LuckyNumber e){

int n;

n = e.getLuckyNumber();

if(n == 0 || n==1)

return "Not Prime";

for(int i = 2; i < n; i++){

if(n%i == 0)

return "Not Prime";

}

return "Prime";

}

}

**LuckyNumber Class:**

/\*\*

\* LuckyNumber constructs a person with a random number.

\* It has get methods, but no set methods, so they can't be changed.

\*

\* @author Dylan Carlson

\*/

import java.util.Random;

public class LuckyNumber {

private String name;

private int luckyNumber;

Random rand = new Random();

//Constructor\\

public LuckyNumber( String name ){

this.name = name;

luckyNumber = rand.nextInt(10);

}

/\*\*

\* getName returns the name of an instance of LuckyNumber.

\* @return

\*/

public String getName(){

return name;

}

/\*\*

\* getLuckyNumber returns the number of an instance

\* of LuckyNumber.

\* @return

\*/

public int getLuckyNumber(){

return luckyNumber;

}

/\*\*

\* toString returns the class name, the name,

\* and the lucky number.

\* @return

\*/

public String toString(){

return getClass().getName() + ":" + name + ":" + luckyNumber;

}

/\*\*

\* equals is passed an object and returns true if it is equal

\* to the lucky number and false otherwise.

\* @param o

\* @return

\*/

public boolean equals( Object o ){

if( !(o instanceof LuckyNumber) )

return false;

LuckyNumber L = (LuckyNumber) o;

return name.equals( L.name )

&& luckyNumber == L.luckyNumber;

}

}

**LuckyNumberList Class:**

/\*\*

\* LuckyNumberList creates a list of LuckyNumbers and contains

\* 3 iterators for the list.

\* @author dylca

\*/

import java.util.NoSuchElementException;

public class LuckyNumberList {

private LinkedPositionalList<LuckyNumber> LuckyList = null;

public LuckyNumberList(){

LuckyList = new LinkedPositionalList<LuckyNumber>();

}

/\*\*

\* isEven is passed an int and returns true if it is even

\* and false otherwise.

\* @param n

\* @return

\*/

public boolean isEven(int n){

return (n%2 == 0);

}

/\*\*

\* isPrime is passed an int and returns true if it is prime,

\* and false otherwise.

\* @param n

\* @return

\*/

public boolean isPrime(int n){

if(n == 0 || n==1)

return false;

for(int i = 2; i < n; i++){

if(n%i == 0)

return false;

}

return true;

}

/\*\*

\* addLuckyNumber adds a luckynumber to the list.

\* @param Ln

\*/

public void addLuckyNumber( LuckyNumber Ln){

LuckyList.addLast(Ln);

}

/\*\*

\* toString returns all the elements of the luckylist.

\* @return

\*/

public String toString(){

String elements = "";

Iterator<LuckyNumber> listIterator = LuckyList.iterator();

while ( listIterator.hasNext() )

elements += listIterator.next() + " ";

return getClass().getName() + elements;

}

/\*\*

\* The following code was copied from the alphabet class examples

\* and modified to fit the LuckyNumberList class.

\*

\* edited by Dylan Carlson

\*/

//----- nested PositionIterator class -----

private class PositionIterator implements Iterator<Position<LuckyNumber>>{

private Position<LuckyNumber> cursor = LuckyList.first(); // position of the next element to report

private Position<LuckyNumber> recent = null; // position of last reported element

/\*\* Tests whether the iterator has a next object. \*/

@Override

public boolean hasNext( ) { return ( cursor != null ); }

/\*\* Returns the next position in the iterator. \*/

@Override

public Position<LuckyNumber> next( ) throws NoSuchElementException {

if ( cursor == null ) throw new NoSuchElementException( "nothing left " );

recent = cursor;

cursor = LuckyList.after( cursor );

return recent;

}

/\*\* Removes the element returned by most recent call to next. \*/

@Override

public void remove( ) throws IllegalStateException {

if ( recent == null ) throw new IllegalStateException( "nothing to remove" );

LuckyList.remove( recent ); // remove from outer list

recent = null; // do not allow remove again until next is called

}

} //----- end of nested PositionIterator class -----

//----- nested PositionIterable class -----

private class PositionIterable implements Iterable<Position<LuckyNumber>>{

@Override

public Iterator<Position<LuckyNumber>> iterator( ) { return new PositionIterator( ); }

} //----- end of nested PositionIterable class -----

/\*\* Returns an iterable representation of the list's positions.

\* @return \*/

public Iterable<Position<LuckyNumber>> positions( ) {

return new PositionIterable( ); // create a new instace of the inner class

}

private class EvenNumberIterator implements Iterator<Position<LuckyNumber>>{

private Position<LuckyNumber> cursor = LuckyList.first();

private Position<LuckyNumber> recent = null;

public boolean hasNext() { return (cursor != null); }

public Position<LuckyNumber> next() throws NoSuchElementException{

if ( recent == null )

{

while ( cursor != null && !isEven( cursor.getElement().getLuckyNumber() ) )

cursor = LuckyList.after( cursor );

}

if ( cursor == null ) throw new NoSuchElementException( "nothing left " );

recent = cursor;

cursor = LuckyList.after( cursor );

// advance cursor to the next vowel

while ( cursor != null && !isEven( cursor.getElement().getLuckyNumber()) )

cursor = LuckyList.after( cursor );

return recent;

}

public void remove( ) throws IllegalStateException {

if ( recent == null ) throw new IllegalStateException( "nothing to remove" );

LuckyList.remove( recent ); // remove from outer list

recent = null; // do not allow remove again until next is called

}

}

//----- nested PositionIterable class -----

private class EvenNumberIterable implements Iterable<Position<LuckyNumber>>{

@Override

public Iterator<Position<LuckyNumber>> iterator( ) { return new EvenNumberIterator( ); }

} //----- end of nested PositionIterable class -----

/\*\* Returns an iterable representation of the list's positions.

\* @return \*/

public Iterable<Position<LuckyNumber>> evenPositions( ) {

return new EvenNumberIterable( ); // create a new instace of the inner class

}

private class PrimeNumberIterator implements Iterator<Position<LuckyNumber>>{

private Position<LuckyNumber> cursor = LuckyList.first();

private Position<LuckyNumber> recent = null;

public boolean hasNext() { return (cursor != null); }

public Position<LuckyNumber> next() throws NoSuchElementException{

if ( recent == null )

{

while ( cursor != null && !isPrime( cursor.getElement().getLuckyNumber() ) )

cursor = LuckyList.after( cursor );

}

if ( cursor == null ) throw new NoSuchElementException( "nothing left " );

recent = cursor;

cursor = LuckyList.after( cursor );

// advance cursor to the next vowel

while ( cursor != null && !isPrime( cursor.getElement().getLuckyNumber()) )

cursor = LuckyList.after( cursor );

return recent;

}

public void remove( ) throws IllegalStateException {

if ( recent == null ) throw new IllegalStateException( "nothing to remove" );

LuckyList.remove( recent ); // remove from outer list

recent = null; // do not allow remove again until next is called

}

}

//----- nested PositionIterable class -----

private class PrimeNumberIterable implements Iterable<Position<LuckyNumber>>{

@Override

public Iterator<Position<LuckyNumber>> iterator( ) { return new PrimeNumberIterator( ); }

} //----- end of nested PositionIterable class -----

/\*\* Returns an iterable representation of the list's positions.

\* @return \*/

public Iterable<Position<LuckyNumber>> primePositions( ) {

return new PrimeNumberIterable( ); // create a new instace of the inner class

}

}

**Output:**

run:

=====Using NumberIterator=====

Bob 2 Even Prime

Dylan 6 Even Not Prime

Tom 2 Even Prime

Amie 0 Even Not Prime

Joel 5 Odd Prime

Eric 3 Odd Prime

Jim 2 Even Prime

Ron 5 Odd Prime

Joe 4 Even Not Prime

Dustin 9 Odd Not Prime

=====Using EvenNumberIterator=====

Bob 2 Even Prime

Dylan 6 Even Not Prime

Tom 2 Even Prime

Amie 0 Even Not Prime

Jim 2 Even Prime

Joe 4 Even Not Prime

=====Using PrimeNumberIterator=====

Bob 2 Even Prime

Tom 2 Even Prime

Joel 5 Odd Prime

Eric 3 Odd Prime

Jim 2 Even Prime

Ron 5 Odd Prime

BUILD SUCCESSFUL (total time: 0 seconds)