**Client Class**

/\*\*

\*@version 03/03/17

\* @author Richelin Metellus

\*/

public class LuckyNumberClient {

public static void main(String[] args) {

LuckyNumberList aList = new LuckyNumberList();

aList.addLuckyNumber(new LuckyNumber("Sam"));

aList.addLuckyNumber(new LuckyNumber("Ken"));

aList.addLuckyNumber(new LuckyNumber("Laura"));

aList.addLuckyNumber(new LuckyNumber("Davis"));

aList.addLuckyNumber(new LuckyNumber("Samantha"));

aList.addLuckyNumber(new LuckyNumber("Jordan"));

aList.addLuckyNumber(new LuckyNumber("Sania"));

aList.addLuckyNumber(new LuckyNumber("Kenny"));

aList.addLuckyNumber(new LuckyNumber("Kara"));

aList.addLuckyNumber(new LuckyNumber("Widner"));

aList.addLuckyNumber(new LuckyNumber("Emmanuel"));

aList.addLuckyNumber(new LuckyNumber("Maria"));

Iterator<Position<LuckyNumber>> defaultListIterator = aList.positions().iterator();

Iterator<Position<LuckyNumber>> evenListIterator = aList.EvenPositions().iterator();

System.out.println("Lucky Number List Contents (printed with default Iterator)");

while(defaultListIterator.hasNext() )

{

Position<LuckyNumber> currentPosition = defaultListIterator.next();

String message = null, evenMessage = null;

if(aList.isPrime(currentPosition.getElement().getLuckyNumber()) == true)

message = "Prime";

else

message = "Not Prime";

if(aList.isEven(currentPosition.getElement().getLuckyNumber()) == true)

evenMessage = "Even";

else

evenMessage = "Odd";

System.out.printf("%-10s\t %10d \t ", currentPosition.getElement().getName(),currentPosition.getElement().getLuckyNumber());

System.out.printf("%-10s\t %-3s\n", message, evenMessage);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PrimeIterator \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Iterator<Position<LuckyNumber>> primeListIterator = aList.primePositions().iterator();

System.out.println("\n\nLucky Number List Contents (printed with prime Iterator)");

while(primeListIterator.hasNext() )

{

Position<LuckyNumber> currentPosition = primeListIterator.next();

String message;

if(aList.isPrime(currentPosition.getElement().getLuckyNumber()) == true)

message = "Prime";

else

message = "Not Prime";

System.out.printf("%-10s\t %10d \t ", currentPosition.getElement().getName(),currentPosition.getElement().getLuckyNumber());

System.out.printf("%-10s\t \n", message);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Even Iterator \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

System.out.println("\n\nLucky Number List Contents (printed with even Iterator)");

while(evenListIterator.hasNext() )

{

Position<LuckyNumber> currentPosition = evenListIterator.next();

String message = null, evenMessage = null;

if(aList.isPrime(currentPosition.getElement().getLuckyNumber()) == true)

message = "Prime";

else

message = "Not Prime";

if(aList.isEven(currentPosition.getElement().getLuckyNumber()) == true)

evenMessage = "Even";

else

evenMessage = "Odd";

System.out.printf("%-10s\t %10d \t ", currentPosition.getElement().getName(),currentPosition.getElement().getLuckyNumber());

System.out.printf("%-3s \t %-10s\n" ,evenMessage, message);

}

}

}

**LuckyNumber Class**

/\*\*

\*

\* @author Richelin Metellus

\* @version 03/03/2017

\* This class define an object that store two elements

\*/

import java.util.Random;

public class LuckyNumber {

private String name;

private int luckyNumber;

public LuckyNumber(String name)

{

this.name = name;

Random rand = new Random();

luckyNumber = rand.nextInt(10);

}

public String getName(){return name;}

public int getLuckyNumber(){return luckyNumber;}

@Override

public String toString()

{

return getClass().getName() +": "+ getName() + "\t" + getLuckyNumber();

}

@Override

public boolean equals(Object o)

{

if(!(o instanceof LuckyNumber))

return false;

LuckyNumber n = (LuckyNumber) o;

return name.equalsIgnoreCase(n.getName()) && (luckyNumber == n.luckyNumber);

}

}

**LuckyNumberList**

import java.util.NoSuchElementException;

/\*\*

\*

\* @author Richelin

\* @version 03/03/2017

\*/

public class LuckyNumberList {

private LinkedPositionalList<LuckyNumber> list; // doubly linked list that will contain objects of LuckyNumber with their positions

public LuckyNumberList()

{

list = new LinkedPositionalList();

}

/\*\*

\*

\* @param ln object to be added as last in list

\*/

public void addLuckyNumber(LuckyNumber ln)

{

list.addLast(ln);

}

private class PositionIterator implements Iterator<Position<LuckyNumber>>

{

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

private Position<LuckyNumber> recent = null;

public boolean hasNext()

{

return (cursor != null);

}

public Position<LuckyNumber> next() throws NoSuchElementException

{

if(cursor == null) throw new NoSuchElementException(" No More Position");

recent = cursor;

cursor = list.after(cursor);

return recent;

}

/\*\* remove the elements retruned by most recent call to next. \*/

public void remove() throws IllegalStateException

{

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

list.remove(recent);

recent = null;

}

}

//-----------------PositionIterator End-----------

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*Nested PositionIterable Class\*\*\*\*\*\*

private class PositionIterable implements Iterable<Position<LuckyNumber>>

{

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

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*End of Nested PostionIterableClass\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*\*

\*

\* @return an iterable representation of the list's positions.

\*/

public Iterable<Position<LuckyNumber>> positions()

{

return new PositionIterable(); // after this execute, think of cursor = position/address of first luckyNumber object.

}

/\* \* \* \* \*Customed Iterator for position of Even number \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//-----------------Nested PositionIterator class ----------

private class EvenPositionIterator implements Iterator<Position<LuckyNumber>>

{

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

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

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

public Position<LuckyNumber> next() throws NoSuchElementException

{

if(recent == null) // if at begining position of the list

{

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

cursor = list.after(cursor);

}

if (cursor == null) throw new NoSuchElementException("Nothing left to see here");

recent = cursor;

cursor = list.after(cursor);

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

cursor = list.after(cursor);

return recent;

}

public void remove() throws IllegalStateException

{

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

list.remove(recent); // remove from outer list linKedPositional List

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

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of nested EvenPositionIterator \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// ----------nested Class ----------

private class EvenPositionIterable implements Iterable<Position<LuckyNumber>>

{

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

} //------ end of EvenPositionIterable---------------

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

return new EvenPositionIterable();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of All needed for even Iterator\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

private Position<LuckyNumber> cursor = list.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 {

// On the first call to next (i.e. when recent == null) you need to //<<< new code

// advance recent until it is pointing to a vowel elemet. //<<< new code

if ( recent == null ) //<<< new code

{ //<<< new code

while ( cursor != null && !isPrime( cursor.getElement().getLuckyNumber()) ) //<<< new code

cursor = list.after( cursor ); //<<< new code

} //<<< new code

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

recent = cursor;

cursor = list.after( cursor );

// advance cursor to the next vowel

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

cursor = list.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" );

list.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 PrimePositionIterable implements Iterable<Position<LuckyNumber>>{

@Override

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

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

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

\* @return \*/

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

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

}

public boolean isEven(int num)

{

return (num % 2 == 0);

}

public boolean isPrime(int num)

{

if(num == 2 ||num == 3 ){return true;}

int maxRange = (int) Math.sqrt(num);

for(int i = 2; i <= maxRange; ++i)

if(num % i == 0)

return false;

else if (i == maxRange)

return true;

return false; // to prevent compiler whining and take care if 1 is passed as argument.

}

}

**Output**

run:

Lucky Number List Contents (printed with default Iterator)

Sam 4 Not Prime Even

Ken 1 Not Prime Odd

Laura 4 Not Prime Even

Davis 6 Not Prime Even

Samantha 3 Prime Odd

Jordan 7 Prime Odd

Sania 8 Not Prime Even

Kenny 6 Not Prime Even

Kara 9 Not Prime Odd

Widner 6 Not Prime Even

Emmanuel 5 Prime Odd

Maria 5 Prime Odd

Lucky Number List Contents (printed with prime Iterator)

Samantha 3 Prime

Jordan 7 Prime

Emmanuel 5 Prime

Maria 5 Prime

Lucky Number List Contents (printed with even Iterator)

Sam 4 Even Not Prime

Laura 4 Even Not Prime

Davis 6 Even Not Prime

Sania 8 Even Not Prime

Kenny 6 Even Not Prime

Widner 6 Even Not Prime

BUILD SUCCESSFUL (total time: 0 seconds)