/\*\*

\*

\* @author Samuel Swedberg

\* @version 9/16/22

\* @param <E>

\*

\* The Bag interface holds methods to be access by other classes

\*/

public interface Bag<E> {

/\*\*

\*

\* @return count

\*/

public int size();

/\*\*

\*

\* @return true if empty, false otherwise

\*/

public boolean isEmpty();

/\*\*

\*

\* @param e

\* @return count number of times e exists

\*/

public int getFrequencyOf(E e);

/\*\*

\*

\* @param e

\* @return checks if list contains e

\*/

public boolean contains(E e);

/\*\*

\* add e to list

\* @param e

\*/

public void add(E e);

/\*\*

\*

\* @param e

\* @return removes first occurrence of e from list

\*/

public E remove(E e);

/\*\*

\*

\* @return removes random number from list, throws if list is empty

\*/

public E remove();

/\*\*

\*

\* @param i

\* @return returns number in i index of list, throws if i is out of bounds

\*/

public E get(int i);

/\*\*

\*

\* @return contents of instance

\*/

@Override

public String toString();

/\*\*

\*

\* @param o

\* @return true if equal, false otherwise

\*/

@Override

public boolean equals(Object o);

}

import java.util.Random;

/\*\*

\*

\* @author Samuel Swedberg

\* @version 9/16/22

\*

\* The ArrayBag class holds information related to accessing and holding information using and interfaces

\*/

public class ArrayBag<E> implements Bag<E>{

private int count;

private E[] list;

/\*\*

\* Default constructor for ArrayBag

\*/

public ArrayBag()

{

list = ( E[] ) new Object[2];

}

/\*\*

\* Overloaded constructor for ArrayBag

\* @param value

\*/

public ArrayBag( int value )

{

list = ( E[] ) new Object[value];

}

/\*\*

\*

\* @return count

\*/

public int size( ) { return count; }

/\*\*

\*

\* @return true if empty, false otherwise

\*/

public boolean isEmpty( )

{

if(size() == 0)

{

return true;

}

else

{

return false;

}

}

/\*\*

\*

\* @param e

\* @return count number of times e exists

\*/

public int getFrequencyOf( E e )

{

int freqCount = 0;

for( int i=0; i<list.length; i++)

{

if( e.equals(list[i]) )

{

freqCount++;

}

}

return freqCount;

}

/\*\*

\*

\* @param e

\* @return checks if list contains e

\*/

public boolean contains( E e )

{

boolean bool = false;

for(int i=0; i<list.length; i++)

{

if( list[i] == e )

{

bool = true;

}

}

return bool;

}

/\*\*

\*

\* @param e add e to list

\*/

public void add ( E e )

{

if( size() == list.length )

{

E[] temp = ( E[] ) new Object[list.length\*2];

for(int i=0; i<list.length; i++)

{

temp[i] = list[i];

}

list = temp;

temp = null;

list[size()] = e;

count++;

}

else

{

list[size()] = e;

count++;

}

}

/\*\*

\*

\* @param e

\* @return removes first occurrence of e from list

\*/

public E remove ( E e )

{

boolean firstOccurence = false;

if( isEmpty() == false )

{

for(int i=0; i<size(); i++)

{

if(e == list[i] && firstOccurence == false)

{

firstOccurence = true;

for(int x=i; x<size()-1; x++)

{

list[x] = list[x+1];

}

list[list.length-1] = null;

--count;

}

}

}

return null;

}

/\*\*

\*

\* @return removes random number from list, throws if list is empty

\*/

public E remove ( )

{

Random random = new Random();

if ( isEmpty() == false )

{

int rand = random.nextInt(size());

for( int i=rand; i<size()-1; i++)

{

list[i] = list[i+1];

}

list[list.length-1] = null;

--count;

}

else

{

throw new IllegalStateException("IllegalStateException: The remove() method cannot be called on an empty list");

}

return null;

}

/\*\*

\*

\* @param i

\* @return returns number in i index of list, throws if i is out of bounds

\*/

public E get( int i )

{

if ( i > list.length)

{

throw new ArrayIndexOutOfBoundsException("ArrayIndexOutOfBoundsException: index is outside the bounds of the array.");

}

return list[i];

}

/\*\*

\*

\* @return contents of instance

\*/

public String toString( ) {

String s = "";

for(int i=0; i<size(); i++)

{

s += " " + list[i];

}

return getClass().getName() + "@ size:" + size() + ", " + "list: " + s;

}

/\*\*

\*

\* @param o

\* @return true if equal, false otherwise

\*/

public boolean equals( Object o )

{

if ( !( o instanceof ArrayBag ) )

return false;

ArrayBag b = ( ArrayBag ) o;

return super.equals( b )

&& list == b.list;

}

}

import java.util.Random;

/\*\*

\*

\* @author Samuel Swedberg

\* @version 9/16/22

\*

\* The LinkedBag class holds information related to accessing and holding information using SinglyLinkedLists and interfaces

\*/

public class LinkedBag<E> implements Bag<E> {

SinglyLinkedList<E> bag;

/\*\*

\* Default constructor for LinkedBag

\*/

public LinkedBag() {

bag = new SinglyLinkedList<>();

}

/\*\*

\* Overloaded constructor for LinkedBag

\* @param capacity

\*/

public LinkedBag(int capacity) {

this();

}

/\*\*

\*

\* @return count

\*/

public int size( ) { return bag.size(); }

/\*\*

\*

\* @return true if empty, false otherwise

\*/

public boolean isEmpty( ) { return bag.isEmpty(); }

/\*\*

\*

\* @param e

\* @return count number of times e exists

\*/

public int getFrequencyOf( E e )

{

int freqCount = 0;

for(int i=0; i<bag.size(); i++)

{

E temp = bag.removeFirst();

if(temp.equals(e))

{

freqCount++;

}

bag.addLast(temp);

}

return freqCount;

}

/\*\*

\*

\* @param e

\* @return checks if list contains e

\*/

public boolean contains( E e )

{

boolean bool = false;

for(int i=0; i<bag.size(); i++)

{

if( bag.equals(e) )

{

bool = true;

}

}

return bool;

}

/\*\*

\*

\* @param e add e to list

\*/

public void add ( E e ) { bag.addLast(e); }

/\*\*

\*

\* @param num

\* @return removes first occurrence of num from list

\*/

public E remove ( E e )

{

boolean firstOccurrence = false;

if( !bag.isEmpty() )

{

for(int i=0; i<bag.size()+1; i++)

{

E temp = bag.removeFirst();

if(temp.equals(e) && firstOccurrence == false)

{

firstOccurrence = true;

}

else

{

bag.addLast(temp);

}

}

}

return null;

}

/\*\*

\*

\* @return removes random number from list, throws if list is empty

\*/

public E remove ( )

{

Random random = new Random();

boolean firstOccurrence = false;

if ( !bag.isEmpty() )

{

int rand = random.nextInt(bag.size());

for(int i=0; i<bag.size()+1; i++)

{

E temp = bag.removeFirst();

if(i == rand && firstOccurrence == false)

{

firstOccurrence = true;

}

else

{

bag.addLast(temp);

}

}

}

else

{

throw new IllegalStateException("IllegalStateException: The remove() method cannot be called on an empty list");

}

return null;

}

/\*\*

\*

\* @param i

\* @return returns number in i index of list, throws if i is out of bounds

\*/

public E get( int i )

{

E getFound = null;

if ( i > bag.size())

{

throw new ArrayIndexOutOfBoundsException("ArrayIndexOutOfBoundsException: index is outside the bounds of the array.");

}

else

{

for(int j=0; j<bag.size(); j++)

{

E temp = bag.removeFirst();

if(j == i)

{

getFound = temp;

}

bag.addLast(temp);

}

}

return getFound;

}

/\*\*

\*

\* @return contents of instance

\*/

public String toString( ) {

String s = "";

for(int i=0; i<bag.size(); i++)

{

E temp = bag.removeFirst();

s += " " + temp;

bag.addLast(temp);

}

return getClass().getName() + "@ size:" + bag.size() + ", " + "list: " + s;

}

/\*\*

\*

\* @param o

\* @return true if equal, false otherwise

\*/

public boolean equals( Object o )

{

if ( !( o instanceof ArrayBag ) )

return false;

ArrayBag b = ( ArrayBag ) o;

return super.equals( b )

&& bag.size() == b.size();

}

}

/\*\*

\*

\* @author Samuel Swedberg

\* @version 9/16/22

\*

\* The Player class holds information related to a player

\*/

public class Player {

String name, positionPlayed;

int jerseyNumber;

/\*\*

\* Constructor for the PLayer class

\* @param name

\* @param positionPlayed

\* @param jerseyNumber

\*/

public Player(String name, String positionPlayed, int jerseyNumber) {

this.name = name;

this.positionPlayed = positionPlayed;

this.jerseyNumber = jerseyNumber;

}

/\*\*

\*

\* @param name

\* @return updates name instance variables

\*/

public String setName( String name ) { return this.name = name; }

/\*\*

\*

\* @return name

\*/

public String getName() { return name; }

/\*\*

\*

\* @param positionPlayed

\* @return updates positionPlayed instance variables

\*/

public String setPositionPlayed( String positionPlayed ) { return this.positionPlayed = positionPlayed; }

/\*\*

\*

\* @return positionPlayed

\*/

public String getPositionPlayed() { return positionPlayed; }

/\*\*

\*

\* @param jerseyNumber

\* @return updates jerseyNumber instance variable

\*/

public int setJerseyNumber( int jerseyNumber ) { return this.jerseyNumber = jerseyNumber; }

/\*\*

\*

\* @return jerseyNumber

\*/

public int getJerseyNumber() { return jerseyNumber; }

/\*\*

\*

\* @return contents of Player instance

\*/

public String toString(){

return getClass().getName() + "@" + name + ":" + positionPlayed + ":" + jerseyNumber;

}

/\*\*

\*

\* @param o object to be compared

\* @return true if objects equal

\*/

public boolean equals( Object o )

{

if ( !( o instanceof Player ) )

return false;

Player p = ( Player ) o;

return name.equals( p.name )

&& positionPlayed.equals( p.positionPlayed )

&& jerseyNumber == p.jerseyNumber;

}

}

/\*\*

\* SinglyLinkedList Class

\* Code Fragments 3.14, 3.15

\* from

\* Data Structures & Algorithms, 6th edition

\* by Michael T. Goodrich, Roberto Tamassia & Michael H. Goldwasser

\* Wiley 2014

\* Transcribed by

\* @author Samuel Swedberg

\* @version 9/16/22

\*/

public class SinglyLinkedList<E> {

//---- nested Node class -----

private static class Node<E> {

private E element; // reference to the element stored at this node

private Node<E> next; // reference to the subsequent node in the list

public Node(E e, Node<E> n) {

element = e;

next = n;

}

public E getElement( ) { return element; }

public Node<E> getNext( ) { return next; }

public void setNext(Node<E> n) { next = n; }

}

// instance variables of the SinglyLinkedList

private Node<E> head = null; // head node of the list (or null if empty)

private Node<E> tail = null; // last node of the list (or null if empty)

private int size = 0; // number of nodes in the lis

public SinglyLinkedList() {} // constructs an initially empty list

// access methods

public int size( ) { return size; }

public boolean isEmpty( ) { return size == 0; }

public E first( ) { // returns (but does not remove) the first element

if (isEmpty( )) return null;

return head.getElement( );

}

public E last( ) { // returns (but does not remove) the last element

if (isEmpty( )) return null;

return tail.getElement( );

}

// update methods

public void addFirst(E e) { // adds element e to the front of the list

head = new Node<>(e, head); // create and link a new node

if (size == 0)

tail = head; // special case: new node becomes tail also

size++;

}

public void addLast(E e) { // adds element e to the end of the list

Node<E> newest = new Node<>(e, null); // node will eventually be the tail

if (isEmpty( ))

head = newest; // special case: previously empty list

else

tail.setNext(newest); // new node after existing tail

tail = newest; // new node becomes the tail

size++;

}

public E removeFirst( ) { // removes and returns the first element

if (isEmpty( )) return null; // nothing to remove

E answer = head.getElement( );

head = head.getNext( ); // will become null if list had only one node

size--;

if (size == 0)

tail = null; // special case as list is now empty

return answer;

}

}

/\*\*

\*

\* @author Samuel Swedberg

\* @version 9/16/22

\*

\* A Client that demonstrates the use of arrays and SinglyLinkedLists

\*/

public class Client {

public static void main(String[] args) {

ArrayBag<Player> mensTeam = new ArrayBag<>();

// Creates instances of players

Player player1 = new Player("Joe", "Quarterback", 2);

Player player2 = new Player("Alex", "Runningback", 12);

Player player3 = new Player("Justin", "Wide Receiver", 33);

Player player4 = new Player("Hector", "Wide Receiver", 25);

Player player5 = new Player("Eddy", "Full Back", 18);

Player player6 = new Player("Dave", "Kicker", 99);

Player player7 = new Player("Lance", "Waterboy", 80);

Player player8 = new Player("Karson", "Safety", 56);

// Adds instances to mensTeam

mensTeam.add(player1);

mensTeam.add(player2);

mensTeam.add(player3);

mensTeam.add(player4);

mensTeam.add(player5);

mensTeam.add(player6);

mensTeam.add(player7);

mensTeam.add(player8);

System.out.println(mensTeam.toString()); // Prints contents of mensTeam

mensTeam.remove(); // Removes a random element from mensTeam

System.out.println(mensTeam.toString()); // Prints contents of mensTeam

Player removeLaterMens = mensTeam.get(5); // Stores value from index 5 to remove later

System.out.println("mensTeam.get(5): " + mensTeam.get(5)); // Prints what player was gotten

Player player9 = new Player("Ethan", "Cornerback", 42); // Creates another instance of a player

mensTeam.add(player9); // Adds player to mensTeam

System.out.println(mensTeam.toString()); // Prints contents of mensTeam

mensTeam.remove(removeLaterMens); // Removes player gotten earlier

System.out.println(mensTeam.toString()); // Prints contents of mensTeam

ArrayBag<String> courses = new ArrayBag<>(); // Creates new ArrayBag named courses

// Creates instances of courses for courses ArrayBag

String course1 = new String("CSCI 161");

String course2 = new String("EE 206");

String course3 = new String("MATH 129");

String course4 = new String("MATH 265");

// Adds course to courses

courses.add(course1);

courses.add(course2);

courses.add(course3);

courses.add(course4);

System.out.println(courses.toString()); // Prints contents of courses

courses.remove(); // Removes random course

System.out.println(courses.toString()); // Prints contents of courses

LinkedBag<Player> womensTeam = new LinkedBag<>(); // Creates new LinkedBag named womensTeam

// Creates instances of players

Player player11 = new Player("Kris", "Shooting Guard", 2);

Player player12 = new Player("Grace", "Point Guard", 12);

Player player13 = new Player("Elise", "Small Forward", 33);

Player player14 = new Player("Stella", "Power Forward", 25);

Player player15 = new Player("Jean", "Center", 18);

Player player16 = new Player("Ashley", "Shooting Guard", 99);

Player player17 = new Player("Skye", "Center", 80);

Player player18 = new Player("Natalie", "Point Guard", 56);

// Adds players to womensTeam

womensTeam.add(player11);

womensTeam.add(player12);

womensTeam.add(player13);

womensTeam.add(player14);

womensTeam.add(player15);

womensTeam.add(player16);

womensTeam.add(player17);

womensTeam.add(player18);

System.out.println(womensTeam.toString()); // Prints content of womensTeam

womensTeam.remove(); // Removes random node from womensTeam

System.out.println(womensTeam.toString()); // Prints content of womensTeam

Player removeLaterWomens = womensTeam.get(5); // Stores value from 5th node of womensTeam

System.out.println("womensTeam.get(5): " + womensTeam.get(5)); // Prints the value from 5th node

Player player19 = new Player("Jill", "Power Forward", 42); // Creates another player

womensTeam.add(player19); // Adds player to womensTeam

System.out.println(womensTeam.toString()); // Prints content of womensTeam

womensTeam.remove(removeLaterWomens); // Removes node saved from earlier

System.out.println(womensTeam.toString()); // Prints content of womensTeam

}

}

run:

ArrayBag@ size:8, list: Player@Joe:Quarterback:2 Player@Alex:Runningback:12 Player@Justin:Wide Receiver:33 Player@Hector:Wide Receiver:25 Player@Eddy:Full Back:18 Player@Dave:Kicker:99 Player@Lance:Waterboy:80 Player@Karson:Safety:56

ArrayBag@ size:7, list: Player@Joe:Quarterback:2 Player@Alex:Runningback:12 Player@Hector:Wide Receiver:25 Player@Eddy:Full Back:18 Player@Dave:Kicker:99 Player@Lance:Waterboy:80 Player@Karson:Safety:56

mensTeam.get(5): Player@Lance:Waterboy:80

ArrayBag@ size:8, list: Player@Joe:Quarterback:2 Player@Alex:Runningback:12 Player@Hector:Wide Receiver:25 Player@Eddy:Full Back:18 Player@Dave:Kicker:99 Player@Lance:Waterboy:80 Player@Karson:Safety:56 Player@Ethan:Cornerback:42

ArrayBag@ size:7, list: Player@Joe:Quarterback:2 Player@Alex:Runningback:12 Player@Hector:Wide Receiver:25 Player@Eddy:Full Back:18 Player@Dave:Kicker:99 Player@Karson:Safety:56 Player@Ethan:Cornerback:42

ArrayBag@ size:4, list: CSCI 161 EE 206 MATH 129 MATH 265

ArrayBag@ size:3, list: EE 206 MATH 129 MATH 265

LinkedBag@ size:8, list: Player@Kris:Shooting Guard:2 Player@Grace:Point Guard:12 Player@Elise:Small Forward:33 Player@Stella:Power Forward:25 Player@Jean:Center:18 Player@Ashley:Shooting Guard:99 Player@Skye:Center:80 Player@Natalie:Point Guard:56

LinkedBag@ size:7, list: Player@Kris:Shooting Guard:2 Player@Elise:Small Forward:33 Player@Stella:Power Forward:25 Player@Jean:Center:18 Player@Ashley:Shooting Guard:99 Player@Skye:Center:80 Player@Natalie:Point Guard:56

womensTeam.get(5): Player@Skye:Center:80

LinkedBag@ size:8, list: Player@Kris:Shooting Guard:2 Player@Elise:Small Forward:33 Player@Stella:Power Forward:25 Player@Jean:Center:18 Player@Ashley:Shooting Guard:99 Player@Skye:Center:80 Player@Natalie:Point Guard:56 Player@Jill:Power Forward:42

LinkedBag@ size:7, list: Player@Kris:Shooting Guard:2 Player@Elise:Small Forward:33 Player@Stella:Power Forward:25 Player@Jean:Center:18 Player@Ashley:Shooting Guard:99 Player@Natalie:Point Guard:56 Player@Jill:Power Forward:42

BUILD SUCCESSFUL (total time: 1 second)