Dylan T Carlson

CSCI 161

Lab103

09/13/2018

**Bag Interface:**

/\*\*

\* This interface, Bag, contains all the method signatures that

\* are implemented into ArrayBag and LinkedBag. It is made

\* generic.

\* @author dylca

\*/

public interface Bag<T> {

/\*\*

\* getCurrentSize returns the value of count.

\* @return

\*/

int getCurrentSize();

/\*\*

\* isEmpty returns true if the list is empty, and false if it isn't.

\* @return

\*/

boolean isEmpty();

/\*\*

\* removes all items from the bag

\*/

void clear();

/\*\*

\* This method adds an item into the bag.

\* @param e

\*/

void add(T e);

/\*\*

\* getFrequencyOf is passed an object and returns the count of

\* the specific object in the bag.

\* @param e

\* @return

\*/

int getFrequencyOf(T e);

/\*\*

\* contains is passed an item, and if that item is found within

\* the list, it will return true, if not, false.

\* @param num

\* @return

\*/

boolean contains(T e);

/\*\*

\* remove(E e) is passed an object, finds the first occurrence and removes it from the list,

\* and returns true if it was successful, and false otherwise.

\* @param e

\*/

boolean remove(T e);

/\*\*

\* remove() removes a random element from the list, only if it is not empty.

\* It returns the object removed.

\* @return

\*/

T remove();

/\*\*

\* toString prints all the contents of the items in the bag.

\* @return

\*/

String toString();

/\*\*

\* equals is passed two bags and is true if they are both the same, and false otherwise

\* @param o

\* @return

\*/

boolean equals( Object o);

}

**ArrayBag Class:**

/\*\*

\* This class, ArrayBag, implements the interface Bag.

\* It is made generic so it can hold any type of objects

\* in an array.

\*

\* @author Dylan Carlson

\*/

import java.util.Random;

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

private T[] list;

private int count;

//Default Constructor

public ArrayBag(){

list = (T[]) new Object[50];

count = 0;

}

//Overload constructor

public ArrayBag(int size){

list = (T[]) new Object[size];

count = 0;

}

/\*\*

\* getCurrentSize returns the count to the user.

\* @return

\*/

public int getCurrentSize(){

return count;

}

/\*\*

\* isEmpty tests if the array is empty and returns either true or false.

\* @return

\*/

public boolean isEmpty(){

if( count == 0 )

return true;

else

return false;

}

/\*\*

\* clear clears the array by setting everything to null.

\*/

public void clear(){

for(int i = 0; i < count; i++){

list[i] = null;

}

}

/\*\*

\* getFrequencyOf is passed an object and returns the

\* number of times that object is in the array.

\* @param e

\* @return

\*/

public int getFrequencyOf(T e){

int frequency = 0;

for(int i = 0; i < count; i++){

if( list[i] == e)

frequency++;

}

return frequency;

}

/\*\*

\* contains is passed an object and returns true or false

\* depending if the object is in the array.

\* @param e

\* @return

\*/

public boolean contains(T e){

for(int i = 0; i < count; i++){

if( list[i] == e)

return true;

}

return false;

}

/\*\*

\* add is passed an object and adds it to the next

\* available position in the array.

\* @param e

\*/

public void add(T e){

if (count == list.length){

T[] temp;

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

//Copying list into temp

for(int k = 0; k<count; k++){

temp[k] = list[k];

}

//Assigning reference temp to list

list = temp;

//Setting temp equal to null

temp = null;

}

list[count] = e;

count++;

}

/\*\*

\* remove(T e) is passed an object and removes it from the list.

\* It returns true or false, true if the object was in the array,

\* and false otherwise.

\* @param e

\* @return

\*/

public boolean remove(T e){

for(int i = 0; i<count; i++ ){

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

for(; i < count-1; i++){

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

}

count--;

list[count] = null;

return true;

}

}

return false;

}

/\*\*

\* remove removes a randomly selected object and returns that object.

\* @return

\*/

public T remove(){

Random rand = new Random();

int random = rand.nextInt(count);

T itemRemoved = list[random];

//Removing the item at the index

for(; random<(count-1); random++){

list[random] = list[random+1];

}

count--;

list[count] = null;

return itemRemoved;

}

/\*\*

\* get is passed an integer and returns the object

\* at that number in the list.

\* @param index

\* @return

\*/

public T get( int index ){

return list[--index];

}

/\*\*

\* toString returns all the objects in the array, and

\* the class name, as well as count.

\* @return

\*/

public String toString(){

String elementsInBag = "";

for(int i = 0; i<count; i++){

elementsInBag = elementsInBag + list[i].toString() + " ";

}

return getClass().getName() + ":" + count + ":" + elementsInBag;

}

/\*\*

\* equals is passed an object and compares it to the ArrayBag.

\* It returns true if they are equal, and false otherwise.

\* @param o

\* @return

\*/

public boolean equals( Object o){

if ( o instanceof ArrayBag ){

ArrayBag e = (ArrayBag) o;

if( e.count == count){

for (int i = 0; i < count; i++){

if( list[i] != e.list[i])

return false;

}

}

else

return false;

}

else

return false;

return true;

}

}

**LinkedBag Class:**

/\*\*

\* LinkedBag is essentially the same as Array Bag, except instead

\* of using an array, it uses a linked list. It creates this by creating

\* an instance of the SinglyLinkedList class.

\* It is generic and implements the Bag interface.

\*

\* @author dylca

\*/

import java.util.Random;

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

private SinglyLinkedList<T> list;

private int count;

//Default Constructor

public LinkedBag(){

list = new SinglyLinkedList<>();

count = 0;

}

/\*\*

\* getCurrentSize returns the count to the user.

\* @return

\*/

public int getCurrentSize(){

return count;

}

/\*\*

\* isEmpty tests if the array is empty and returns either true or false.

\* @return

\*/

public boolean isEmpty(){

if( count == 0 )

return true;

else

return false;

}

/\*\*

\* clear removes all the contents out of the list.

\*/

public void clear(){

for(int i = 0; i < count ; i++){

list.removeFirst();

}

}

/\*\*

\* getFrequencyOf is passed an object and returns the number

\* of times the object is in the list.

\* @param e

\* @return

\*/

public int getFrequencyOf(T e){

int frequency = 0;

for(int i = 0; i < count; i++){

T temp = list.removeFirst();

if (temp.equals(e))

frequency++;

list.addLast(temp);

}

return frequency;

}

/\*\*

\* contains is passed an object and returns true if the list contains

\* the object, and false otherwise.

\* @param e

\* @return

\*/

public boolean contains(T e){

boolean truth = false;

for(int i = 0; i < count; i++){

T temp = list.removeFirst();

if( temp.equals(e))

truth = true;

list.addLast(temp);

}

return truth;

}

/\*\*

\* add is passed an object and adds in to the end of the list.

\* @param e

\*/

public void add(T e){

list.addLast(e);

count++;

}

/\*\*

\* remove(T e) is passed an object and removes it from the list.

\* It returns true if it removed an object, and false otherwise.

\* @param e

\* @return

\*/

public boolean remove(T e){

boolean truth = false;

for (int i = 0; i < count; i++ ){

T temp = list.removeFirst();

if(temp.equals(e)){

truth = true;

}

else{

list.addLast(temp);

}

}

count--;

return truth;

}

/\*\*

\* remove randomly removes an object from the list, and fills

\* in the gap. It then returns the object that was removed.

\* @return

\*/

public T remove(){

Random rand = new Random();

int random = rand.nextInt(count);

T randomRemoved = null;

for (int i = 1; i < count+1; i++ ){

T temp = list.removeFirst();

if(i != random){

list.addLast(temp);

}

else{

randomRemoved = temp;

}

}

count--;

return randomRemoved;

}

/\*\*

\* get is passed an index and returns the object that is

\* at that index.

\* @param index

\* @return

\*/

public T get(int index){

T objAtIndex = null;

for (int i=1; i <= count; i++){

T temp = list.removeFirst();

if( index == i){

objAtIndex = temp;

}

list.addLast(temp);

}

return objAtIndex;

}

/\*\*

\* toString returns the class name, count, and all

\* of the objects in the list.

\* @return

\*/

public String toString(){

String elementsInBag = "";

for(int i = 0; i < count; i++){

elementsInBag = elementsInBag + ":" + list.first();

T temp = list.removeFirst();

list.addLast(temp);

}

return getClass().getName() + ":" + count + ":" + elementsInBag;

}

/\*\*

\* equals checks to see if two bags are equal. It returns

\* true if they are, and false otherwise.

\* @param o

\* @return

\*/

public boolean equals( Object o){

boolean truth = true;

if( !(o instanceof LinkedBag) )

return false;

LinkedBag e = (LinkedBag) o;

if (e.count != count )

return false;

for (int i = 0; i < count; i++){

T temp = list.removeFirst();

if ( temp.equals( e.get(++i) ) )

list.addLast(temp);

else

truth = false;

}

return truth;

}

}

**Player Class:**

/\*\*

\* Player is a class that holds information for a player.

\* It has the necessary accessor and manipulator methods.

\* @author dylca

\*/

public class Player {

private String name;

private String positionPlayed;

int jerseyNumber;

//Overload Constructor

public Player(String n, String p, int j){

name = n;

positionPlayed = p;

jerseyNumber = j;

}

//Default Constructor

public Player(){};

/\*\*

\* setName is passed a string and sets it to name.

\* @param n

\*/

public void setName(String n){

name = n;

}

//---Setters---\\

/\*\*

\* setPositionPlayed is passed a String and sets it

\* to positionPlayed.

\* @param p

\*/

public void setPositionPlayed(String p){

positionPlayed = p;

}

/\*\*

\* setJerseyNumber is passed an integer and sets

\* it to jerseyNumber.

\* @param j

\*/

public void setJerseyNumber(int j){

jerseyNumber = j;

}

//---Getters---\\

/\*\*

\* getName returns name to the user.

\* @return

\*/

public String getName(){

return name;

}

/\*\*

\* getPositionPlayed returns positionPlayed to the user.

\* @return

\*/

public String getPositionPlayed(){

return positionPlayed;

}

/\*\*

\* getJerseyNumber returns jerseyNumber to the user.

\* @return

\*/

public int getJerseyNumber(){

return jerseyNumber;

}

/\*\*

\* toString returns all the class variables of Player.

\* @return

\*/

public String toString(){

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

}

/\*\*

\* equals tests if two Players are equal to each other.

\* It returns true if they are equal, and false otherwise.

\* @param o

\* @return

\*/

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;

}

}

**Client Class:**

/\*\*

\* Client includes the Main method and does multiple test to assure

\* the correctness of the generic ArrayBag and LinkedBag classes. It uses

\* these two classes to make two lists of its own, and manipulates them

\* in multiple ways.

\*

\* @author dylca

\*/

import java.util.Scanner;

public class Client {

public static void main(String[] args){

ArrayBag footballTeam = new ArrayBag(2);

Scanner scan = new Scanner(System.in);

System.out.println("How many players would you like to enter?");

int numberOfPlayers = scan.nextInt();

//Making an array of football team.

for(int i = 0; i < numberOfPlayers; i++){

Player p = new Player();

System.out.println("\nEnter their name: ");

scan.nextLine(); //Needed otherwise it skips to the second one.

p.setName( scan.nextLine() );

System.out.println("Enter their position: ");

p.setPositionPlayed( scan.nextLine() );

System.out.println("Enter their jersey number: ");

p.setJerseyNumber( scan.nextInt() );

footballTeam.add(p);

}

System.out.println("\n\n"); //Formatting

//Displaying Contents of footballTeam

for(int i = 0; i < numberOfPlayers; i++){

int h = i+1;

System.out.println( footballTeam.get(h) );

}

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

//-----Removing a random player-----

footballTeam.remove();

//Re - Displaying of Contents of footballTeam

for(int i = 0; i < numberOfPlayers; i++){

int h = i+1;

System.out.println( footballTeam.get(h) );

}

System.out.println("\n\n"); //Formatting

//-----Adding a new player with made up info-----

Player newPlayer = new Player("Dylan", "QB", 44);

footballTeam.add( newPlayer );

//Re - Displaying of Contents of footballTeam

for(int i = 0; i < numberOfPlayers; i++){

int h = i+1;

System.out.println( footballTeam.get(h) );

}

System.out.println("\n\n"); //Formatting

//-----Removing the new player-----

footballTeam.remove( newPlayer );

//Re - Displaying of Contents of footballTeam

for(int i = 0; i < numberOfPlayers; i++){

int h = i+1;

System.out.println( footballTeam.get(h) );

}

System.out.println("\n\n"); //Formatting

//ArrayBag of my course ids are implemented

ArrayBag courses = new ArrayBag(5);

courses.add("CSCI 161");

courses.add("CSCI 222");

courses.add("MATH 265");

courses.add("EE 206");

courses.add("STEM 303");

//Displaying information

for(int i = 0; i < 5; i++){

int h = i+1;

System.out.println( courses.get(h) );

}

System.out.println("\n\n"); //Formatting

//-----Removing a random course id-----

courses.remove();

//Re - Displaying the infofmation for courses

for(int i = 0; i < 5; i++){

int h = i+1;

System.out.println( courses.get(h) );

}

//--------Start of LinkedBag Portion--------\\

LinkedBag basketballTeam = new LinkedBag();

System.out.println("\n\n"); //Formatting

System.out.println("How many players would you like to enter?");

int numberOfBasketballPlayers = scan.nextInt();

//Making array basketball team.

for(int i = 0; i < numberOfBasketballPlayers; i++){

Player p = new Player();

System.out.println("\nEnter their name: ");

scan.nextLine(); //Needed otherwise it skips to the second one.

p.setName( scan.nextLine() );

System.out.println("Enter their position: ");

p.setPositionPlayed( scan.nextLine() );

System.out.println("Enter their jersey number: ");

p.setJerseyNumber( scan.nextInt() );

basketballTeam.add(p);

}

System.out.println("\n\n"); //Formatting

//Displaying Contents of basketballTeam

for(int i = numberOfBasketballPlayers; i > 0; i--){

System.out.println( basketballTeam.get(i) );

}

System.out.println("\n\n"); //Formatting

//-----Removing a random player-----

basketballTeam.remove();

//Re - Displaying Contents of basketballTeam

for(int i = numberOfBasketballPlayers; i > 0; i--){

System.out.println( basketballTeam.get(i) );

}

System.out.println("\n\n"); //Formatting

//-----Adding a new player with random info-----

Player newPlayer2 = new Player("Dylan", "QB", 44);

basketballTeam.add( newPlayer2 );

//Re - Displaying Contents of basketballTeam

for(int i = numberOfBasketballPlayers; i > 0; i--){

System.out.println( basketballTeam.get(i) );

}

System.out.println("\n\n"); //Formatting

//-----Removing the new player-----

basketballTeam.remove( newPlayer2 );

//Re - Displaying Contents of basketballTeam

for(int i = numberOfBasketballPlayers; i > 0; i--){

System.out.println( basketballTeam.get(i) );

}

}

}

**Output:**

run:

How many players would you like to enter?

6

Enter their name:

Bob

Enter their position:

QB

Enter their jersey number:

1

Enter their name:

Tom

Enter their position:

Saftey

Enter their jersey number:

2

Enter their name:

Billy

Enter their position:

Tackle

Enter their jersey number:

3

Enter their name:

James

Enter their position:

Guard

Enter their jersey number:

4

Enter their name:

Dustin

Enter their position:

Running Back

Enter their jersey number:

5

Enter their name:

Joel

Enter their position:

Tight End

Enter their jersey number:

6

Player: Bob : QB : 1

Player: Tom : Saftey : 2

Player: Billy : Tackle : 3

Player: James : Guard : 4

Player: Dustin : Running Back : 5

Player: Joel : Tight End : 6

Player: Bob : QB : 1

Player: Billy : Tackle : 3

Player: James : Guard : 4

Player: Dustin : Running Back : 5

Player: Joel : Tight End : 6

null

Player: Bob : QB : 1

Player: Billy : Tackle : 3

Player: James : Guard : 4

Player: Dustin : Running Back : 5

Player: Joel : Tight End : 6

Player: Dylan : QB : 44

Player: Bob : QB : 1

Player: Billy : Tackle : 3

Player: James : Guard : 4

Player: Dustin : Running Back : 5

Player: Joel : Tight End : 6

null

CSCI 161

CSCI 222

MATH 265

EE 206

STEM 303

CSCI 161

CSCI 222

MATH 265

STEM 303

null

How many players would you like to enter?

6

Enter their name:

Timmy

Enter their position:

Center

Enter their jersey number:

7

Enter their name:

Joe

Enter their position:

Forward

Enter their jersey number:

8

Enter their name:

Jimmy

Enter their position:

Center

Enter their jersey number:

9

Enter their name:

Eric

Enter their position:

Forward

Enter their jersey number:

10

Enter their name:

Ian

Enter their position:

Point Guard

Enter their jersey number:

11

Enter their name:

Amie

Enter their position:

Power Forward

Enter their jersey number:

12

Player: Amie : Power Forward : 12

Player: Ian : Point Guard : 11

Player: Eric : Forward : 10

Player: Jimmy : Center : 9

Player: Joe : Forward : 8

Player: Timmy : Center : 7

null

Player: Amie : Power Forward : 12

Player: Ian : Point Guard : 11

Player: Jimmy : Center : 9

Player: Joe : Forward : 8

Player: Timmy : Center : 7

Player: Dylan : QB : 44

Player: Amie : Power Forward : 12

Player: Ian : Point Guard : 11

Player: Jimmy : Center : 9

Player: Joe : Forward : 8

Player: Timmy : Center : 7

null

Player: Amie : Power Forward : 12

Player: Ian : Point Guard : 11

Player: Jimmy : Center : 9

Player: Joe : Forward : 8

Player: Timmy : Center : 7

BUILD SUCCESSFUL (total time: 2 minutes 17 seconds)