Andrew Vetter

CSCI 161

Lab 102

**Client**

import java.util.Random;

/\*\*

\* the client class that tests Scores class and bag class

\* @author Andrew Vetter

\* @version 01/28/18

\*/

public class Client {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

System.out.println("Beginning of client. Lab 102\n\n");

//create a sack of random numbers

Scores sack = new Scores(100);

System.out.println("populating sack with random numbers. ");

//polulate sack with 100 random numbers

Random rand = new Random();

for(int i = 0; i <100; i++){

//add random value from -100 to 100

sack.add(rand.nextInt(200)-100);

}

//print everything

System.out.println(sack);

sack.add(86);

System.out.println("current size of sack: "+ sack.getCurrentSize());

sack.remove();

int index75 = sack.get(75);

System.out.println("frequency of the number at the 75th index: "+ sack.getFrequencyOf(index75));

sack.remove(index75);

System.out.println("frequency of the number at the 75th index after removing one : "+ sack.getFrequencyOf(index75));

System.out.println("frequency of the number 86: "+ sack.getFrequencyOf(86));

if(sack.contains(86)){

System.out.println("we got the number 86 in there!!\n");

}else{

System.out.println("we don't have 86, sorry lads\n");

}

System.out.println("we're done for the day boys. Lab 102 finished");

}

}

**Scores class**

import java.util.Random;

/\*\*

\*

\* @author Andrew Vetter

\* @verson 01/30/18

\* The scores class implements the bag class.

\* it's a simple class that handles like the Array class

\*/

public class Scores implements Bag{

private int[] list;

private int count;

/\*\*\*

\* a basic constructor that initalize it

\*/

public Scores(){

this.list = new int[50];

count = 0;

}

/\*\*\*

\* a constructor that inits list to size

\* @param size the size of list

\*/

public Scores(int size){

this.list = new int[size];

count = 0;

}

/\*\*\*

\* gets the current size of the array

\* @return the size of the populated array

\*/

@Override

public int getCurrentSize() {

return count;

}

/\*\*\*

\*

\* @return true if empty, else false

\*/

@Override

public boolean isEmpty() {

return (count==0);

}

/\*\*\*

\* adds a number to the array

\* @param num

\*/

@Override

public void add(int num) {

//check if list is maxed out

// System.out.println("length of list" + list.length + " count: "+ count);

if(list.length == count ){

// System.out.println("in the if ");

//create temp list 2x the size of list

int[] temp = new int[count\*2];

//copy stuff from list to temp

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

temp[i] = list[i];

}

list = temp;

//clear temp

// temp = null;

}

//add num to end and increase count

list[count++] = num;

}

/\*\*\*

\* removes a random number from the array and fills the hole

\*/

@Override

public void remove() {

Random rand = new Random();

int num = rand.nextInt(count);

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

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

}

}

/\*\*\*

\* removes the first instance of num

\* @param num

\*/

@Override

public void remove(int num) {

//look for num and i

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

//see if it contains num

if(list[i]==num){

//store location of i

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

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

}

count--;

break;

}

}

}

/\*\*\*

\* returns the number at index i

\* @param i

\* @return

\*/

public int get(int i) {

if( i <= count ){

return list[i];

}else{

throw new ArrayIndexOutOfBoundsException();

}

}

/\*\*\*

\* clears out the list

\*/

@Override

public void clear() {

this.list = null;

this.count = 0;

}

/\*\*\*

\* returns the frequency of Num

\* @param num

\* @return

\*/

@Override

public int getFrequencyOf(int num) {

// check if we have values

if(count == 0){

return 0;

}

//create frequency holder

int numCount= 0;

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

//count the number of times the num is in the array

if(list[i]==num){

numCount++;

}

}

return numCount;

}

/\*\*\*

\* returns true if it contains num, otherwise false

\* @param num

\* @return

\*/

@Override

public boolean contains(int num) {

//make sure we have an array to check

if(count == 0){

return false;

}

//look for num and return true

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

//see if it contains num

if(list[i]==num){

return true;

}

}

return false;

}

/\*\*\*

\* To string method returns name of class and list of all elements in the array

\* @return

\*/

public String toString(){

String temp = new String();

temp = temp + getClass().getName() + "@";

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

temp+=list[i] + ":";

}

return temp;

}

/\*\*\*

\* equals function returns true if the function is equivelent

\* @param o

\* @return

\*/

public boolean equals(Object o){

if(!(o instanceof Scores)){

return false;

}

Scores s = (Scores) o;

if(this.count != s.count){

return false;

}

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

if(this.list[i] != s.list[i]){

return false;

}

}

return true;

}

}

**Bag interface**

/\*\*

\*

\* @author Andrew Vetter

\* @version 01/30/18

\* INterface that defines methods for Scores

\*/

public interface Bag {

int getCurrentSize();

boolean isEmpty();

void add( int num );

void remove( );

void remove(int num);

void clear( );

int getFrequencyOf( int num );

boolean contains( int num );

String toString();

boolean equals(Object o);

}

**output**

run:

Beginning of client. Lab 102

populating sack with random numbers.

Scores@15:10:-80:-50:-53:-34:67:32:73:-61:28:-7:63:49:66:26:-72:-83:-21:-56:-8:-94:78:-31:-61:30:68:86:90:-56:-21:51:-93:-27:1:57:92:34:1:45:-69:-100:-11:82:-91:99:25:81:33:13:71:-95:-16:-9:-99:-55:-84:-35:39:-47:-46:-37:57:89:-59:44:62:57:-88:-89:-82:64:16:-45:47:-55:-60:79:-85:23:66:-45:10:-55:-57:-11:-28:32:-38:-28:61:-51:-19:-95:61:55:33:-77:19:-77:

current size of sack: 101

frequency of the number at the 75th index: 3

frequency of the number at the 75th index after removing one : 2

frequency of the number 86: 2

we got the number 86 in there!!

we're done for the day boys. Lab 102 finished

BUILD SUCCESSFUL (total time: 0 seconds)