/\*\*

\*

\* @author Zhenhua Yang

\* @param <T>

\* @date 20190120

\*/

public interface Bag <T> {

public int getCurrentSize();

public boolean isEmpty();

public void add( T t );

public boolean remove( T t );

public T remove();

public void clear();

public int getFrequencyOf( T t );

public boolean contains(T t);

@Override

public String toString();

@Override

public boolean equals(Object o);

}

import java.util.Random;

/\*\*

\*

\* @author Zhenhua Yang

\* @date 20190201

\* the Scores class implements all the methods in the interface Bag

\*/

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

// create instance variables

private E[] list; // the array list stores the data of Scores

private static int count; // count stores the number of elements of Scores

// default constructor

public ArrayBag(){

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

count = 0;

}

// overloaded constructor

public ArrayBag( int size ){

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

count = 0;

}

// implement the getCurrentSize(), isEmpty() and clear() methods

@Override

public int getCurrentSize(){

return count;

}

@Override

public boolean isEmpty(){

return count == 0;

}

@Override

public void clear(){

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

list[i] = null;

}

count = 0;

}

// Implement the add (int num) method

@Override

public void add( E e ){

if(list.length > count){

list[count] = e;

count++;

}else{

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

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

temp[i] = list[i];

}

list = temp;

list[count] = e;

count++;

}

}

// Implement the getFrequencyOf (int num) method

@Override

public int getFrequencyOf( E e ){

if(contains(e)){

int countNum = 0;

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

if( list[i] == e )

countNum++;

}

return countNum;

}else{

throw new IllegalArgumentException("Cannot find the element!");

}

}

// Implement the remove (int num) and remove() methods

@Override

public boolean remove( E e ){

if(contains(e)){

int index = 0;

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

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

index = i;

break;

}

}

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

list[index] = list[index+1];

}

count--;

return true;

} else {

return false;

}

}

@Override

public E remove(){

if(count != 0){

Random rand = new Random();

int r = rand.nextInt(((count - 1) - 0) + 1) + 0;

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

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

}

count--;

return list[r];

} else {

return null;

}

}

// create the get(int i)method that returns the value in a given index position

public E get(int i){

if( i >= count ){

throw new ArrayIndexOutOfBoundsException("The index number you entered is too big!");

}else{

return list[i];

}

}

// Implement the contains(int num) method

@Override

public boolean contains(E e){

int countNum = 0;

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

if(list[i] == e)

countNum++;

}

return !(countNum==0);

}

// Implement the toString() and euqlas() method

@Override

public String toString(){

String str = "";

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

str += " " + list[i];

}

return getClass().getName() + " @ " + "count = " + count + ", list = [" + str + " ]";

}

@Override

public boolean equals(Object o){

if(!(o instanceof ArrayBag))

return false;

else{

ArrayBag s = (ArrayBag)o;

int d = 0;

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

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

d++;

}

return d == 0;

}

}

}

/\*\*

\*

\* @author aaronyang

\* @param <T>

\*/

public interface GenInterfaceIntro <T> {

public int getCount();

public int getCapacity();

public void add(T newInt) throws IllegalArgumentException;

public T get(int index) throws IllegalArgumentException, IndexOutOfBoundsException;

}

/\*\*

\*

\* @author Zhenhua Yang

\* @date 20190201

\*/

import java.util.Random;

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

private SinglyLinkedList list;

private static int count;

public LinkedBag(){

list = new SinglyLinkedList();

}

@Override

public int getCurrentSize() {

return count;

}

@Override

public boolean isEmpty() {

return count == 0;

}

@Override

public void add(E e) {

list.addLast(e);

}

@Override

public boolean remove(E e) {

E element = null;

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

element = (E) list.removeFirst();

if(element.equals(e)){

return true;

} else {

list.addLast(element);

}

}

return false;

}

@Override

public E remove() {

if(list.isEmpty()) return null;

Random rand = new Random();

int r = rand.nextInt(count);

E pointer = null;

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

E e = (E) list.removeFirst();

if( i == r ){

pointer = e;

} else {

list.addLast(e);

}

}

return pointer;

}

@Override

public void clear() {

}

@Override

public int getFrequencyOf(E e) {

E pointer = null;

int countF = 0;

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

pointer = (E) list.removeFirst();

if( e.equals(pointer) ){

countF++;

list.addLast(pointer);

} else {

list.addLast(pointer);

}

}

return countF;

}

@Override

public boolean contains(E e) {

return this.getFrequencyOf(e) == 0;

}

public String toString(){

return list.toString();

}

}

/\*\*

\* 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 joseph.latimer

\*/

public class SinglyLinkedList<E> {

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

private static class Node<E>{

private E element;

private Node<E> next;

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; }

} // --------------end of nested Node class----------

// instance variables of the SinglyLinkedList

private Node<E> head = null;

private Node<E> tail = null;

private int size = 0;

public SinglyLinkedList(){}

// access methods

public int size() { return size; }

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

public E first(){

if(isEmpty()) return null;

return head.getElement();

}

// update methods

public void addFirst(E e){

head = new Node<>(e, head); //public Node(E e, Node<E> n){ element = e; next = n;}

if(size == 0)

tail = head;

size++;

}

public void addLast(E e){ // add 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 Zhenhua Yang

\* @date 20190201

\*/

public class Player {

private String playerName;

private String positionPlayed;

private int jerseyNum;

public Player(String name, String position, int jersey){

this.playerName = name;

this.positionPlayed = position;

this.jerseyNum = jersey;

}

public String getPlayerName(){

return playerName;

}

public String getPosition(){

return positionPlayed;

}

public int getJersey(){

return jerseyNum;

}

public void setName(String newName){

playerName = newName;

}

public void setPosition(String newPosition){

positionPlayed = newPosition;

}

public void setJersey(int newJersey){

jerseyNum = newJersey;

}

public String toString(){

return this.getClass().getName() + " @ " +

playerName + " : " +

positionPlayed + " : " +

jerseyNum + " | ";

}

public boolean equals(Object o){

if(!(o instanceof Player))

return false;

else{

Player p = (Player) o;

return this.playerName.equals(p.playerName) &&

this.positionPlayed.equals(p.positionPlayed) &&

this.jerseyNum == p.jerseyNum;

}

}

}

/\*\*

\*

\* @author Zhenhua Yang

\* @date 20190201

\* this class tests the methods in the ArrayBag Object

\*/

import java.util.Random;

import java.util.Scanner;

import java.util.InputMismatchException;

public class Client {

public static void main(String[] args) {

// Create an object of ArrayBag called footballTeam

ArrayBag footballTeam = new ArrayBag(2);

String playerName = null;

String position = null;

int jerseyNum = 0;

int addNum = 1;

// create Player objects and add them to footballTeam;

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

Scanner scan = new Scanner(System.in);

System.out.println("Player's name: ");

playerName = scan.next();

while(!(playerName.matches("^[a-zA-Z]\*$"))){

System.out.println("That doesn't look like a name, try again");

playerName = scan.next();

}

System.out.println("Player's position: ");

position = scan.next();

while(!(position.matches("^[a-zA-Z]\*$"))){

System.out.println("That doesn't look like a position, try again");

position = scan.next();

}

String garbage;

System.out.println("Jersey Number:");

while(!scan.hasNextInt())

{

garbage = scan.nextLine();

System.out.println("Try again:");

}

jerseyNum = scan.nextInt();

footballTeam.add(new Player(playerName, position, jerseyNum));

}

// 2.display the contens of the Bag.

System.out.println("The contents of the Bag: \n" + footballTeam.toString());

System.out.println("");

// 3.Remove a random player from the team.

footballTeam.remove();

System.out.println("The contents of the Bag after removement: \n" + footballTeam.toString());

System.out.println("");

//5.Add a new Player with some made up information.

Player j = new Player("Juan", "QB", 23);

footballTeam.add(j);

System.out.println("Add a new player: \n" + footballTeam.toString());

System.out.println("");

// 7.Remove the Player that you just added earlier.

// Return the status if the this player is removed successfully.

boolean removed = footballTeam.remove(j);

if(removed == true){

System.out.println("Juan was removed!");

}else{

System.out.println("Didn't find the object!");

}

System.out.println("Now the bag is: \n" + footballTeam.toString());

System.out.println("");

/\*\*

\* Demonstrate that the generic class can support objects of different types

\*/

//Create an instance of a Bag called courses to store the course ids

ArrayBag courses = new ArrayBag();

// add all my courses:

courses.add("CSCI-161");

courses.add("CSCI-313");

courses.add("CSCI-716");

courses.add("CSCI-718");

courses.add("CSCI-846");

// Display the contents of the Bag.

System.out.println("All the courses: " + courses.toString());

// Remove a random course id from the Bag.

courses.remove();

// Display the contents of the Bag.

System.out.println("Remove a random course: " + courses.toString());

// basketballTeam

LinkedBag basketballTeam = new LinkedBag();

// add Players

basketballTeam.add(new Player("Rylee Nudell", "Position 1", 21));

basketballTeam.add(new Player("Emily Dietz", "Position 2", 34));

basketballTeam.add(new Player("Michelle Gaislerova", "Position 3", 22));

basketballTeam.add(new Player("Tyrah Spencer", "Position 1", 24));

basketballTeam.add(new Player("Marina Fernandez", "Position 3", 31));

basketballTeam.add(new Player("Sofija Zivaljevic", "Position 2", 4));

basketballTeam.add(new Player("Cirkeline Rimdal", "Position 1", 13));

basketballTeam.add(new Player("Danneka Voegeli", "Position 1", 44));

// Display the contents of the Bag.

System.out.println("All the Players: " + basketballTeam.toString());

// Remove a random course id from the Bag.

basketballTeam.remove();

// Display the contents of the Bag.

System.out.println("Remove a random player: " + basketballTeam.toString());

}

}