package lab04;

/\*\*

\* Generic interface that defines methods for use with a bag. The bag can hold

\* any data types, does not order the items and does not guard against

\* duplicate entries.

\* @param T Datatype of the bag

\* @author Jaden Young

\*/

public interface Bag<T> {

/\*\* Returns the current number of items in the bag \*/

public int getCurrentSize();

/\*\* Tells whether or not the bag is empty \*/

public boolean isEmpty();

/\*\* Tells whether or not the bag is full \*/

public boolean isFull();

/\*\*

\* Adds an item to the bag.

\* Returns true if the item was added successfully, false if it failed

\* @param item Item to be added to the bag

\*/

public boolean add(T item);

/\*\*

\* Removes a random item from the bag, if the bag is not empty.

\* Returns the item that was removed

\*/

public T remove();

/\*\*

\* Removes the first occurrence of the specified item in the bag.

\* Returns true if removal was successful, false if not.

\* @param item Item to be removed

\*/

public boolean remove(T item);

/\*\* Clears the bag, removing all items \*/

public void clear();

/\*\*

\* Returns the number of times a specified object appears in the bag

\* @param item Item to be searched for

\*/

public int getFrequencyOf(T item);

/\*\*

\* Returns true if specified item is present in the bag, false if not

\* @param item Item to be checked for

\*/

public boolean contains(T item);

}

package lab04;

/\*\*

\* Defines a Player object with values for name, positionPlayed, and number.

\* One constructor is provided that initializes all fields, along with get/set

\* methods for each field.

\* @author Jaden Young

\*/

public class Player {

//instance variables

private String name;

private String positionPlayed;

private int jerseyNumber;

/\*\*

\* Constructs a new Player object with values for all of the fields

\* @param xName Name of the player

\* @param xPosition Position of the player

\* @param xNumber The player's number

\*/

public Player(String xName, String xPosition, int xNumber){

name = xName;

positionPlayed = xPosition;

this.setJerseyNumber(xNumber);

}

//accessor methods---------------------------------

/\*\*

\* Returns the name of the player

\* @return Name of the player

\*/

public String getName() {

return name;

}

/\*\*

\* Returns the position of the player

\* @return Position of the player

\*/

public String getPositionPlayed() {

return positionPlayed;

}

/\*\*

\* Returns the number of the player

\* @return Player's number

\*/

public int getJerseyNumber() {

return jerseyNumber;

}

//mutator methods -------------------------

/\*\*

\* Sets the name of the player

\* @param xName New name for the player

\*/

public void setName(String xName) {

name = xName;

}

/\*\*

\* Sets the position of the player

\* @param xPosition New position for the player

\*/

public void setPositionPlayed(String xPosition) {

positionPlayed = xPosition;

}

/\*\*

\* Sets the number of the player.

\* Number must be an integer 1-99, else throws exception

\* @param xNumber

\*/

public final void setJerseyNumber(int xNumber) {

if (xNumber >= 0 && xNumber < 100)

jerseyNumber = xNumber;

else

throw new IllegalArgumentException("Number must be a "

+ "whole number 1-99");

}

/\*\*

\* Checks the contents of two Player objects for equality.<BR>

\* Returns true if objects contents are equal, false if not

\* @param xObj Object to be compared

\* @return True if objects are equal, false if not

\*/

@Override

public boolean equals(Object xObj) {

if(!(xObj instanceof Player))

return false;

Player obj = (Player)xObj;

if(!(name.equals(obj.getName())))

return false;

if(!(positionPlayed.equals(obj.getPositionPlayed())))

return false;

return jerseyNumber == obj.getJerseyNumber();

}

/\*\*

\* Returns a string representation of the data contained in the object

\* @return Single printable string of data stored in the object

\*/

@Override

public String toString() {

String output = "";

output += "Name: " + this.name;

output += "\nPosition: " + this.positionPlayed;

output += "\nJersey Number: " + this.jerseyNumber;

return output;

}

}

package lab04;

/\*\*

\* Defines a singly linked list with a specified data type.<BR>

\* Implements the Bag interface and all of its methods.

\* @author Jaden Young

\* @param <T> Data type for the bag

\*/

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

// Nested node class

private static class Node<T> {

private T element;

private Node<T> next;

public Node( T newElement, Node <T> newNext) {

element = newElement;

next = newNext;

}

// ----- Accessor Methods -------

public T getElement() {

return element;

}

public Node<T> getNext() {

return next;

}

// ----- Mutator Methods ------

public void setElement(T newElement) {

element = newElement;

}

public void setNext(Node<T> newNext) {

next = newNext;

}

}

private Node<T> bagHead = null;

private Node<T> bagTail = null;

private int count;

/\*\*

\* Creates an empty list

\*/

public LinkedBag(){

// empty list

}

/\*\* Returns the current number of nodes in the list \*/

@Override

public int getCurrentSize() {

return count;

}

/\*\* Returns true if there are no nodes in the list, false if there are \*/

@Override

public boolean isEmpty() {

return count == 0;

}

/\*\*

\* This method must be implemented from the bag interface, otherwise java

\* will have a heart attack. There's nothing to really do here, as a linked

\* list can never be "full". The method simply always returns true, and

\* does nothing else.

\* @return True. Always. The only line of the method is "return true;"

\*/

@Override

public boolean isFull() {

return true;

}

/\*\*

\* Adds a new node at the beginning of the list.

\* @param newElement Data for the node to contain

\* @return True if addition was successful (hint: it always will be)

\*/

@Override

public boolean add(T newElement) {

bagHead = new Node<>(newElement, bagHead);

if(isEmpty())

bagTail = bagHead;

count++;

return true;

}

/\*\*

\* Removes the first node in the linked list

\* @return The removed node at the head of the list

\*/

@Override

public T remove() {

if(isEmpty())

return null;

T temp = bagHead.getElement();

bagHead = bagHead.getNext();

count--;

if(isEmpty())

bagTail = null;

return temp;

}

/\*\*

\* Removes the first item that is equal to the specified item. The data in

\* the node is set to the element contained in head, and then the head is

\* removed.

\* @param item Item to be removed

\* @return True if the item was removed successfully, false if the item was

\* not removed or was not found in the list.

\*/

@Override

public boolean remove(T item) {

Node<T> temp = bagHead.getNext();

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

if(item.equals(temp.element)){

temp.setElement(bagHead.element);

bagHead = bagHead.getNext();

count--;

return true;

}

temp = temp.getNext();

}

return false;

}

/\*\* Sets head and tail equal to null, clearing the list \*/

@Override

public void clear() {

bagHead = null;

bagTail = null;

}

/\*\* Returns the number of times a specified item is present in the list \*/

@Override

public int getFrequencyOf(T item) {

Node<T> temp = bagHead;

int countOfItem = 0;

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

if(item.equals(temp.element))

countOfItem++;

temp = temp.getNext();

}

return countOfItem;

}

/\*\* Returns true if the item is in the list, false if not \*/

@Override

public boolean contains(T item) {

Node<T> temp = bagHead;

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

if(item.equals(temp.element))

return true;

temp = temp.getNext();

}

return false;

}

/\*\*

\* Returns the item stored at the specified index position.

\* @param i index of the item to be returned

\* @return Item at specified index

\* @throws IllegalArgumentException Thrown if the index is negative or if

\* the index is greater than the size of the list

\*/

public T getItem(int i) throws IllegalArgumentException {

if(i >= count)

throw new IllegalArgumentException("There aren't that many nodes"

+ " in the list");

if(i < 0)

throw new IllegalArgumentException("No such thing as a negative "

+ "index position");

Node<T> temp = bagHead;

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

temp = temp.getNext();

}

return temp.element;

}

/\*\* Returns an array of the elements held in the nodes of the linked list \*/

public T[] toArray() {

Object[] tempArray = new Object[count];

T []listArray = (T[])tempArray;

Node<T> tempNode = bagHead;

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

listArray[i] = tempNode.element;

tempNode = tempNode.getNext();

}

return listArray;

}

}

package lab04;

import java.util.Scanner;

/\*\*

\* Creates a singly linked list that represents the members of the NDSU Men's

\* Basketball team, stored as Player objects, and another singly linked list

\* that holds course ID's as strings.

\* @author Jaden Young

\*/

public class NDSUBasketBall {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

LinkedBag<Player> team = new LinkedBag();

final int NUM\_PLAYERS = 14; // magic numbers are worse than toejam

System.out.println("--------- NDSU Basketball Team Roster ---------");

for(int i = 0; i < NUM\_PLAYERS; i++) {

System.out.println("Player " + i + ":");

// get name

System.out.print("Name > ");

String name = scan.nextLine();

// get position

System.out.print("Position > ");

String position = scan.nextLine();

// get number

System.out.print("Jersey number > ");

while(!scan.hasNextInt()) {

String garbage = scan.nextLine();

System.out.print("Needs to be between 0 and 99 > ");

}

int number = scan.nextInt();

// add player to team

team.add(new Player(name, position, number));

//eats extra character left behind

scan.nextLine();

}

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

//remove the first player

team.remove();

//add a player with made up info

Player cletus = new Player("Cletus Spuckler", "4th Grade", 1);

System.out.println("Size of the team: " + team.getCurrentSize());

team.remove(cletus);

//print the entire list

int currentSize = team.getCurrentSize();

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

System.out.println("Player " + i + ":");

System.out.println(team.getItem(i).toString());

}

LinkedBag<String> courses = new LinkedBag();

courses.add("MATH 166");

courses.add("SPAN 312");

courses.add("CSCI 161");

courses.add("HNES 111");

courses.add("COMM 111");

courses.remove("CSCI 161");

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

currentSize = courses.getCurrentSize();

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

System.out.println(courses.getItem(i));

}

}

}

run:

--------- NDSU Basketball Team Roster ---------

Player 0:

Name > Brian Ishola

Position > G/F

Jersey number > 0

Player 1:

Name > Paul Miller

Position > G

Jersey number > 2

Player 2:

Name > Jake Showalter

Position > G

Jersey number > 3

Player 3:

Name > Zach Checkal

Position > G

Jersey number > 4

Player 4:

Name > Lawrence Alexander

Position > G

Jersey number > 12

Player 5:

Name > Carlin Dupree

Position > G

Jersey number > 13

Player 6:

Name > A.J. Jacobson

Position > G/F

Jersey number > 21

Player 7:

Name > Kory Brown

Position > G

Jersey number > 22

Player 8:

Name > Trey Miller

Position > G/F

Jersey number > 24

Player 9:

Name > Spencer Eliason

Position > F

Jersey number > 30

Player 10:

Name > Evan Wesenberg

Position > F

Jersey number > 32

Player 11:

Name > Chris Kading

Position > F

Jersey number > 34

Player 12:

Name > Dexter Werner

Position > F

Jersey number > 40

Player 13:

Name > Matt Kourouma

Position > G/F

Jersey number > 44

Size of the team with all players and made up player: 13

Player 0:

Name: Dexter Werner

Position: F

Jersey Number: 40

Player 1:

Name: Chris Kading

Position: F

Jersey Number: 34

Player 2:

Name: Evan Wesenberg

Position: F

Jersey Number: 32

Player 3:

Name: Spencer Eliason

Position: F

Jersey Number: 30

Player 4:

Name: Trey Miller

Position: G/F

Jersey Number: 24

Player 5:

Name: Kory Brown

Position: G

Jersey Number: 22

Player 6:

Name: A.J. Jacobson

Position: G/F

Jersey Number: 21

Player 7:

Name: Carlin Dupree

Position: G

Jersey Number: 13

Player 8:

Name: Lawrence Alexander

Position: G

Jersey Number: 12

Player 9:

Name: Zach Checkal

Position: G

Jersey Number: 4

Player 10:

Name: Jake Showalter

Position: G

Jersey Number: 3

Player 11:

Name: Paul Miller

Position: G

Jersey Number: 2

Player 12:

Name: Brian Ishola

Position: G/F

Jersey Number: 0

HNES 111

COMM 111

SPAN 312

MATH 166

BUILD SUCCESSFUL (total time: 1 minute 37 seconds)

