package lab03;

/\*\*

\*

\* @author Marcus

\* @version 9/16/16

\* @param <T>

\* \* The Bag Generic Interface

\* \* \* Calls on methods without the use of adding any bodies to the methods. Methods are abstract, meaning they can't no bodies

\* \* \* \* Methods don't do anything in Bag, but once implemented in ArrayBag, all the methods must be called and have bodies in the methods.

\*/

public interface Bag<T> {

/\*\*

\*

\* @return

\*/

int getCount();

/\*\*

\*

\* @return

\*/

boolean isEmpty();

/\*\*

\*

\* @return

\*/

boolean isFull();

/\*\*

\*

\*

\* @param num

\*/

void add(T num);

/\*\*

\*

\* @return

\*/

T remove();

/\*\*

\*

\* @param item

\*/

void remove(int item);

/\*\*

\*

\* @return

\*/

int removeAll();

/\*\*

\*

\* @return

\*/

boolean itemExists();

}

package lab03;

import com.sun.webkit.dom.CounterImpl;

import java.util.Random;

/\*\*

\*

\* @author Marcus

\* @version 9/16/16

\* @param <T>

\* \* The ArrayBag Generic Class

\* \* \* Implements Bag Generic Interface

\* \* \* \* Methods do specific things once called on(i.e. remove a random player or add specific player).

\* \* \* \* \* Provides generic methods from the Bag interface that will be called in the main class, with descriptions.

\*/

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

public T[] bag;

public int count;

/\*\*

\*

\*/

public ArrayBag() {

bag = (T[]) new Object[1];

}

/\*\*

\*

\* @param c

\*/

public ArrayBag(int c) {

for (int a = 0; a < c; a++) {

bag = (T[]) new Object[a];

}

}

/\*\*

\*

\* @return

\*/

@Override

public int getCount() {

return count;

}

/\*\*

\*

\* @return

\*/

@Override

public boolean isEmpty() {

return count == 0;

}

/\*\*

\*

\* @return

\*/

@Override

public boolean isFull() {

return count == bag.length+1;

}

/\*\*

\*

\* @param num

\*/

@Override

public void add(T num) {

if (count == bag.length) {

T[] bag2 = (T[]) new Object[bag.length \* 2];

System.arraycopy(bag, 0, bag2, 0, bag.length);

bag = bag2;

}

bag[count] = num;

count++;

}

/\*\*

\*

\* @return

\*/

@Override

public T remove() {

Random rand = new Random();

int index = rand.nextInt(count);

T number = bag[index];

for (int d = index; d < count-1; d++) {

bag[d] = bag[d + 1];

}

count--;

return number;

}

/\*\*

\*

\* @param item

\*/

@Override

public void remove(int item) {

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

if (bag[f].equals(item)) {

for (int g = f; g < count-1; g++) {

bag[g] = bag[g + 1];

}

count--;

break;

}

}

}

/\*\*

\*

\* @return

\*/

@Override

public int removeAll() {

return count = 0;

}

/\*\*

\*

\* @return

\*/

@Override

public boolean itemExists() {

for (T bag1 : bag) {

if (bag1.equals(bag)) {

return true;

}

}

return false;

}

/\*\*

\*

\* @param o

\* @return

\*/

@Override

public boolean equals(Object o) {

if (!(o instanceof ArrayBag)) {

return false;

} else {

ArrayBag s = (ArrayBag) o;

return count == s.count;

}

}

/\*\*

\*

\* @return

\*/

@Override

public String toString() {

String s = "[ ";

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

s += bag[u];

s += ", ";

}

s += "]";

return s;

}

/\*\*

\*

\* @return

\*/

public T[] arrayCopy() {

T[] temp = (T[]) new Object[bag.length];

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

temp[a] = (T) bag[a];

}

return temp;

}

/\*\*

\*

\* @param i

\* @return

\* @throws ArrayIndexOutOfBoundsException

\*/

public T get(int i) throws ArrayIndexOutOfBoundsException {

if (i <= 0 || i > count-1) {

throw new ArrayIndexOutOfBoundsException("Invalid index entry: " + i);

}

T scores = bag[i];

return scores;

}

}

package lab03;

/\*\*

\*

\* @author Marcus

\* @version 9/16/16

\* \* The Player class

\* \* \* Makes methods and a constructor for adding a player to the list.

\* \* \* \* Methods allow the player to get created or get and set the parameters in the default constructor

\* \* \* \* \* Used in NDSUBasketball Client.

\*/

public class Player {

public String name;

public String positionPlayed;

public String jerseyNumber;

/\*\*

\*

\* @param name

\* @param positionPlayed

\* @param jerseyNumber

\*/

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

this.name = name;

this.positionPlayed = positionPlayed;

this.jerseyNumber = jerseyNumber;

}

/\*\*

\*

\* @return

\*/

public String getName() {

return name;

}

/\*\*

\*

\* @return

\*/

public String getPositionPlayed() {

return positionPlayed;

}

/\*\*

\*

\* @return

\*/

public String getJerseyNumber() {

return jerseyNumber;

}

/\*\*

\*

\* @param name

\*/

public void setName(String name) {

this.name = name;

}

/\*\*

\*

\* @param positionPlayed

\*/

public void setPositionPlayed(String positionPlayed) {

this.positionPlayed = positionPlayed;

}

/\*\*

\*

\* @param jerseyNumber

\*/

public void setJerseyNumber(String jerseyNumber) {

this.jerseyNumber = jerseyNumber;

}

/\*\*

\*

\* @return

\*/

@Override

public String toString() {

return getClass().getName() + " : " + getName() + " : " + getPositionPlayed()

+ " : " + getJerseyNumber();

}

/\*\*

\*

\* @param o

\* @return

\*/

@Override

public boolean equals(Object o) {

if (!(o instanceof Player)) {

return false;

} else {

Player p = (Player) o;

return name.equalsIgnoreCase(p.name) && positionPlayed.equalsIgnoreCase(p.positionPlayed) && jerseyNumber.equalsIgnoreCase(p.jerseyNumber);

}

}

}

package lab03;

/\*\*

\*

\* @author Marcus

\* @version 9/16/16

\* \* The Courses class

\* \* \* Adds a string method to the course constructor, so you can enter the courses.

\* \* \* \* Methods allow the ability to add a String class name to an object reference/also can either get or set the name.

\* \* \* \* \* Used in NDSUBasketball Client.

\*/

public class Courses {

public String name;

/\*\*

\*

\* @param name

\*/

public Courses(String name) {

this.name = name;

}

/\*\*

\*

\* @return

\*/

public String getName()

{

return name;

}

/\*\*

\*

\* @param name

\*/

public void setName(String name)

{

this.name = name;

}

/\*\*

\*

\* @return

\*/

@Override

public String toString() {

return getName();

}

/\*\*

\*

\* @param o

\* @return

\*/

@Override

public boolean equals(Object o) {

if (!(o instanceof Courses)) {

return false;

} else {

Courses c = (Courses) o;

return name.equalsIgnoreCase(c.name);

}

}

}

package lab03;

import java.util.Scanner;

/\*\*

\*

\* @author Marcus

\* @version 9/16/16

\* \* The NDSUBasketball main client

\* \* \* Uses methods from ArrayBag, Scores and Player Class.

\*/

public class NDSUBasketball {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args)

{

ArrayBag team = new ArrayBag<>(); //calls a new object from the arraybag (explict casted) that is referrenced from <Player>

ArrayBag courses = new ArrayBag<>(); //calls a new object from the arraybag (explict casted) that is referrenced from <Courses>

Scanner s = new Scanner(System.in); //declares and inherits a new scanner variable

String name; //declares name variable

String jerseyNumber; //declares jerseyNumber variable

String positionPlayed; //declares positionPlayed variable

//creates new player variables from the player class

Player player1 = new Player("Jimmy", "Guard", "11");

Player player2 = new Player("Graham", "Guard", "8");

Player player3 = new Player("Lane", "Forward", "20");

Player player4 = new Player("Peter", "Forward", "44");

Player player5 = new Player("Center", "Guard", "54");

//adds the players to the team object

team.add(player1);

team.add(player2);

team.add(player3);

team.add(player4);

team.add(player5);

// //prints out a for loop of players(5) you would like to enter in the team object

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

// {

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

// System.out.println("Player " + (i + 1 ));

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

// name = s.nextLine();

// System.out.println("What position do you play? ");

// positionPlayed = s.nextLine();

// System.out.println("Player's Jersey Number: ");

// jerseyNumber = s.nextLine();

// System.out.println();

//

// team.add(new Player(name, positionPlayed, jerseyNumber)); //adds players to the tesm object

// }

team.remove(); //removes a random variable from the bag

team.add(new Player("Billy", "Power Forward", "50")); //hardcoded a new player to the bag

System.out.println("Count: " + team.getCount()); //get's the current ammount of players in a bag

team.remove(1); //removes 1 player, first occurence, from the bag

System.out.println("Updated count " + team.getCount()); //get's the current count after you remove the player

System.out.println("Team: " + team.toString()); //prints out everybody from the team

System.out.println();

//creates and institates new course variables

Courses course1 = new Courses("CSci 161");

Courses course2 = new Courses("Chemistry 121");

Courses course3 = new Courses("Spanish 201");

Courses course4 = new Courses("Calculus I");

//adds courses to the list

courses.add(course1);

courses.add(course2);

courses.add(course3);

courses.add(course4);

//removes a random course from the list

courses.remove();

//prints out updated list of courses

System.out.println("My courses are: " + courses.toString());

}

}

run:

Count: 5

Updated count 5

Team: [ lab03.Player : Jimmy : Guard : 11, lab03.Player : Graham : Guard : 8, lab03.Player : Lane : Forward : 20, lab03.Player : Center : Guard : 54, lab03.Player : Billy : Power Forward : 50, ]

My courses are: [ CSci 161, Spanish 201, Calculus I, ]

BUILD SUCCESSFUL (total time: 1 second)

