**Bag Interface**

/\*\*

\*

\* @author Rich

\* @param <T>

\* @version 02/02/2017

\*/

public interface Bag<T>{

/\*\*

\* returns current count of items in the bag

\* @return size of the bag

\*/

public int getCurrentSize();

/\*\*

\* check if bag is empty

\* @return true when it's empty.

\*/

public boolean isEmpty();

/\*\*

\*

\* @return true if the bag is full.

\*/

public boolean isFull();

/\*\*

\* add obj to the bag

\* @param obj object of generic type T

\* @return

\*/

public boolean add( T obj);

/\*\*

\* remove a randomly selected object from the bag.

\* @return on the removed object of type T

\*/

public T remove();

/\*\*

\*remove the first occurrence of a specific item in the bag

\* @param obj

\* @return true if item successfully removed

\*/

public boolean remove (T obj);

/\*\*

\*clear the entire content of the list

\*/

public void clear();

/\*\*

\*

\* @param obj

\* @return the frequence of occurrent of the argument

\*/

public int getFrequencyOf(T obj);

/\*\*

\*

\* @param obj

\* @return true when the argument is present in the list.

\*/

public boolean contains(T obj);

/\*\*

\*

\* @return the contents of the list

\*/

@Override

public String toString();

/\*\*

\*

\* @param o Object

\* @return true if equals to the class defined.

\*/

@Override

public boolean equals(Object o);

}

**ArrayBag Class**

import java.util.Random;

/\*\*

\*

\* @author Rich

\* @version 02/02/2017

\* @param <T> Generic

\*/

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

private T[] list;

private int count;

public ArrayBag()

{

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

}

public ArrayBag(int num)

{

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

}

@Override

public int getCurrentSize()

{

return count;

}

@Override

public boolean isEmpty()

{

if( count == 0)

return true;

else

return false;

}

@Override

public boolean isFull()

{

if(count < list.length)

return false;

else if(count == list.length)

{

try

{

T[] temp = list;

temp = (T[]) new Object[count\* 2];

}

catch(OutOfMemoryError e)

{

return true;

}

}

return false;

}

@Override

public boolean add( T obj)

{

if(!(isFull()))

{

if (count < list.length)

{

list[count] = obj;

count++ ;

}

else if ( count == list.length)

{

T[] temp = (T[]) new Object[count\*2];

int i;

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

temp[i] = list[i];

list = temp;

list[i] = obj;

count++;

temp = null;

}

System.out.println("Item successfully added to the bag");

return true; // if object successfully added to the bag.

}

return false;

}

@Override

public T remove()

{

Random rand = new Random();

int randomIndex = rand.nextInt(count);

T objAtRandomIndex = list[randomIndex];

remove(list[randomIndex]);

return objAtRandomIndex;

}

@Override

public boolean remove(T obj){

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

{

if (list[i].equals(obj))

{

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

list[j]= list[j+1];

count--;

return true;

}

}

return false;

}

@Override

public void clear()

{

count = 0;

}

@Override

public int getFrequencyOf(T obj)

{

int numOccurrence = 0;

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

{

if(list[i]== obj)

numOccurrence++;

}

return numOccurrence;

}

@Override

public boolean contains(T obj)

{

boolean flag = false;

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

{

if(list[i]== obj)

{

flag = true;

break;

}

}

return flag;

}

public T get(int index) throws ArrayIndexOutOfBoundsException{

T objAtIndex;

if (index < 0 || index >= count )

throw new ArrayIndexOutOfBoundsException(" Invalid:The Index not in between 0<=" + index +" <=" + count);

objAtIndex = list[index];

return objAtIndex;

}

public int getCurrentCapacity()

{

return list.length;

}

@Override

public String toString()

{

String listContent = "[";

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

{

listContent = listContent + list[i] + ", ";

}

return listContent + "]";

}

}

**Player Class**

/\*\*

\*

\* @author Rich

\* @version 02/03/2017

\*/

public class Player {

private String name;

private String position;

private int jerseyNumber;

public Player(String name, String position, int jerseyNumber)

{

this.name = name;

this.position = position;

this.jerseyNumber = jerseyNumber;

}

public String getName()

{

return name;

}

public String getPosition()

{

return position;

}

public int getJerseyNumber()

{

return jerseyNumber;

}

public void setName(String newName)

{

name = newName;

}

public void setPosition(String position)

{

this.position = position;

}

public void setJerseyNumber(int jerseyNumber)

{

this.jerseyNumber = jerseyNumber;

}

@Override

public String toString()

{

return "Player: " + getName() + ":" + getPosition() + ": No." + getJerseyNumber();

}

public boolean equals(Object o)

{

if (!(o instanceof Player))

return false;

Player a = (Player) o;

return a.getName().equalsIgnoreCase(name)

&& a.getPosition().equalsIgnoreCase(position) &&

(a.getJerseyNumber() == jerseyNumber);

}

}

**Client Class NDSU\_BASKETBALL**

/\*\*

\*

\* @author Richelin Metellus

\* @version 02/03/2017

\* The main test the generic type bag.

\*/

import java.util.Scanner;

public class NDSU\_BasketBall {

public static void main(String[] args) {

ArrayBag team = new ArrayBag(10);

Scanner input = new Scanner(System.in);

// declaring input var to intialize player object

String playerName, playerPosition;

int playerJersey;

int n; // array size initializer.

System.out.print("Enter the number of player(s) to initialize the team with: ");

n = input.nextInt();

input.nextLine();

for (int i = 1; i <= n; i++)

{

System.out.printf("\nEnter Player %d Name: ", i);

playerName = input.nextLine();

System.out.printf("Enter Player %d Position: ", i);

playerPosition = input.nextLine();

System.out.printf("Enter Player %d jersey number: ", i);

playerJersey = input.nextInt();

input.nextLine();

Player p = new Player(playerName, playerPosition, playerJersey);

team.add(p);

}

System.out.println("\nRemoving random player in the team " + team.remove());

System.out.println("Adding made-up player");

Player newPlayer = new Player("Josh Curry","Offense", 23);

team.add(newPlayer);

System.out.println("Current count of players in the team is : " + team.getCurrentSize());

System.out.println("Removing the made-up player added earlier ");

team.remove(newPlayer);

System.out.println("Current count after removing made-up player: " + team.getCurrentSize());

ArrayBag<String> courses = new ArrayBag(5);

System.out.println("\n Checking that the generic works with other types.\nPopulating courses id");

courses.add("CSCI 161");

courses.add("ECE 374");

courses.add("ECE 311");

courses.add("ECE 376");

courses.add("ECE 320");

System.out.println("Removing a random course id");

courses.remove();

System.out.print(courses);

}

}

**Output**

run:

Enter the number of player(s) to initialize the team with: 3

Enter Player 1 Name: Lebron James

Enter Player 1 Position: Power Forward

Enter Player 1 jersey number: 23

Item successfully added to the bag

Enter Player 2 Name: Kevin Durant

Enter Player 2 Position: Small Forward

Enter Player 2 jersey number: 14

Item successfully added to the bag

Enter Player 3 Name: Rich

Enter Player 3 Position: Metellus

Enter Player 3 jersey number: 34

Item successfully added to the bag

Removing random player in the team Player: Kevin Durant:Small Forward: No.14

Adding made-up player

Item successfully added to the bag

Current count of players in the team is : 3

Removing the made-up player added earlier

Current count after removing made-up player: 2

Checking that the generic works with other types.

Populating courses id

Item successfully added to the bag

Item successfully added to the bag

Item successfully added to the bag

Item successfully added to the bag

Item successfully added to the bag

Removing a random course id

[CSCI 161, ECE 374, ECE 311, ECE 320 ]

BUILD SUCCESSFUL (total time: 3 minutes 4 seconds)