Bag Interface:

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package lab03;

/\*\*

\* A generic Bag interface.

\* @author jacob.huesman

\*/

public interface Bag<T> {

/\*\*

\* Returns a count of the items in the bag.

\* @return A count of items in bag.

\*/

public int getCurrentSize();

/\*\*

\* Checks if the bag is empty.

\* @return true - if empty; false - otherwise

\*/

public boolean isEmpty();

/\*\*

\* Checks if the bag is full.

\* @return true - if full; false - otherwise

\*/

public boolean isFull();

/\*\*

\* Adds a new item to the bag.

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

\* @return true - if the item was added successfully; false - otherwise

\*/

public boolean add(T item);

/\*\*

\* Removes the first occurrence of the item from the bag.

\* @param item Item to be removed.

\* @return true - if item was successfully removed; false - if the item isn't in the bag

\*/

public boolean remove(T item);

/\*\*

\* Removes randomly an item from the bag as long as the bag is not empty.

\* @return The item removed from bag.

\*/

public T remove();

/\*\*

\* Creates an empty bag and replaces the old one.

\*/

public void clear();

/\*\*

\* Counts the number of times the item exists in the bag.

\* @param item The item to check for.

\* @return Number of times that item occurs in the bag.

\*/

public int getFrequencyOf(T item);

/\*\*

\* Tests whether the bag contains the item.

\* @param item The item to check for.

\* @return true - if bag contains item; false - otherwise

\*/

public boolean contains(T item);

}

ArrayBag Class:

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package lab03;

import java.util.Random;

/\*\*

\* Generic class that implements the generic Bag interface.

\* @author Jacob Huesman

\*/

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

private T[] bag;

private int count;

/\*\*

\* Default constructor initiates bag to a capacity of 50.

\*/

public ArrayBag(){

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

}

/\*\*

\* Overloaded constructor initiates bag to capacity specified.

\* @param size Initial capacity of the bag.

\*/

public ArrayBag(int size){

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

}

/\*\*

\* Returns a count of the items in the bag.

\* @return A count of items in bag.

\*/

@Override

public int getCurrentSize() {

return count;

}

/\*\*

\* Checks if the bag is empty.

\* @return true - if empty; false - otherwise

\*/

@Override

public boolean isEmpty() {

if(count <= 0){

return true;

}

return false;

}

/\*\*

\* Checks if the bag is full.

\* @return true - if full; false - otherwise

\*/

@Override

public boolean isFull() {

if(count >= bag.length){

return true;

}

return false;

}

/\*\*

\* Adds a new item to the bag.

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

\* @return true - if the item was added successfully; false - otherwise

\*/

@Override

public boolean add(T item) {

try{

if(bag.length <= count){

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

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

temp[i] = bag[i];

}

bag = temp;

temp = null;

bag[count++] = item;

} else {

bag[count++] = item;

}

return true;

} catch (Exception e){

return false;

}

}

/\*\*

\* Removes the first occurrence of the item from the bag.

\* @param item Item to be removed.

\* @return true - if item was successfully removed; false - if the item isn't in the bag

\*/

@Override

public boolean remove(T item) {

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

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

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

if(bag.length == (a+1)){

bag[a] = null;

} else {

bag[a] = bag[a+1];

}

}

count--;

return true;

}

}

return false;

}

/\*\*

\* Removes randomly an item from the bag as long as the bag is not empty.

\* @return The item removed from bag.

\*/

@Override

public T remove() {

Random rand = new Random();

if(count > 0){

int i = rand.nextInt(count);

T item = bag[i];

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

if(bag.length == (a+1)){

bag[a] = null;

} else {

bag[a] = bag[a+1];

}

}

count--;

return item;

}

return null;

}

/\*\*

\* Creates an empty bag and replaces the old one.

\*/

@Override

public void clear() {

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

}

/\*\*

\* Counts the number of times the item exists in the bag.

\* @param item The item to check for.

\* @return Number of times that item occurs in the bag.

\*/

@Override

public int getFrequencyOf(T item) {

int freq = 0;

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

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

freq++;

}

}

return freq;

}

/\*\*

\* Tests whether the bag contains the item.

\* @param item The item to check for.

\* @return true - if bag contains item; false - otherwise

\*/

@Override

public boolean contains(T item) {

if(this.getFrequencyOf(item) > 0){

return true;

}

return false;

}

/\*\*

\* Finds the item at index i and returns it.

\* @param i Index of the item desired.

\* @return The item desired.

\* @throws ArrayIndexOutOfBoundsException Throws if index specified is out of bounds.

\*/

public T get(int i) throws ArrayIndexOutOfBoundsException {

if(i >= bag.length){

throw new ArrayIndexOutOfBoundsException();

} else {

return bag[i];

}

}

/\*\*

\* Returns the current capacity of the bag.

\* @return The capacity of the bag.

\*/

public int capacity(){

return bag.length;

}

/\*\*

\* Returns a copy of the bag array instance variable.

\* @param bag Bag array instance variable to be copied.

\* @return A copy of the bag array instance variable.

\*/

public ArrayBag copy(ArrayBag bag){

ArrayBag temp = new ArrayBag(bag.getCurrentSize());

for(int i=0; i<temp.getCurrentSize(); i++){

temp.add(bag.get(i));

}

return temp;

}

}

Player:

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package lab03;

/\*\*

\*A simple representation of a basketball player.

\*

\* @author jacob.huesman

\*/

public class Player {

/\*\*

\* Declares instance variables.

\*/

private String name, position;

private int jerseyNumber;

/\*\*

\* Constructs a new Player instance.

\* @param name the name of the Player (e.g. "John Doe")

\* @param position the position played (e.g. "F")

\* @param jerseyNumber the jersey number of the player? (e.g. 1)

\*/

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

this.name = name;

this.position = position;

this.jerseyNumber = jerseyNumber;

}

/\*\*

\* Returns the name of the player.

\* @return name - current name of the player

\*/

public String getName(){

return name;

}

/\*\*

\* Returns position of the player.

\* @return position - current position of the player

\*/

public String getPosition(){

return position;

}

/\*\*

\* Returns the jersey number of the player.

\* @return jerseyNumber - current jersey number

\*/

public int getJerseyNumber(){

return jerseyNumber;

}

/\*\*

\* Changes the name of the player.

\* @param name the new player name

\*/

public void setName(String name){

this.name = name;

}

/\*\*

\* Changes the position of the player.

\* @param position the new position

\*/

public void setPosition(String position){

this.position = position;

}

/\*\*

\* Changes the jerseyNumber of the player.

\* @param jerseyNumber the new jersey number

\*/

public void setJerseyNumber(int jerseyNumber){

this.jerseyNumber = jerseyNumber;

}

/\*\*

\* Checks to see if the two Player objects are equal.

\* @param player2 The Player object to compare.

\* @return true - if they are equal; false - otherwise

\*/

public boolean equals(Player player2){

if(this.getJerseyNumber() != player2.getJerseyNumber() || this.getName() != player2.getName() || this.getPosition() != player2.getPosition()){

return false;

}

return true;

}

}

NDSUBasketBall client:

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package lab03;

import java.util.Scanner;

/\*\*

\* The NDSU-BasketBall class tests the functionality of the Bag, ArrayBag and Player classes/interfaces.

\* @author jacob.huesman

\*/

public class NDSUBasketBall {

//Doesn't need a public constructor.

private NDSUBasketBall(){}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

//Scanner object for input.

Scanner scan = new Scanner(System.in);

//Array Bag to hold the Player objects.

ArrayBag<Player> team = new ArrayBag<>();

//Prompt user to enter each Player's information.

System.out.println("Please enter team info one player at a time: ");

for(int i=0; i<team.capacity(); i++){

//Temp information holders.

String[] info = new String[2];

int jersey = 0;

//Prompt user for info.

System.out.print("Please enter the player's name: ");

info[0] = scan.nextLine();

System.out.print("Please enter the player's position: ");

info[1] = scan.nextLine();

System.out.print("Please enter the player's jersey number: ");

while(!scan.hasNextInt()){ //Ensure input is an integer.

scan.nextLine();

System.out.print("Please enter an integer: ");

}

jersey = scan.nextInt();

//Clears buffer

scan.nextLine();

//Add Player to team with collected info.

team.add(new Player(info[0], info[1], jersey));

System.out.println("Added " + info[0] + " to the team.");

//Break loop if all Players have been added.

System.out.print("Do you have additional players to enter? (y/n): ");

if(!scan.nextLine().equalsIgnoreCase("y")){break;};

}

System.out.println("All players added.");

System.out.println("");

//Remove a random Player from the Team.

if(!team.isEmpty()){

System.out.println("Randomly removing the following Player: " + team.remove().getName());

} else {

System.out.println("No players to remove!");

}

//Add a player with made up information.

Player madeUp = new Player("Jake Huesman", "Running back", 1);

team.add(madeUp);

System.out.println("Added the following player to the team: " + team.get(team.getCurrentSize()-1).getName());

//Display the current count of players in the team.

System.out.println("The current count of players on the team is: " + team.getCurrentSize());

//Remove Player added earlier with made up info.

System.out.println("Removing the following player from the team: " + madeUp.getName());

team.remove(madeUp);

//Display the current count of players in the team again.

System.out.println("The current count of players on the team is: " + team.getCurrentSize());

//Use a for loop to print the information of the Players in the team.

System.out.println("The current Players on the team are: ");

for(int i=0; i<team.getCurrentSize(); i++){

System.out.format("%-2s%30s%30s", (i+1) + ") Name: " + team.get(i).getName(), "| Postion: " + team.get(i).getPosition(), "| Jersey Number: " + team.get(i).getJerseyNumber() + "\n");

}

//Create an object of ArrayBag called courses to store the course ids of the courses taken this semester as Strings.

ArrayBag<String> courses = new ArrayBag<>(4);

//Populate the bag with the course ids.

System.out.println("\nPopulating courses with course ids.");

courses.add("Math 265");

courses.add("ECE 111");

courses.add("CSci 161");

courses.add("Math 129");

//Remove a random course id from the bag.

System.out.println("Randomly removing " + courses.remove() + " from the courses");

//Print the course ids from the bag.

System.out.println("Course ids in bag:");

for(int i=0; i<courses.getCurrentSize(); i++){

System.out.println((i+1) + ") " + courses.get(i));

}

}

}

Output:

ant -f "F:\\NetBeans Projects\\lab03" -Dnb.internal.action.name=run run

init:

Deleting: F:\NetBeans Projects\lab03\build\built-jar.properties

deps-jar:

Updating property file: F:\NetBeans Projects\lab03\build\built-jar.properties

compile:

run:

Please enter team info one player at a time:

Please enter the player's name: Jake

Please enter the player's position: Center

Please enter the player's jersey number: 4

Added Jake to the team.

Do you have additional players to enter? (y/n): y

Please enter the player's name: Jerry

Please enter the player's position: Center

Please enter the player's jersey number: 5

Added Jerry to the team.

Do you have additional players to enter? (y/n): y

Please enter the player's name: David

Please enter the player's position: Center

Please enter the player's jersey number: 6

Added David to the team.

Do you have additional players to enter? (y/n): n

All players added.

Randomly removing the following Player: Jerry

Added the following player to the team: Jake Huesman

The current count of players on the team is: 3

Removing the following player from the team: Jake Huesman

The current count of players on the team is: 2

The current Players on the team are:

1) Name: Jake | Postion: Center | Jersey Number: 4

2) Name: David | Postion: Center | Jersey Number: 6

Populating courses with course ids.

Randomly removing CSci 161 from the courses

Course ids in bag:

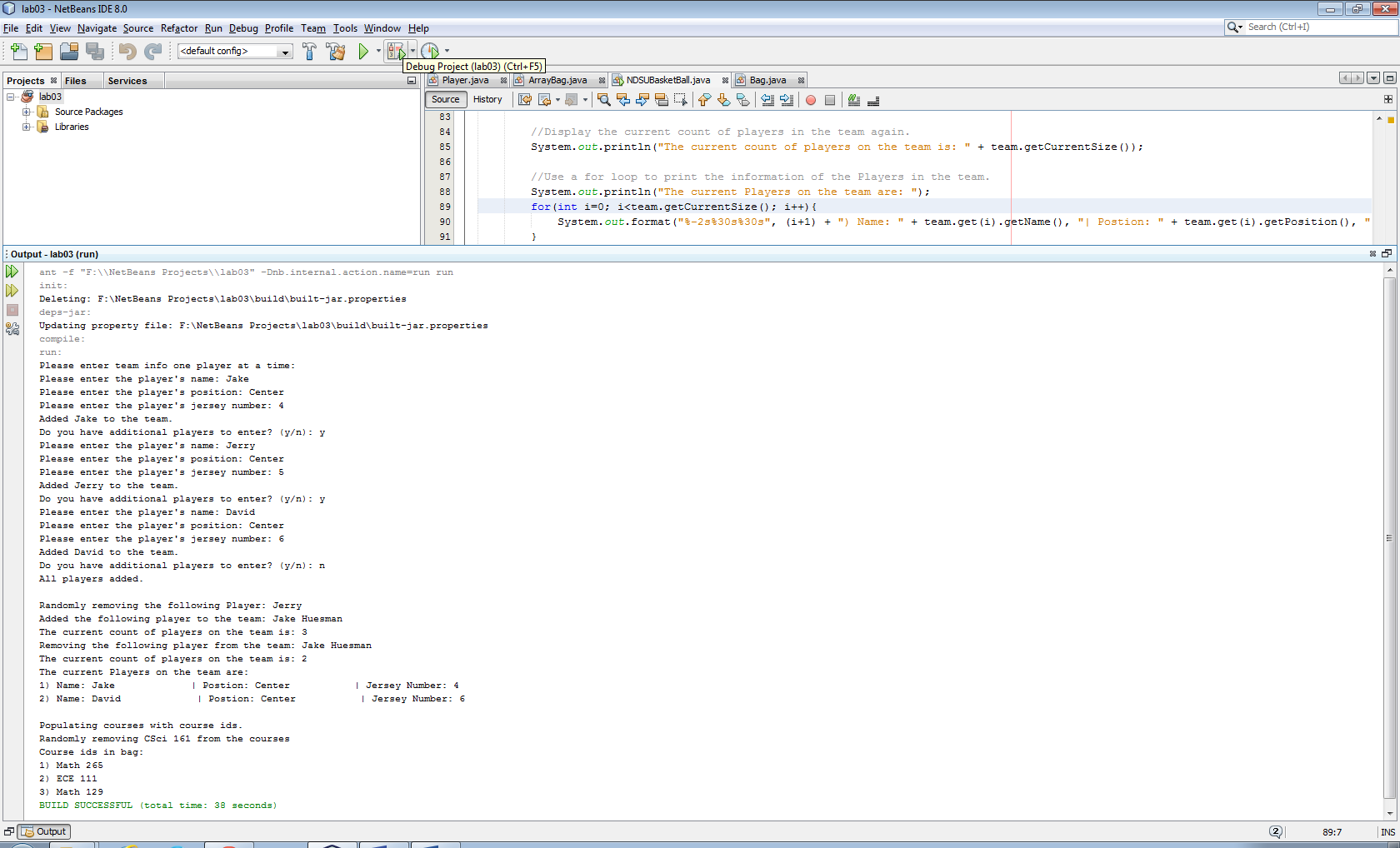
1) Math 265

2) ECE 111

3) Math 129

BUILD SUCCESSFUL (total time: 38 seconds)

Print Screen:



UML class diagram:

