Bag Interface:

package lab04;

/\*

\* 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.

\*/

/\*\*

\* 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);

}

LinkedBag:

/\*

\* 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 lab04;

/\*\*

\* Generic Singularly Linked List implementation of the generic interface Bag.

\* @author Jacob

\*/

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

/\*\*

\* The objects of this class form the nodes of the singly linked list.

\* @param <T> Type of the object the node will hold.

\*/

private static class Node<T> {

private T element;

private Node<T> next;

public Node(T e, Node<T> n){

element = e;

next = n;

}

public T getElement(){ return element; }

public Node<T> getNext() { return next; }

public void setNext( Node<T> n ){ next = n; }

}

private Node<T> bagHead;

private Node<T> bagTail;

int count;

public LinkedBag(){

bagHead = null;

bagTail = null;

count = 0;

}

/\*\*

\* 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;

}

/\*\*

\* The bag will never be full.

\* @return false - always

\*/

@Override

public boolean isFull() {

return false;

}

/\*\*

\* Adds an element to the beginning of the list.

\* @param item The item the node will point to.

\* @return true - When the operation is successful.

\*/

@Override

public boolean add(T item) {

bagHead = new Node(item, bagHead);

count++;

return true;

}

/\*\*

\* Removes and returns the first node in the list.

\* @return The element, or null if the list is empty.

\*/

@Override

public T remove() {

if(bagHead == null){

return null;

}

T element = bagHead.getElement();

bagHead = bagHead.getNext();

count--;

return element;

}

/\*\*

\* Removes a given item and returns true or false according to the success of the operation.

\* @param item Item to check the list for.

\* @return true - if item was found and removed; false - otherwise

\*/

@Override

public boolean remove(T item) {

//Priming

Node<T> previous;

Node<T> current;

if(bagHead != null && bagHead.getElement().equals(item)){

bagHead = bagHead.getNext();

count--;

return true;

}

if(bagHead != null && bagHead.next != null){

previous = bagHead;

current = bagHead.next;

} else {

return false;

}

//Loop through SLL, looking for item.

while(current.next != null){

if(current.getElement().equals(item)){

count--;

return true;

}

previous = current;

current = previous.getNext();

}

//Return false if the item was not found.

return false;

}

/\*\*

\* Clears the current bag.

\*/

@Override

public void clear() {

bagHead = null;

bagTail = null;

count = 0;

}

/\*\*

\* Checks for the frequency of a particular item.

\* @param item Item to check the frequency of.

\* @return Amount of times the object occurs in the list.

\*/

@Override

public int getFrequencyOf(T item) {

//primer

Node<T> previous;

Node<T> current;

int freq = 0;

if(bagHead != null && bagHead.getElement().equals(item)){

freq++;

}

if(bagHead != null && bagHead.next != null){

previous = bagHead;

current = bagHead.next;

} else {

return freq;

}

//Loop through SLL and count the frequency of the item.

do {

if(current.getElement().equals(item)){

previous.setNext(current.getNext());

current = current.getNext();

freq++;

}

previous = current;

current = previous.getNext();

} while(current.next != null);

//Return false if the item was not found.

return freq;

}

/\*\*

\* Checks to see if the LinkedBag contains the item.

\* @param item Item to check for.

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

\*/

@Override

public boolean contains(T item) {

//primer

Node<T> previous;

Node<T> current;

if(bagHead != null && bagHead.getElement().equals(item)){

return true;

}

if(bagHead != null && bagHead.next != null){

previous = bagHead;

current = bagHead.next;

} else {

return false;

}

//Loop through SLL and count the frequency of the item.

do {

if(current.getElement().equals(item)){

return true;

}

previous = current;

current = previous.getNext();

} while(current.next != null);

//Return false if the item was not found.

return false;

}

/\*\*

\* Returns an item at a specific index position in the bag.

\* @param i Index to grab item from.

\* @return Item collected form specified index.

\*/

public T getItem(int i){

//primer

Node<T> current;

int step = 0;

if (bagHead!= null && step == i){

return bagHead.getElement();

} else if(bagHead != null && bagHead.getNext() != null){

current = bagHead;

} else {

return null;

}

//Loop through SLL until the specified index is hit.

do {

step++;

if(step == i){

return current.getElement();

}

current = current.getNext();

} while(current.next != null);

return null;

}

/\*\*

\* Returns an array containing the copy of the items in the list.

\* @return Array of items.

\*/

public T[] toArray(){

T[] array = (T[]) new Object[count];

Node<T> current = bagHead;

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

array[i] = current.getElement();

current = current.getNext();

}

return array;

}

}

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 lab04;

/\*\*

\*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:

/\*

\* 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 lab04;

import java.util.Scanner;

/\*\*

\* The NDSU-BasketBall class tests the functionality of the Bag, LinkedBag 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);

//LinkedBag to hold the Player objects.

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

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

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

while(true){

//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 first Player from the Team.

if(!team.isEmpty()){

System.out.println("Removing the first 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", "Center", 1);

team.add(madeUp);

System.out.println("Added the following player to the team: " + team.getItem(0).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("| %1$-30s | %2$-20s | %3$-20s\n", (i+1) + ") Name: " + team.getItem(i).getName(), "Postion: " + team.getItem(i).getPosition(), "Jersey Number: " + team.getItem(i).getJerseyNumber());

}

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

LinkedBag<String> courses = new LinkedBag<>();

//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 removed " + 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.getItem(i));

}

}

}

Output:

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: John

Please enter the player's position: 342

Please enter the player's jersey number: 433

Added John 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: 423

Please enter the player's jersey number: 3

Added Jerry to the team.

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

Please enter the player's name: Mary

Please enter the player's position: 33

Please enter the player's jersey number: 4

Added Mary to the team.

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

All players added.

Removing the first Player: Mary

Added the following player to the team: Jake Huesman

The current count of players on the team is: 4

Removing the following player from the team: Jake Huesman

The current count of players on the team is: 3

The current Players on the team are:

| 1) Name: Jerry | Postion: 423 | Jersey Number: 3

| 2) Name: Jerry | Postion: 423 | Jersey Number: 3

| 3) Name: John | Postion: 342 | Jersey Number: 433

Populating courses with course ids.

Randomly removed Math 129 from the courses

Course ids in bag:

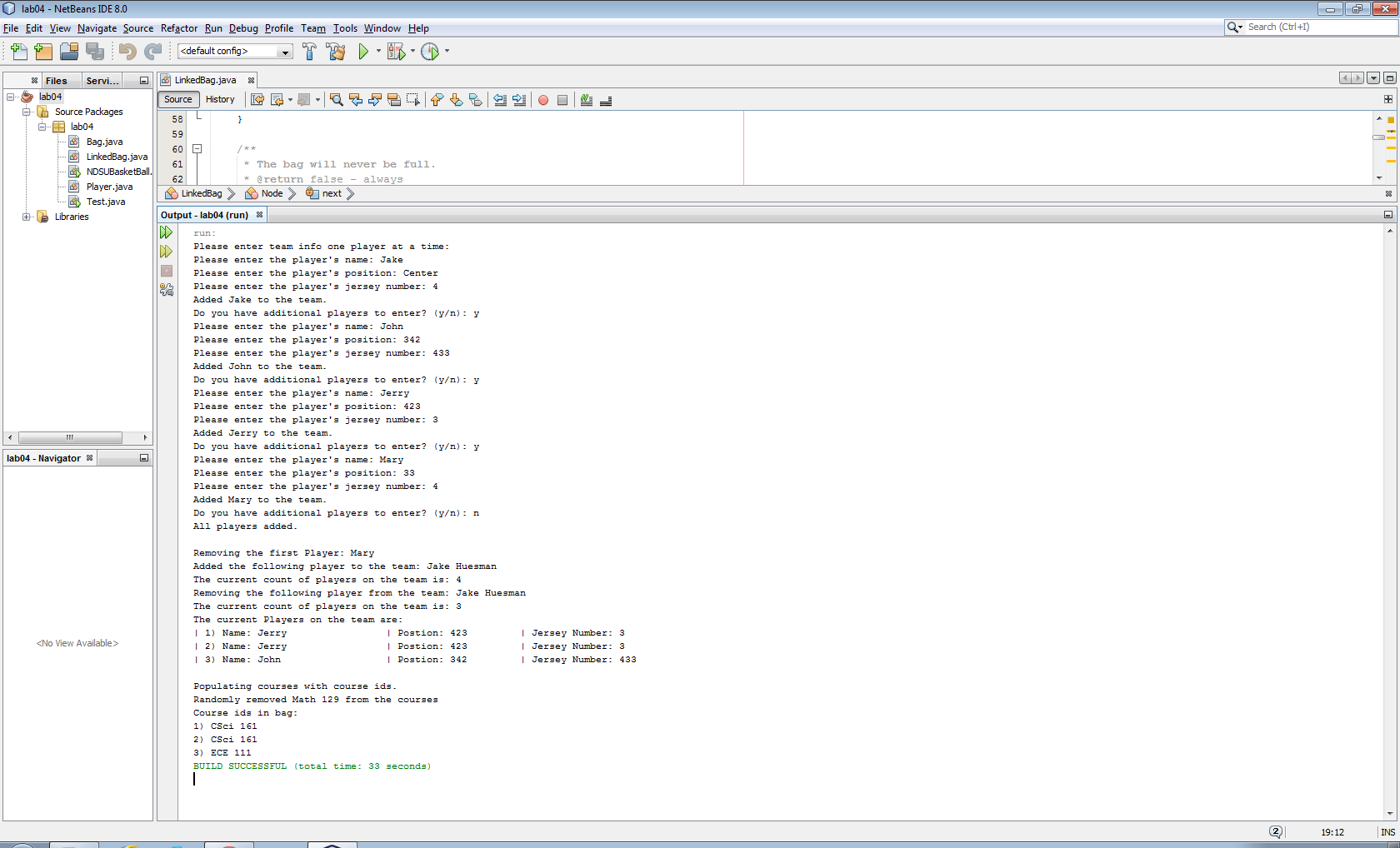
1) CSci 161

2) CSci 161

3) ECE 111

BUILD SUCCESSFUL (total time: 33 seconds)

Output:



UML Diagram:

