GROUP WORK REPORT

CSC248

Fundamentals Of Data Structure

CLASS: JCS1103C

SEMESTER: 1 OCTOBER 2021 - FEBRUARY 2022

LECTURER: NAFISAH JAMINGAN @ AMIN

NAME	MATRIX NUMBER
NADIAH NUR TASNEEM BINTI DZAKWAN	2020617654
SERI BATRISYIA BINTI MOHD KHALIL	2020460698

TABLE OF CONTENT

Introduction of Project

Our project is a program that is used by a teacher in charge of the Archery Club. For students that haven't paid for the club fee, a pending list is shown. This is to provide an alternative way for the teacher in order to give permission for the students to borrow the equipment provided by the Archery Club. Only students that have paid for the club fee are the only ones who have permission to borrow the equipment. For those that haven't paid the club fee, the total amount of fee is calculated to record the loss. Providing a user-friendly interface, our program has come out with a search system. This is to find and update any necessary changes. Other than that, our program also put a different type of bow in their own list. There are 4 types of bow and the program has 4 different lists that consist of the student's information based on the bow that they choose. Moreover, the student's name is sorted by their name followed by the id in ascending order.

List of processing

1. Create Linked List object name studList and store students into the studList.
2. Remove students that haven't paid the club fee from LinkedList studList and insert into Queue pendingList.
3. Display the elements in pendingList.
4. Calculate total student fee in pendingList using Queue.
5. Search for name based on user input and update the searched name's bowCode to recurve bow.
6. Get all the students that borrowed from studList and insert them into 4 different. queues recurveQ, compoundQ, longBowQ and crossBowQ.
7. Display the elements in recurveQ, compoundQ, longBowQ and crossBowQ.
8. Sort the names followed by the id in studList in ascending order.
9. Display the elements in studList.

Complete coding of all classes

Class Node

```
public class Node
{
    Archery data;
    Node next;

    public Node(Archery data){ this.data = data; }

    //constructor: create a ListNode that refers to Object o and
to the next ListNode in the list
    Node(Archery o, Node nextNode)
    {
        data = o;
        next = nextNode;
    }

    //return a reference to the Object in this node
    Archery getObject()
    {
        return data;
    }

    //return the next node
    Node getLink()
    {
        return next;
    }
}
```

Class LinkedList

```
import java.util.*;

public class LinkedList
{
    Node first, last, current;
    String name;

    //constructor: constructor an empty List with s as the name
    public LinkedList(String s)
    {
        name = s;
        first = last = current = null;
    }

    //constructor: constructor an empty List with "list" as the
name
   public LinkedList()
    {
        this("linked list");
    }

    //returns true if the list is empty
    public boolean isEmpty()
    {
        return first == null;
    }

    //inserts and Object at the front of the list
    //If List is empty, firstNode and lastNode will refer to the
same object.
    //Otherwise, firstNode refers to new Node
    public void addFirst(Archery insertItem)
    {
        if(isEmpty())
            first = last = new Node(insertItem);
    else
            first = new Node(insertItem, first);
    }

    //insert an Object at the end of the list
    //If list is empty, firstNode and lastNode will refer to the
same object.
    //Otherwise, lastNode's next instance variable refers to new
node
    public void addLast(Archery insertItem)
    {
        if (isEmpty())
            first = last = new Node(insertItem);
        else
            last = last.next = new Node(insertItem);
    }
```

```
public Node getNode(int index)
{
    Node current = first;
    if(index < 0 || index > size())
        throw new IndexOutOfBoundsException();

    for(int i = 0; i < index; i++)
    {
        current = current.next;
    }

    return current;
}

//returns the first element
public Archery getFirst()
{
    if(isEmpty())
        return null;
    else
    {
        current = first;
        return current.data;
    }
}

public Archery getNext()
{
    if(current != last)
    {
        current = current.next;
        return current.data;
    }
    else
        return null;
}

public Archery getLast()
{
    if (isEmpty())
        return null;
    else
    {
        current = last;
        return current.data;
    }
}

public Archery set(int i, Archery insertItem)
{
    if(i < 0 || i > size())
        throw new IndexOutOfBoundsException();
```

```
    Archery old = insertItem;

    return old;
}

//remove the first node from the List
public Archery removeFirst() throws EmptyListException
{
    Archery removeItem = null;
    if (isEmpty())
        throw new EmptyListException(name);

    removeItem = first.data; //retrieve the data

    //insert the first and last references
    if (first.equals(last))
        first = last = null;
    else
    first = first.next;

    return removeItem;
}

public Archery removePending() throws EmptyListException
{
    Node current, previous = null;
    current = first;
    Archery removeItem = current.data;

    while(current!=null)
    {
        if (current.data.getPayment() == false)
        {
            if (first == last)
            {
                first = last = null;
            }
            else if (current == first)
            {
                first = first.next;
                current = current.next;
            }
            else if (current == last)
            {
                last = previous;
                last.next = null;
            }
        }
    }
    return removeItem;
}
```

```
    //returns the size of the list
    public int size()
    {
        Node current = first;
        int size = 0;
        while(current != null)
        {
            size += 1;
            current = current.next;
        }
        return size;
    }
}
```

Class Queue

```
import java.util.*;

public class Queue
{
   LinkedList list;

   public Queue()
   {
      list = new LinkedList(); //default constructor
   }

   public boolean isEmpty()
   {
      return list.isEmpty(); //method is empty
   }

   public int size()
   {
      return list.size(); //method size
   }

   public void enqueue(Archery element)
   {
      list.addLast(element); //method enqueue
   }

   public Archery dequeue()
   {
      return list.removeFirst(); //method dequeue
   }

   public Archery front()
   {
      return list.getFirst(); //method front
   }

   public Archery rear()
   {
      return list.getLast();
   }
}
```

Class EmptyListException

```
public class EmptyListException extends RuntimeException
{
    public EmptyListException(String name)
    {
        super("The " + name + " is empty");
    }
}
```

Class Archery

```
import java.text.DecimalFormat;

public class Archery
{
    private String studentID;
    private String studentName;
    private int age;
    private char gender; //male, female
    private int bowCode; //recurve, compound, longbow, crossbow
    private boolean payment; // if true = paid, if false =
pending

    //normal constructor
    public Archery(String i, String n, int a, char g, int c,
boolean p)
    {
        studentID = i;
        studentName = n;
        age = a;
        gender = g;
        bowCode = c;
        payment = p;
    }

    public String getStuID() { return studentID; }
    public String getStuName() { return studentName; }
    public int getAge() { return age; }
    public char getGender() { return gender; }
    public int getBowCode() { return bowCode; }
    public boolean getPayment() { return payment; }

    public void setBowCode(int bowCode) { this.bowCode = bowCode;
}

    public String bowType(int bowCode)
    {
        String bowType = null;

        if(bowCode == 1)
```