**National Aviation University**

Faculty of Cybersecurity, Computer and Software Engineering

Software Engineering Department

**REPORT**

**Educational and Technological Training (Part 1)**

Student 1 course PI-121B(A)

Speciality 121 Software Engineering

Salivon Valerie Kostyantynivna

Base of training : NAU

Head of training: Senior Lecturer of SED Bezkorovaina Yuliia Mykolaivna

Kyiv – 2019

Content

[Individual tasks: 3](#_Toc11888030)

[**Programming environment** 6](#_Toc11888031)

[**The difference between C # and Java** 7](#_Toc11888032)

[**Listing (C #)** 8](#_Toc11888033)

[**Listing (java)** 16](#_Toc11888034)

[**Conclusion** 22](#_Toc11888035)

The basis for the training: curriculum specialty 121 "Software Engineering", based on the National Aviation University, the order of the rector from "\_\_" \_\_\_\_\_\_\_\_\_ 2019, №\_\_\_\_\_.

Terms: 10.06.2019 - 21.06.2019 р.

# Individual tasks:

Common task

Write a program that performs the following actions:

1. Description of the type of data stored in the bidirectional list, according to table (see variant).

The components perform as private members of the organization of access to them or as automatic properties.

2. Description of the type of nodes in the list

3. Description of the abstract data type (ADТ), which includes:

• value area: reference to the first (if necessary - and to the last one) element of the bidirectional list

• scope of action (see variant).

Provide in the methods of generation (processing) of exceptional situations when incorrect hits to the values of the list, the source code ADТ to organize in a separate project (dll) and independent of the type of application interface (console, graphic, web).

4. Demonstration of the implementation of ADТ:

• add multiple items to the list;

• remove multiple items in the list;

• output list items to the console in the form of a table;

• find list items according to the task and output the values of these items to the console

• output the updated list to the console after each change in the list

The interface is preferably implemented in the form of elementary console menu with the ability to perform actions with the list in any order and quantity Should be provided control of user input.

**Variant** 13

Scope of action:

a) method of adding an item to the n-position of the list;

b) method of removing the items with the n value of the list;

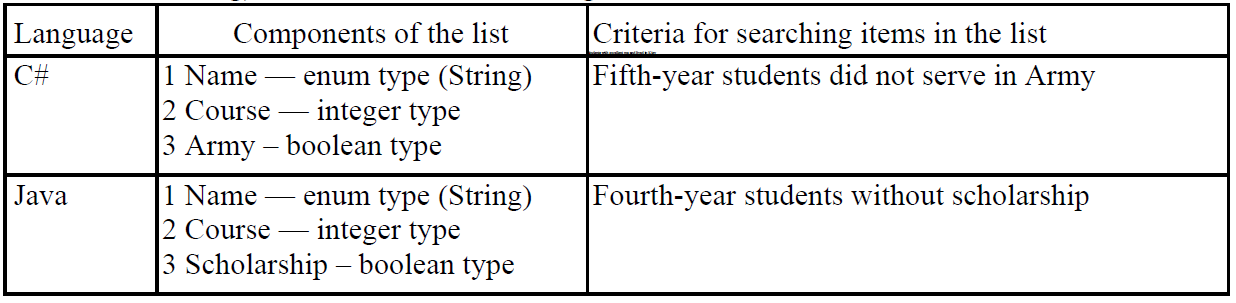
c) indexer read-edit values given node sequence number (numbered list starts with node 0);

d) property (read only) - length of the list (number of nodes);

e) methods of iterating the list in the natural order, starting from the first (method of obtaining the initial value, method of obtaining the next);

f) method of sorting a list by one of the real or integer components of the values of the list items for growth (bubble sort method);

g) search list items according to table.



CALENDAR PLAN-SCHEDULE OF TRAINING

|  |  |  |  |
| --- | --- | --- | --- |
| # | The object of training and types of work | Terms | |
| FROM | TO |
| 1 | Constituent Assembly. Introduction to the instruction № 784 on the safety of life for students during the course of technological technological training, part 1, 2017-2018 s.y. | 10.06.2019 | 10.06.2019 |
| 2 | Obtaining an individual task | 11.06.2019 | 11.06.2019 |
| 3 | Creating a project | 12.06.2019 | 12.06.2019 |
| 4 | Creating classes Data, Node and DoubleLinkedList | 12.06.2019 | 12.06.2019 |
| 5 | Creating methods, which checking if the list is empty and searching the size of the list | 12.06.2019 | 13.06.2019 |
| 6 | Creating the method of adding an item to the n-position of the list and removing the items in the n-position of the list | 13.06.2019 | 14.06.2019 |
| 7 | Creating method of displaying the list in the natural order, starting with the first | 14.06.2019 | 15.06.2019 |
| 8 | Creating method, which searching students according to the table | 15.06.2019 | 15.06.2019 |
| 9 | Creating method of sorting a list by one of the real or integer components of the values of the list items for growth (bubble sort method) | 15.06.2019 | 16.06.2019 |
| 10 | Creating the property (read only) - length of the list (number of nodes) | 15.06.2019 | 15.06.2019 |
| 11 | Creating method, which outputs list items to the console in the form of a table | 16.06.2019 | 17.06.2019 |
| 12 | Defense of training | 21.06.2018 | 21.06.2018 |

Head of Training from the University : \_\_\_\_\_\_\_\_\_\_\_\_\_ Y.M. Bezkorovaina

# **Programming environment**

**Microsoft Visual Studio** is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) from [Microsoft](https://en.wikipedia.org/wiki/Microsoft). It is used to develop [computer programs](https://en.wikipedia.org/wiki/Computer_program), as well as [websites](https://en.wikipedia.org/wiki/Web_site), [web apps](https://en.wikipedia.org/wiki/Web_app), [web services](https://en.wikipedia.org/wiki/Web_service) and [mobile apps](https://en.wikipedia.org/wiki/Mobile_app).

Visual Studio supports 36 different [programming languages](https://en.wikipedia.org/wiki/Programming_language) and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include [C](https://en.wikipedia.org/wiki/C_(programming_language)), [C++](https://en.wikipedia.org/wiki/C%2B%2B), [C++/CLI](https://en.wikipedia.org/wiki/C%2B%2B/CLI), [Visual Basic,.NET](https://en.wikipedia.org/wiki/Visual_Basic_.NET), [C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)), [F#](https://en.wikipedia.org/wiki/F_Sharp_(programming_language)), [JavaScript](https://en.wikipedia.org/wiki/JavaScript), [TypeScript](https://en.wikipedia.org/wiki/TypeScript" \o "TypeScript), [XML](https://en.wikipedia.org/wiki/XML), [XSLT](https://en.wikipedia.org/wiki/XSLT), [HTML](https://en.wikipedia.org/wiki/HTML), and [CSS](https://en.wikipedia.org/wiki/Cascading_Style_Sheets). Support for other languages such as [Python](https://en.wikipedia.org/wiki/Python_(programming_language)),  [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)), [Node.js](https://en.wikipedia.org/wiki/Node.js), and [M](https://en.wikipedia.org/wiki/MUMPS) among others is available via [plug-ins](https://en.wikipedia.org/wiki/Plug-in_(computing)). [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) (and [J#](https://en.wikipedia.org/wiki/J_Sharp)) were supported in the past.

**C#** (pronounced C sharp) is a general-purpose, [multi-paradigm programming language](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language) encompassing [strong typing](https://en.wikipedia.org/wiki/Strong_typing), [lexically scoped](https://en.wikipedia.org/wiki/Lexically_scoped), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [declarative](https://en.wikipedia.org/wiki/Declarative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming), [generic](https://en.wikipedia.org/wiki/Generic_programming), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) ([class](https://en.wikipedia.org/wiki/Class_(computer_science))-based), and [component-oriented](https://en.wikipedia.org/wiki/Component-based_software_engineering)programming disciplines.

By design, C# is the programming language that most directly reflects the underlying [Common Language Infrastructure](https://en.wikipedia.org/wiki/Common_Language_Infrastructure) (CLI).  Most of its intrinsic types correspond to value-types implemented by the CLI framework. However, the language specification does not state the code generation requirements of the compiler: that is, it does not state that a C# compiler must target a Common Language Runtime, or generate [Common Intermediate Language](https://en.wikipedia.org/wiki/Common_Intermediate_Language) (CIL), or generate any other specific format. Theoretically, a C# compiler could generate machine code like traditional compilers of C++ or [Fortran](https://en.wikipedia.org/wiki/Fortran).

**Java** is a [general-purpose](https://en.wikipedia.org/wiki/General-purpose_language) [programming language](https://en.wikipedia.org/wiki/Programming_language) that is [class-based](https://en.wikipedia.org/wiki/Class-based_programming), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) (although not a pure OO language, as it contains primitive types), and designed to have as few implementation [dependencies](https://en.wikipedia.org/wiki/Dependency_(computer_science)) as possible. It is intended to let [application developers](https://en.wikipedia.org/wiki/Application_developer) write once, run anywhere (WORA), meaning that [compiled](https://en.wikipedia.org/wiki/Compiler) Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to [bytecode](https://en.wikipedia.org/wiki/Java_bytecode" \o "Java bytecode)that can run on any [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine) (JVM) regardless of the underlying [computer architecture](https://en.wikipedia.org/wiki/Computer_architecture). The [syntax](https://en.wikipedia.org/wiki/Syntax_(programming_languages)) of Java is similar to [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B), but it has fewer [low-level](https://en.wikipedia.org/wiki/Low-level_programming_language) facilities than either of them. As of 2018, Java was one of the most [popular programming languages in use](https://en.wikipedia.org/wiki/Measuring_programming_language_popularity) according to [GitHub](https://en.wikipedia.org/wiki/GitHub), particularly for [client-server](https://en.wikipedia.org/wiki/Client%E2%80%93server) [web applications](https://en.wikipedia.org/wiki/Web_applications), with a reported 9 million developers.

**IntelliJ IDEA** is a [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) for developing computer software. It is developed by [JetBrains](https://en.wikipedia.org/wiki/JetBrains" \o "JetBrains) (formerly known as IntelliJ), and is available as an [Apache 2 Licensed](https://en.wikipedia.org/wiki/Apache_2_License) community edition, and in a [proprietary](https://en.wikipedia.org/wiki/Proprietary_software) commercial edition. Both can be used for commercial development. The IDE provides certain features like [code completion](https://en.wikipedia.org/wiki/Autocomplete) by analyzing the context, code navigation which allows jumping to a class or declaration in the code directly, [code refactoring](https://en.wikipedia.org/wiki/Code_refactoring), linting and options to fix inconsistencies via suggestions.

# **The difference between C # and Java**

**.NET platform**

.NET applications are built by developing any .NET language such as C #, VB, or C ++, which are then compiled into the CIL (Common Intermediate Language) bytecode. Next, CIL becomes a collection that can then be executed on any Windows machine.

**Java platform**

J2EE applications are built by developing only Java, which are then compiled into Java in bytecode. An intermediate bytecode will then be converted into a JVM (Java Virtual Machine). Unlike. NET platform, the collection code that is created with JVM can Work on any platform, including Windows, Unix, Solaris, or Mac OS.

**Platform dependence**

The main difference between Java and. NET is their platform dependence. . NET applications you can run only on Windows, while Java applications can run on any one the machine.

**Cost**

The next big difference between Java and. NET is the cost. Since Java is open source code, it's completely free to develop and deploy Java applications. It the most popular integrated development environments, such as Eclipse, and JBuilder. In addition, as well Java applications independent of the platform, they can be applied on UNIX machines, which is a free OS. In fact, only the costs associated with launching Java programs will be worth the cost equipment.

. NET applications, on the other hand, can only work on Windows machines, yes, a license Windows server will need to be purchased. Unlike the Java platform that uses free development environments. . NET applications are built with Visual Studio, which also requires paid license.

When developing the program in these two languages, I noticed some differences:

1. Console menu - in C # - the command line of Windows opens and Java has its own console menu, which is embedded in the development environment IntelliJ IDEA;
2. In C # - the name of the enumeration type can only be selected as an integer number in the list, and in Java - there is valueof - the ability to enter the term is the name of the enumeration type.

# **Listing (C #)**

Project – Practica

File – Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Practica

{

class Program

{

static void Main(string[] args)

{

Data Student\_1 = new Data("Max", 1, true);

Data Student\_2 = new Data("Valerie", 4, false);

Data Student\_3 = new Data("Vania", 2, true);

Data Student\_4 = new Data("Ann", 1, false);

Data Student\_5 = new Data("Julia", 3, false);

DoublyLinkedList MyList = new DoublyLinkedList();

MyList.addFront(Student\_1);

MyList.addFront(Student\_2);

MyList.addBack(Student\_4);

MyList.addBack(Student\_5);

MyList.insertAfter(1, Student\_3);

MyList.print();

Console.WriteLine("\n======================================\n");

Console.WriteLine("Deleting of element: \n");

MyList.deleteKey(2);

MyList.print();

Console.WriteLine("\n======================================\n");

MyList.sort();

Console.WriteLine("Sorted List: \n");

MyList.print();

Console.WriteLine("\n======================================\n");

Console.WriteLine("Element for Searching was found: \n");

MyList.Search(3);

Console.WriteLine("\n======================================\n");

Console.WriteLine($"Size: {MyList.getSize()}");

Console.WriteLine("\n======================================\n");

MyList.NameSearch("Valerie");

Console.WriteLine("\n======================================\n");

}

}

}

File – Data.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Practica

{

public class Data

{

private String name;

private int course;

private Boolean army;

public Data()

{

name = "";

course = 0;

army = false;

}

public Data(String name, int course, Boolean army)

{

this.name = name;

this.course = course;

this.army = army;

}

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

public int getCourse()

{

return course;

}

public void setCourse(int course)

{

this.course = course;

}

public Boolean getArmy()

{

return army;

}

public void setArmy(Boolean army)

{

this.army = army;

}

}

}

File – Node.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Practica

{

public class Node : Data

{

private Node next;

private Node previous;

private Data myStudent = new Data();

public Node(Data Student)

{

this.myStudent.setArmy(Student.getArmy());

this.myStudent.setCourse(Student.getCourse());

this.myStudent.setName(Student.getName());

this.next = null;

this.previous = null;

}

public String valueName()

{

return myStudent.getName();

}

public int valueCourse()

{

return myStudent.getCourse();

}

public Boolean valueArmy()

{

return myStudent.getArmy();

}

public Node getNext()

{

return next;

}

public void setNext(Node next)

{

this.next = next;

}

public Node getPrevious()

{

return previous;

}

public void setPrevious(Node previous)

{

this.previous = previous;

}

public Data getMyStudent()

{

return myStudent;

}

public void setMyStudent(Data myStudent)

{

this.myStudent = myStudent;

}

public String toString()

{

return "Name: " + myStudent.getName() +

"\nCourse: " + myStudent.getCourse() +

"\nArmy: " + myStudent.getArmy() + "\n";

}

}

}

File – DoublyLinkedList.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Practica

{

public class DoublyLinkedList

{

private Node head;

private int size;

public DoublyLinkedList()

{

head = null;

size = 0;

}

private bool isEmpty()

{

return head == null;

}

public void Search(int course)

{

Node current = head;

while (current != null)

{

int a = current.getMyStudent().getCourse();

if (a == course)

{

Console.WriteLine(current.toString());

}

current = current.getNext();

}

}

public void NameSearch(string name)

{

Node current = head;

while (current != null)

{

string a = current.getMyStudent().getName();

if (a == name)

{

Console.WriteLine(current.toString());

}

current = current.getNext();

}

}

public void addFront(Data node)

{

if (head == null)

{

head = new Node(node);

head.setPrevious(null);

head.setNext(null);

}

else

{

Node newNode = new Node(node);

newNode.setNext(head);

head = newNode;

}

size++;

}

public void addBack(Data node)

{

if (head == null)

{

head = new Node(node);

head.setPrevious(null);

head.setNext(null);

}

else

{

Node current = head;

while (current.getNext() != null)

{

current = current.getNext();

}

Node newNode = new Node(node);

newNode.setPrevious(current);

current.setNext(newNode);

}

size++;

}

public void removeFront()

{

if (head == null) return;

head = head.getNext();

head.setPrevious(null);

size--;

}

public void removeBack()

{

if (head == null) return;

if (head.getNext() == null)

{

head = null;

size--;

return;

}

Node current = head;

while (current.getNext().getNext() != null)

{

current = current.getNext();

}

current.setNext(null);

size--;

}

public void print()

{

if (isEmpty()) { Console.WriteLine("No nodes for print"); return; }

Console.WriteLine("First ---> Last");

Node current = head;

while (current != null)

{

Console.WriteLine(current.toString());

current = current.getNext();

}

}

public void sort()

{

if (head == null)

{

Console.WriteLine("List is empty");

return;

}

if ((head.getNext() == null)

&& (head.getPrevious() == null))

{

Console.WriteLine("List has one node");

return;

}

Node current = head;

for (int i = 0; i < size - 1; i++)

{

for (int j = 0; current.getNext() != null; j++)

{

int h1 = current.getMyStudent().getCourse();

int h2 = current.getNext().getMyStudent().getCourse();

if (h1 > h2)

{

Data buffer = current.getMyStudent();

current.setMyStudent(current.getNext().getMyStudent());

current.getNext().setMyStudent(buffer);

}

current = current.getNext();

}

current = head;

}

}

public int getSize() { return size; }

public void insertAfter(int key, Data Student)

{

if (this.isEmpty()) return;

Node current = head; // start at beginning

int count = 0;

while (current.getNext() != null && count != key) // until match is found,

{

current = current.getNext(); // move to next Node

count++;

}

Node newNode = new Node(Student); // make new Node

if (current.getNext() == null) // if last Node,

{

newNode.setNext(null); // newNode --> null

newNode.setPrevious(current);

current.setNext(newNode);

current = newNode; // newNode <-- last

}

else // not last Node,

{

newNode.setNext(current.getNext()); // newNode --> old next

// newNode <-- old next

current.getNext().setPrevious(newNode);

}

newNode.setPrevious(current); // old current <-- newNode

current.setNext(newNode); // old current --> newNode

size++;

}

public void deleteKey(int key)

{ // (assumes non-empty list)??????????????

if (this.isEmpty()) return;

Node current = head; // start at beginning

int count = 0;

while (current.getNext() != null && count != key) // until match is found,

{

current = current.getNext(); // move to next Node

count++;

}

if (current == head) // found it; first item?

head = current.getNext(); // first --> old next

if (current.getNext() == null) // last item?

current = current.getPrevious(); // old previous <-- last

else // not last

// old previous <-- old next

current = current.getPrevious();

current.setNext(current.getNext().getNext());

current.getNext().setPrevious(current);

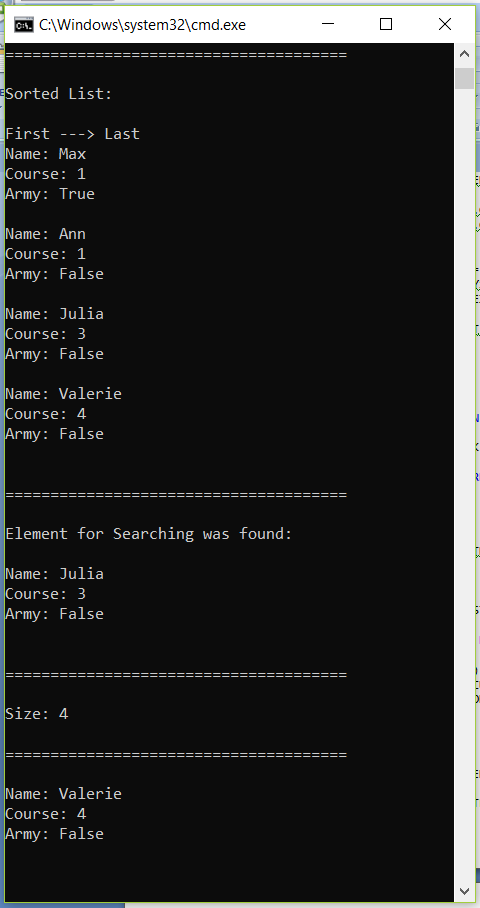
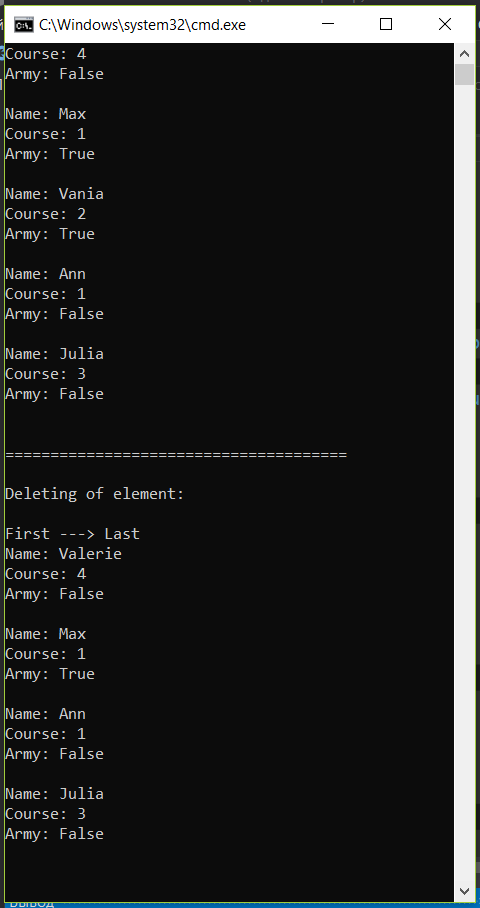
size--;

}

}

}

Examples of program execution:



# **Listing (java)**

Project – Practica

File – Main.java

package com.company;  
  
import LinkedListNodee.LinkedListNodee;  
import Node.Node;  
import Student.Student;  
  
public static void main(String[] args) {  
  
 Data Student\_1 = new Data("Max", 1, true);  
 Data Student\_2 = new Data("Valerie", 4, false);  
 Data Student\_3 = new Data("Vania", 2, true);  
 Data Student\_4 = new Data("Ann", 1, false);  
 Data Student\_5 = new Data("Julia", 3, false);  
  
 DoublyLinkedList MyList = new DoublyLinkedList();  
  
 MyList.addFront(Student\_1);  
 MyList.addFront(Student\_2);  
 MyList.addBack(Student\_4);  
 MyList.addBack(Student\_5);  
 MyList.insertAfter(1, Student\_3);  
 MyList.print();  
 System.*out*.println("\n======================================\n");  
  
 System.*out*.println("Deleting of element: \n");  
 MyList.deleteKey(2);  
 MyList.print();  
 System.*out*.println("\n======================================\n");  
  
 MyList.sort();  
 System.*out*.println("Sorted List: \n");  
 MyList.print();  
 System.*out*.println("\n======================================\n");  
  
 System.*out*.println("Element for Searching was found: \n");  
 MyList.Search(3);  
 System.*out*.println("\n======================================\n");  
  
 System.*out*.println("Size: ");  
 MyList.getSize();  
 System.*out*.println("\n======================================\n");  
  
 MyList.NameSearch("Valerie");  
 System.*out*.println("\n======================================\n");  
  
}

File – Data.java

public class Data {  
  
 private String name;  
 private int course;  
 private Boolean army;  
  
 public Data()  
 {  
 name = "";  
 course = 0;  
 army = false;  
 }  
  
 public Data(String name, int course, Boolean army)  
 {  
 this.name = name;  
 this.course = course;  
 this.army = army;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getCourse() {  
 return course;  
 }  
  
 public void setCourse(int course) {  
 this.course = course;  
 }  
  
 public Boolean getArmy() {  
 return army;  
 }  
  
 public void setArmy(Boolean army) {  
 this.army = army;  
 }  
}

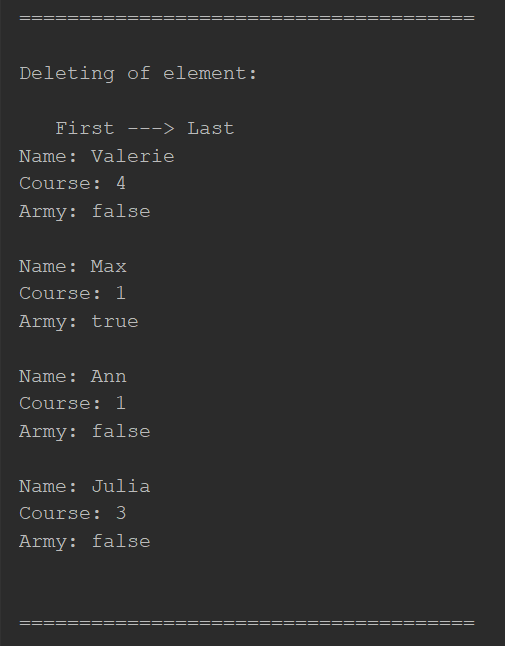
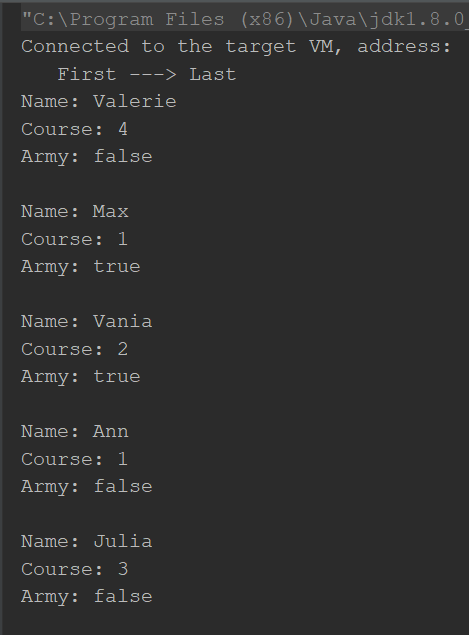
File – Node.java

public class Node extends Data {  
  
 private Node next;  
 private Node previous;  
  
 private Data myStudent = new Data();  
  
 public Node(Data Student)  
 {  
 this.myStudent.setArmy(Student.getArmy());  
 this.myStudent.setCourse(Student.getCourse());  
 this.myStudent.setName(Student.getName());  
 this.next = null;  
 this.previous = null;  
 }  
  
 public String valueName()  
 {  
 return myStudent.getName();  
 }  
  
 public int valueCourse()  
 {  
 return myStudent.getCourse();  
 }  
  
 public Boolean valueArmy()  
 {  
 return myStudent.getArmy();  
 }  
  
 public Node getNext() {  
 return next;  
 }  
  
 public void setNext(Node next) {  
 this.next = next;  
 }  
  
 public Node getPrevious() {  
 return previous;  
 }  
  
 public void setPrevious(Node previous) {  
 this.previous = previous;  
 }  
  
 public Data getMyStudent() {  
 return myStudent;  
 }  
  
 public void setMyStudent(Data myStudent) {  
 this.myStudent = myStudent;  
 }  
  
  
  
  
 @Override  
 public String toString() {  
 return "Name: " + myStudent.getName() +  
 "\nCourse: " + myStudent.getCourse() +  
 "\nArmy: " + myStudent.getArmy() + "\n";  
 }  
}

File – DoublyLinkedList.java

public class DoublyLinkedList {  
 private Node head;  
 private int size;  
  
 public DoublyLinkedList() {  
 head = null;  
 size = 0;  
 }  
  
 private boolean isEmpty(){  
 return head == null;  
 }  
  
 public void Search(int course)  
 {  
 Node current = head;  
 while (current != null){  
 int a = current.getMyStudent().getCourse();  
 if(a == course){  
 System.*out*.println(current);  
 }  
 current = current.getNext();  
 }  
 }  
  
 public void addFront(Data node){  
 if(head == null){  
 head = new Node(node);  
 head.setPrevious(null);  
 head.setNext(null);  
 } else {  
 Node newNode = new Node(node);  
 newNode.setNext(head);  
 head = newNode;  
 }  
 size++;  
 }  
  
 public void addBack(Data node){  
 if(head == null){  
 head = new Node(node);  
 head.setPrevious(null);  
 head.setNext(null);  
 } else {  
 Node current = head;  
 while(current.getNext() != null){  
 current = current.getNext();  
 }  
 Node newNode = new Node(node);  
 newNode.setPrevious(current);  
 current.setNext(newNode);  
 }  
 size++;  
 }  
  
 public void removeFront(){  
 if (head == null) return;  
 head = head.getNext();  
 head.setPrevious(null);  
 size--;  
 }  
  
 public void removeBack(){  
 if (head == null) return;  
 if(head.getNext() == null){  
 head = null;  
 size--;  
 return;  
 }  
  
 Node current = head;  
 while(current.getNext().getNext() != null){  
 current = current.getNext();  
 }  
 current.setNext(null);  
 size--;  
 }  
  
 public void print(){  
 if (isEmpty()) {System.*out*.println("No nodes for print"); return;}  
 System.*out*.println(" First ---> Last ");  
 Node current = head;  
 while(current != null){  
 System.*out*.println(current);  
 current = current.getNext();  
 }  
 }  
  
 public void sort(){  
 if (head == null) {  
 System.*out*.println("List is empty");  
 return;}  
 if ((head.getNext() == null)  
 && (head.getPrevious() == null)) {  
 System.*out*.println("List has one node");  
 return;}  
 Node current = head;  
 for(int i = 0; i < size - 1; i++){  
 for(int j = 0; current.getNext() != null ; j++){  
 int h1= current.getMyStudent().getCourse();  
 int h2 = current.getNext().getMyStudent().getCourse();  
 if(h1 > h2){  
 Data buffer = current.getMyStudent();  
 current.setMyStudent(current.getNext().getMyStudent());  
 current.getNext().setMyStudent(buffer);  
 }  
 current = current.getNext();  
 }  
 current = head;  
 }  
 }  
  
 public void getSize(){System.*out*.print(size);}  
  
 public void insertAfter(int key, Data Student)  
 {  
 if (this.isEmpty()) return;  
 Node current = head; // start at beginning  
 int count = 0;  
 while(current.getNext() != null && count != key) // until match is found,  
 {  
 current = current.getNext(); // move to next Node  
 count++;  
 }  
  
 Node newNode = new Node(Student); // make new Node  
  
 if(current.getNext()==null) // if last Node,  
 {  
 newNode.setNext(null); // newNode --> null  
 newNode.setPrevious(current);  
 current.setNext(newNode);  
 current = newNode; // newNode <-- last  
 }  
 else // not last Node,  
 {  
 newNode.setNext(current.getNext()); // newNode --> old next  
 // newNode <-- old next  
 current.getNext().setPrevious(newNode);  
 }  
 newNode.setPrevious(current); // old current <-- newNode  
 current.setNext(newNode); // old current --> newNode  
 size++;  
  
 }  
  
 public void deleteKey(int key) // delete item w/ given key  
 { // (assumes non-empty list)??????????????  
 if (this.isEmpty()) return;  
 Node current = head; // start at beginning  
 int count = 0;  
 while(current.getNext() != null && count != key) // until match is found,  
 {  
 current = current.getNext(); // move to next Node  
 count++;  
 }  
 if(current==head) // found it; first item?  
 head = current.getNext(); // first --> old next  
  
  
 if(current.getNext()== null) // last item?  
 current = current.getPrevious(); // old previous <-- last  
 else // not last  
 // old previous <-- old next  
 current = current.getPrevious();  
 current.setNext(current.getNext().getNext());  
 current.getNext().setPrevious(current);  
 size--;  
 }  
  
 public void NameSearch(String name)  
 {  
 Node current = head;  
 while (current != null)  
 {  
 String a = current.getMyStudent().getName();  
 if (a == name)  
 {  
 System.*out*.println(current.toString());  
 }  
 current = current.getNext();  
 }  
 }  
  
 }

Examples of program execution:



# 

# **Conclusion**

The goals were achieved - to learn how to implement the ADT, namely, at the form of a two-way list. I got better understanding the data structures that they are, and how to implement them in practice. The biggest difficulty in writing the program came from use the enumeration type to designate student names. There were also difficulties with the iteration of the list, I had to follow different interfaces and implement them methods.