

# CEGEP VANIER COLLEGE

## CENTRE FOR CONTINUING EDUCATION

### Programming Algorithms and Patterns

#### 420-930-VA

Teacher: Samir Chebbine

Lab 2

Jul 21, 2023

### Lab 2: ArrayList and Linked List

Complete all these following programs as explained during **Zoom Synchronous classes**. All *missing coding statements* were provided there with explanation. Create and Submit a Word file **Lab2OOPProgramminAlgorithmsYourName.docx** which includes output screenshots for every Java Project. Submit the Java projects too.

#### 1. ArrayLists Data Structure

Create a Java Project **ArrayListPayRollProject** using Eclipse IDE that allows payroll department to issue a pay stub for a given employee. The end user has to read input text file Payroll.in (provided to you) and populates data file Payroll.in. into an **ArrayList** data structure of type PayRollEmployee class type.

- You need to design a **Java class** called **PayRollEmployee**, which takes the **emp\_id**, **emp\_name**, **emp\_ssn**, **number\_whr**, **h\_rate** as **private** members. The variables called **Fed\_Tax**, **Prv\_Tax**, **QP\_Ins**, **E\_ins**, **Qpp**, **Union\_d**, as **public** and static data members.
- Create **TestArrayListPayRoll.java** where you populate an ArrayList data structure of PayRollEmployee class type to be referenced by (payRollArrList) from input file PayRoll.in. Set every component using the implemented setter methods.

- Add **default constructor**, setters, getters, and toString()
- Add methods called calculate\_TotalIncome(), calculate\_TotalDeduction(), calculate\_TotalNetAmount() in PayRollEmployee class to calculate the following respectively:

$$\text{Total\_Income} = \text{number\_whr} * \text{h\_rate}$$

#### Deductions:

Provincial tax (Prv\_Tax): 9% of Total\_income.

Federal tax (Fed\_Tax): 7% of Total\_income.

Que. parental insurance. plan (QP\_Ins): 0.55% of Total\_income.

Employment insurance (E\_ins): 1.4% of Total\_income.

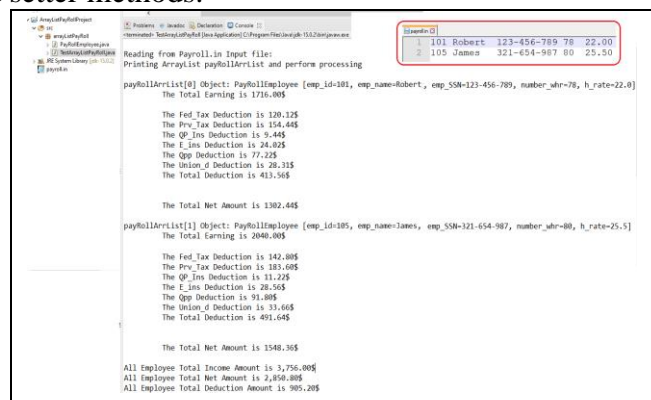
(Quebec pension plan) Qpp : 4.5% of Total\_income.

Union dues (Union\_d): 1.65% of Total\_income.

The total Net Amount (*Net\_Amount*) is calculated according to the following formula:

$$\text{Net\_Amount} = \text{Total\_Income} - \text{Deductions}$$

Calculate the total of different employee amounts of all ArrayList components. Display totals as shown above.



```
Reading from Payroll.in Input file:
Printing ArrayList payrollArrList and perform processing

payrollArrList[0] Object: PayRollEmployee [emp_id=101, emp_name=Robert, emp_ssn=123-456-789, number_whr=78, h_rate=22.0]
The Total Earning is 1716.00$

The Fed_Tax Deduction is 120.12$
The Prv_Tax Deduction is 154.44$
The QP_Ins Deduction is 9.44$
The E_ins Deduction is 24.02$
The Qpp Deduction is 77.22$
The Union_d Deduction is 28.11$
The Total Deduction is 493.36$

The Total Net Amount is 1302.64$

payrollArrList[1] Object: PayRollEmployee [emp_id=105, emp_name=James, emp_ssn=321-654-987, number_whr=80, h_rate=25.5]
The Total Earning is 2040.00$

The Fed_Tax Deduction is 142.80$
The Prv_Tax Deduction is 183.60$
The QP_Ins Deduction is 11.22$
The E_ins Deduction is 28.56$
The Qpp Deduction is 91.80$
The Union_d Deduction is 33.66$
The Total Deduction is 469.64$

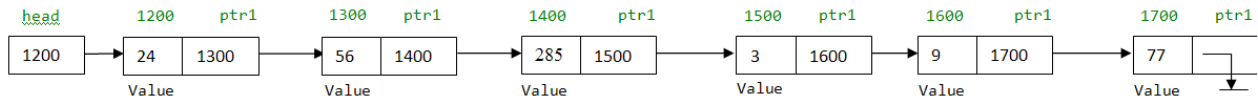
The Total Net Amount is 1548.36$

All Employee Total Income Amount is 3,756.00$
All Employee Total Net Amount is 2,850.00$
All Employee Total Deduction Amount is 905.20$
```

## 2. Linked List:

- Create *LinkedList1Project* using Eclipse IDE. Create *TestLinkedList1.java* where Items will be added to the linked list.

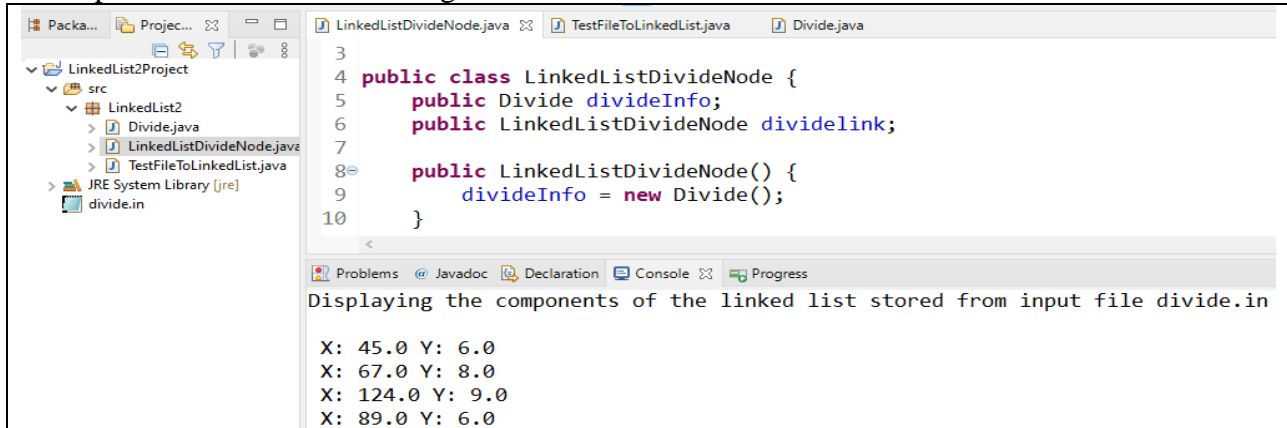
You construct a **Linked List**, if you point every reference object to the subsequent reference object node where value info is stored.



| Linked list class   | Linked list construction  |
|---|---|
| <pre> public class LinkedListNode {      public int info;     public LinkedListNode link;  } </pre>   | <pre> LinkedListNode headNode, newNode;  headNode = new LinkedListNode(); headNode.info= 24;           // store 24 in the object headNode headNode.link= null;  newNode = new LinkedListNode(); newNode.info= 56;           // store 56 in the object newNode newNode.link= null; headNode.link = newNode;    // Link the address of                              // newNode to headNode </pre> |
| <pre> public class LinkList1 {      public static void main(String[] args) {          LinkedListNode headNode, newNode;          headNode = new LinkedListNode();         headNode.info= 24;           // store 24 in the object headNode         headNode.link= null;          newNode = new LinkedListNode();         newNode.info= 56;           // store 56 in the object newNode         newNode.link= null;         headNode.link = newNode;    // Link the address of                                     // newNode to headNode          newNode = new LinkedListNode();         newNode.info= 285;          // store 285 in the object newNode         newNode.link= null;         headNode.link.link= newNode; // Link the address of                                     // newNode to headNode          .....          .....          System.out.println("Displaying the components of the linked list \n\n");          LinkedListNode travNode;         while (.....)         {             System.out.println(" Value: " + travNode.info + " ");             .....         }      }  } </pre> |   |

- Create *TestLinkedList2.java* to read input integer from the console until the user enters -999 (acts as sentinel), store the values into a *linked list* (build in *forward* manner), and display its components by traversing the link in *forward* manner.

- c) Create *LinkedList2Project* using Eclipse IDE. Create *TestFileToLinkedList.java* to read from input file *divide.in* and storing its content into Linked list.



```

3
4 public class LinkedListDivideNode {
5     public Divide divideInfo;
6     public LinkedListDivideNode dividelink;
7
8     public LinkedListDivideNode() {
9         divideInfo = new Divide();
10    }

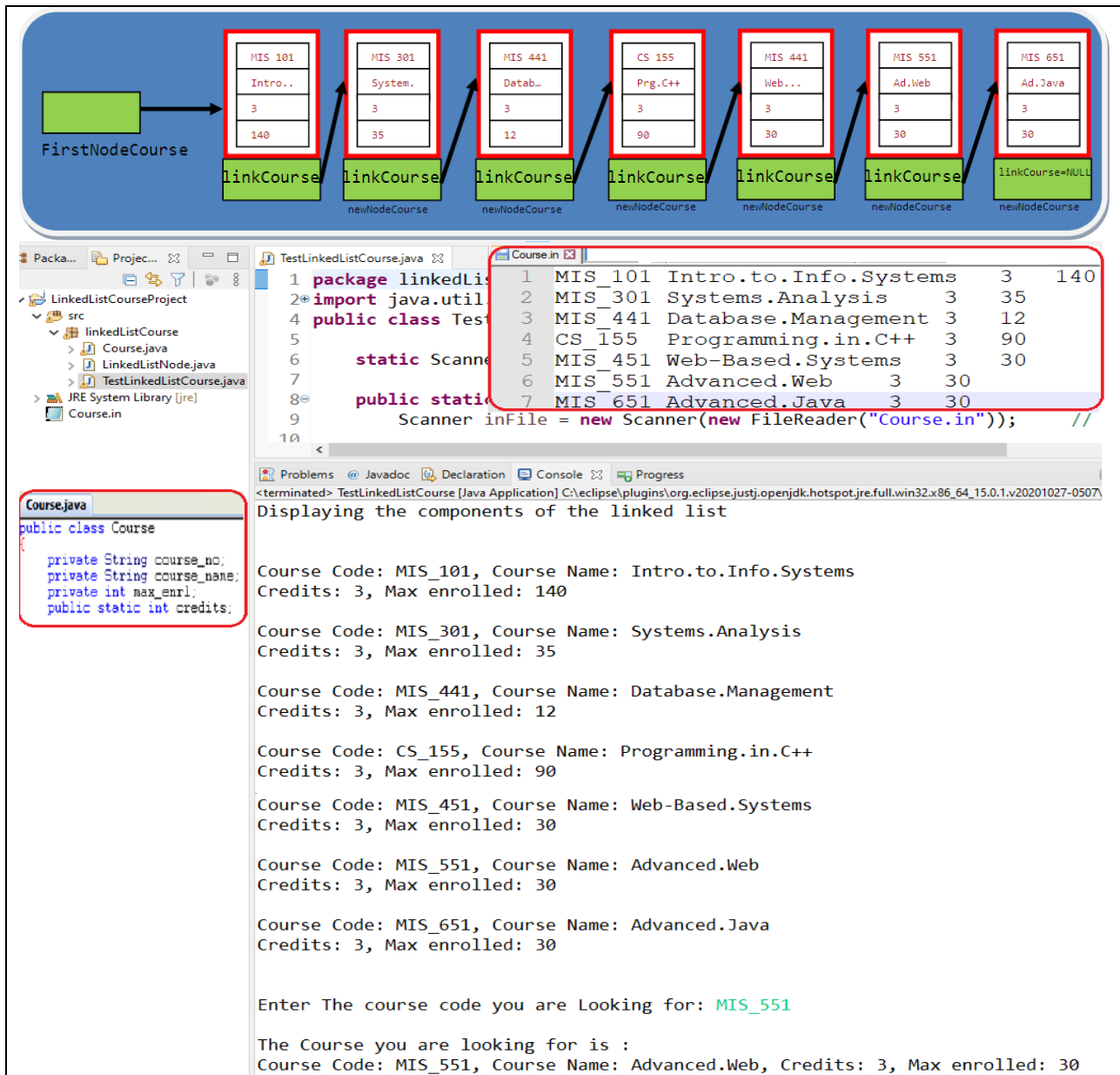
```

Displaying the components of the linked list stored from input file divide.in

X: 45.0 Y: 6.0  
X: 67.0 Y: 8.0  
X: 124.0 Y: 9.0  
X: 89.0 Y: 6.0

### 3. Application of Linked List: Storing records of *Course.in* File

- a) Create *LinkedListCourseProject*, to store records *read from* the file input *Course.in* onto a *linked list*.



**LinkedList Diagram:**

| Course Code | Course Name | Credits | Max Enrolled |
|-------------|-------------|---------|--------------|
| MIS_101     | Intro..     | 3       | 140          |
| MIS_301     | System.     | 3       | 35           |
| MIS_441     | Datab..     | 3       | 12           |
| CS_155      | Prg.C++     | 3       | 90           |
| MIS_451     | Web...      | 3       | 30           |
| MIS_551     | Ad.Web      | 3       | 30           |
| MIS_651     | Ad.Java     | 3       | 30           |

**Course.in File Content:**

```

1 MIS_101 Intro.to.Info.Systems 3 140
2 MIS_301 Systems.Analysis 3 35
3 MIS_441 Database.Management 3 12
4 CS_155 Programming.in.C++ 3 90
5 MIS_451 Web-Based.Systems 3 30
6 MIS_551 Advanced.Web 3 30
7 MIS_651 Advanced.Java 3 30

```

**Course.java Class:**

```

public class Course
{
    private String course_no;
    private String course_name;
    private int max_enrl;
    public static int credits;
}

```

**TestLinkedListCourse.java Code:**

```

1 package linkedList;
2 import java.util.*;
3
4 public class TestLinkedListCourse
5 {
6     static Scanner sc;
7
8     public static void main(String[] args)
9     {
10        Scanner inFile = new Scanner(new FileReader("Course.in"));

```

**Output:**

Displaying the components of the linked list

Course Code: MIS\_101, Course Name: Intro.to.Info.Systems  
Credits: 3, Max enrolled: 140

Course Code: MIS\_301, Course Name: Systems.Analysis  
Credits: 3, Max enrolled: 35

Course Code: MIS\_441, Course Name: Database.Management  
Credits: 3, Max enrolled: 12

Course Code: CS\_155, Course Name: Programming.in.C++  
Credits: 3, Max enrolled: 90

Course Code: MIS\_451, Course Name: Web-Based.Systems  
Credits: 3, Max enrolled: 30

Course Code: MIS\_551, Course Name: Advanced.Web  
Credits: 3, Max enrolled: 30

Course Code: MIS\_651, Course Name: Advanced.Java  
Credits: 3, Max enrolled: 30

Enter The course code you are Looking for: MIS\_551

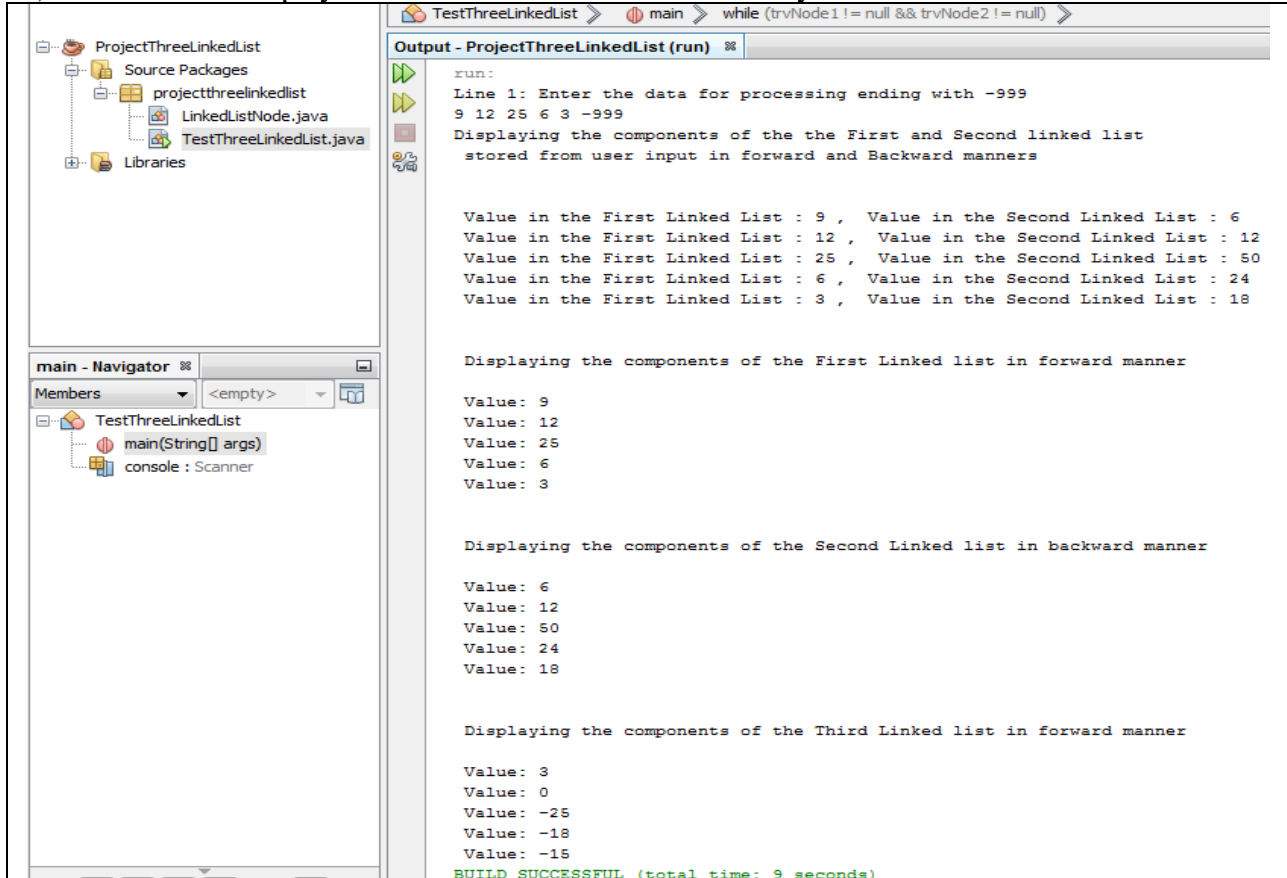
The Course you are looking for is :  
Course Code: MIS\_551, Course Name: Advanced.Web, Credits: 3, Max enrolled: 30

b) Search method and Passing Linked List Object as Parameter to a Java method

- Add Java method `searchCourses(LinkedListNode wcourse, String wcourse_code)` into `Course` class in order to perform a search operation with respect to course code.
- Add more Java Statements into your main program in order to call the search method `searchCourses(...)` implemented previously with respect to course code as user input as shown hereafter. The method `searchCourses (...)` takes a reference of the linked list as parameter and returns the reference pointing to the found Node course in the linked list.

#### 4. Linked List Project: *ProjectThreeLinkedList*

- a) Create a Java project *ProjectThreeLinkedList* in order to read input integer from the console until the user enters -999 (acts as sentinel), insert the values into the first *linked list* build in *forward* manner referenced by *headNode1* and create the second list build in *backward* referenced by *headNode2* with values equals twice the value in nodes available in the first linked list.
- b) Traverse and display the value of every linked list referenced by *headNode1* and *headNode2* as shown in Figure below.
- c) Add more Java statements while traversing the previous linked lists to build a third linked list build in *forward* manner referenced by *headNode3* with values equals to the difference between values in the first linked list and the second linked list.
- d) Traverse and display the value of linked list referenced by *headNode3* as shown hereafter.



The screenshot displays the IDE environment for the `ProjectThreeLinkedList` project. The **Project Explorer** on the left shows the project structure with `Source Packages` containing `projectthreeLinkedList`, which includes `LinkedListNode.java` and `TestThreeLinkedList.java`. The **main - Navigator** shows the `main(String[] args)` method in `TestThreeLinkedList` with a `console : Scanner` input.

The **Output - ProjectThreeLinkedList (run)** window shows the following execution results:

```
run:
Line 1: Enter the data for processing ending with -999
9 12 25 6 3 -999
Displaying the components of the the First and Second linked list
stored from user input in forward and Backward manners

Value in the First Linked List : 9 , Value in the Second Linked List : 6
Value in the First Linked List : 12 , Value in the Second Linked List : 12
Value in the First Linked List : 25 , Value in the Second Linked List : 50
Value in the First Linked List : 6 , Value in the Second Linked List : 24
Value in the First Linked List : 3 , Value in the Second Linked List : 18

Displaying the components of the First Linked list in forward manner
Value: 9
Value: 12
Value: 25
Value: 6
Value: 3

Displaying the components of the Second Linked list in backward manner
Value: 6
Value: 12
Value: 50
Value: 24
Value: 18

Displaying the components of the Third Linked list in forward manner
Value: 3
Value: 0
Value: -25
Value: -18
Value: -15
BUILD SUCCESSFUL (total time: 9 seconds)
```