CEGEP VANIER COLLEGE
CENTRE FOR CONTINUING EDUCATION
Programming Algorithms and Patterns
420-930-VA

Teacher: Samir Chebbine **Lab 2** **Jul 21, 2023**

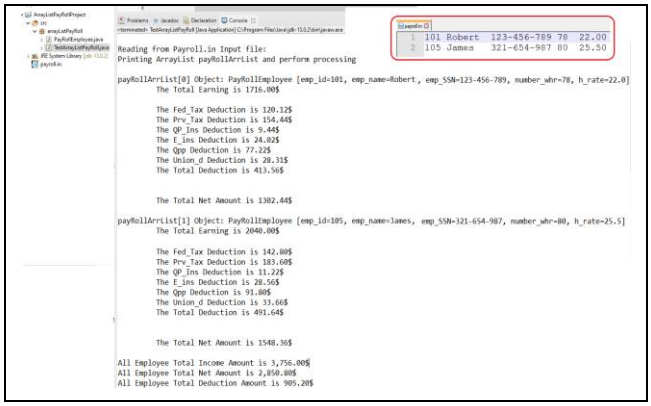
Lab 2: ArrayList and Linked List

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.docs*** 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.

a) 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.

b) 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.



   1) Add **default constructor**, setters, getters, and toString()
   2) Add methods called calculate_TotalIncome(), calculate_TotalDeduction(), calculate_TotalNetAmount() in `PayRollEmployee` class to calculate the following respectively:

   Total_Income = number_whr * 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*.

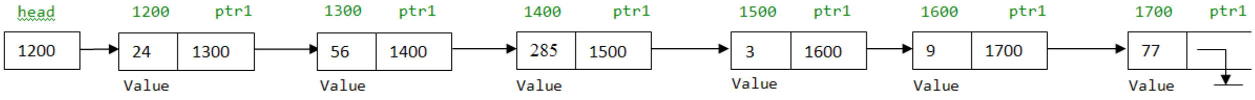
   The total Net Amount (*Net_Amount*) is calculated according to the following formula:
   ***Net_Amount = Total_Income* - Deductions**

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

2. Linked List:

a) 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.

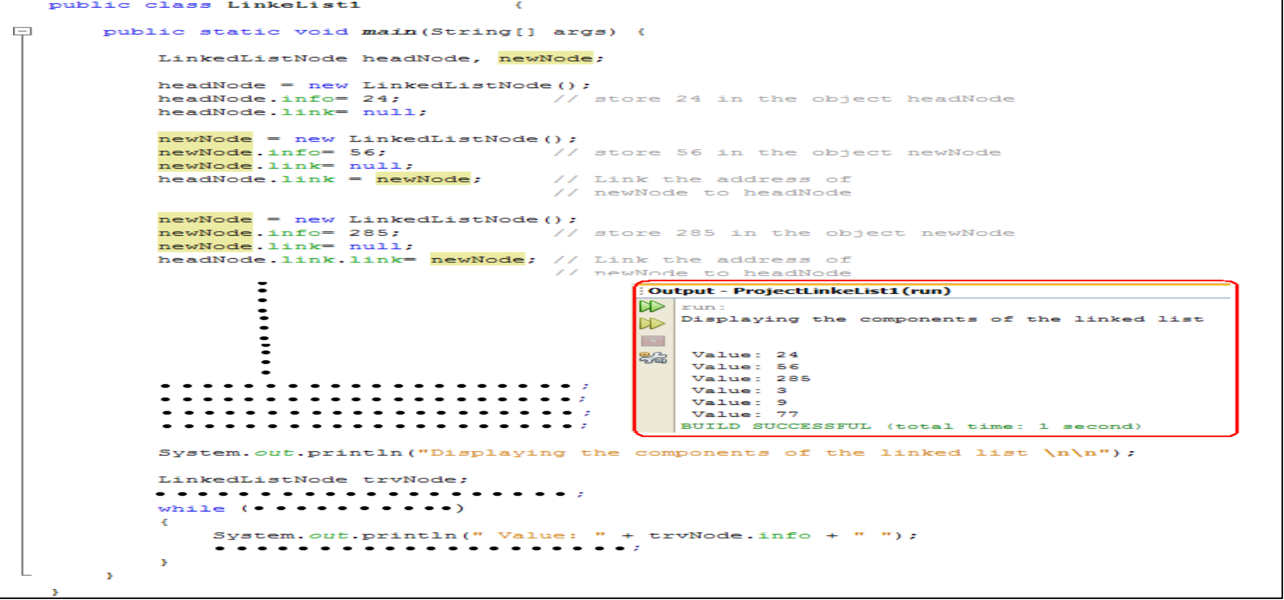


Linked list class

```java
public class LinkedListNode {

    public int info;
    public LinkedListNode link;

}
```

Linked list construction

```java
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
```

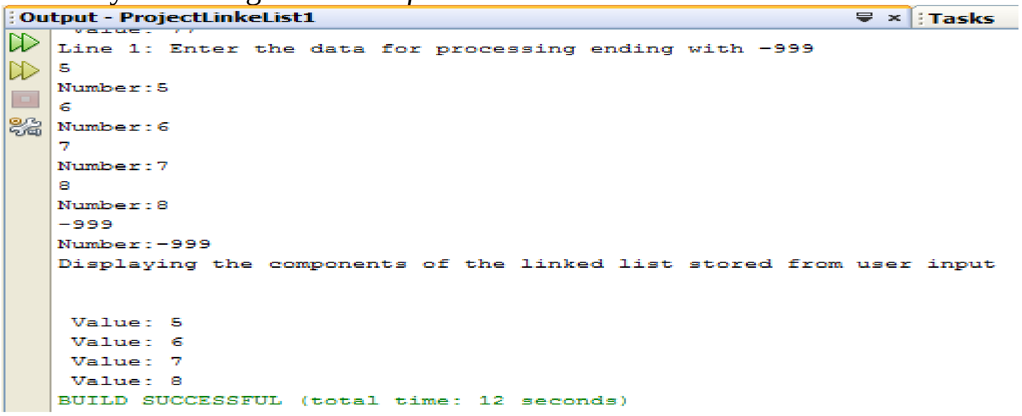


```java
public class LinkeList1                  {

    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 trvNode;
        ......................;
        while (............)
        {
            System.out.println(" Value: " + trvNode.info + " ");
            ......................;
        }
    }
}
```

Output - ProjectLinkeList1(run)

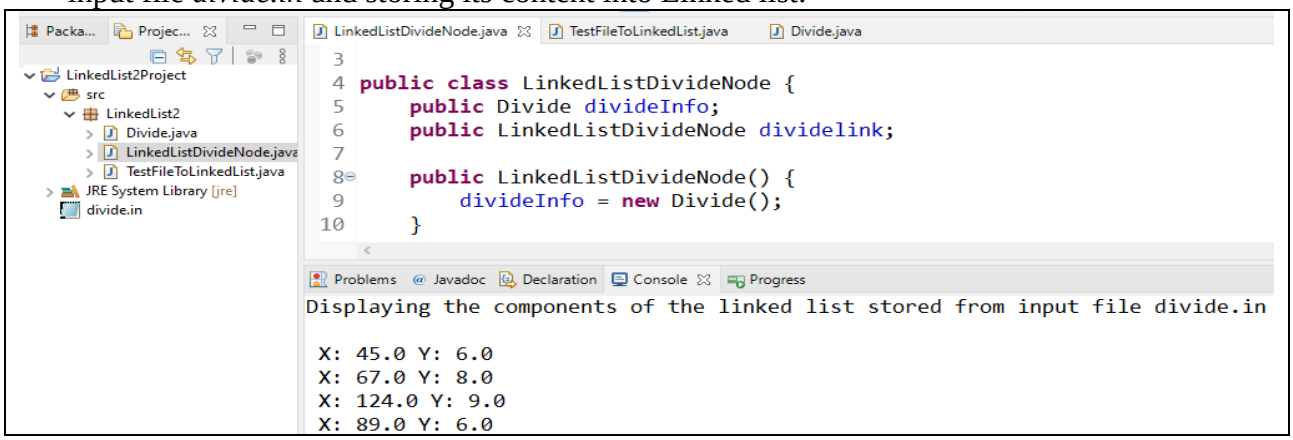
```
run:
Displaying the components of the linked list


 Value: 24
 Value: 56
 Value: 285
 Value: 3
 Value: 9
 Value: 77
BUILD SUCCESSFUL (total time: 1 second)
```

b) 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.



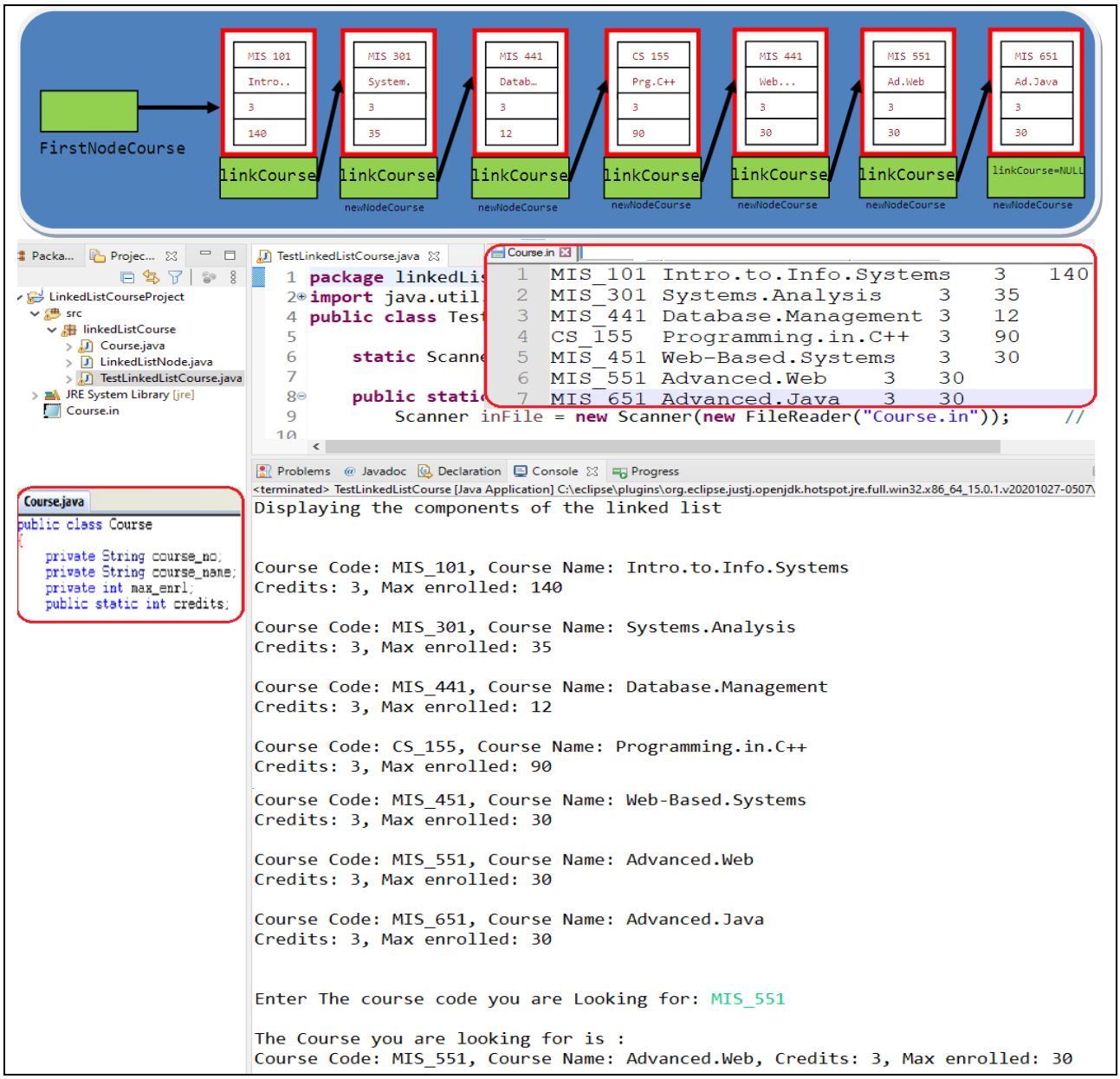
```
Line 1: Enter the data for processing ending with -999
5
Number:5
6
Number:6
7
Number:7
8
Number:8
-999
Number:-999
Displaying the components of the linked list stored from user input


 Value: 5
 Value: 6
 Value: 7
 Value: 8
BUILD SUCCESSFUL (total time: 12 seconds)
```

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



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*.



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.