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

Teacher: Samir Chebbine **Lab 2** **May 28, 2024**

Lab 2: ArrayList and Linked List

Complete all these following programs as explained during **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.

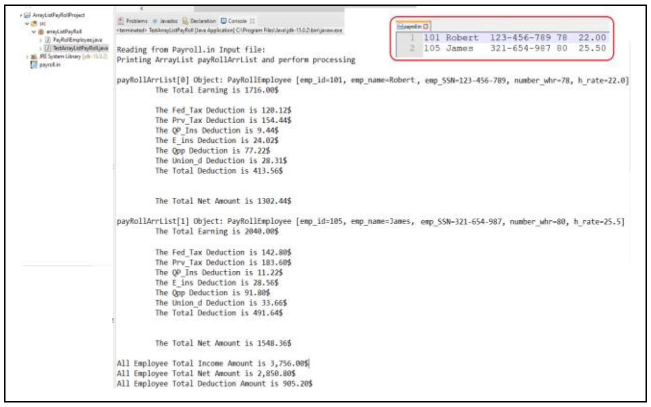
1. ArrayLists Data Structure

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:



```
<terminated> TestArrayListPayRoll [Java Application] C:\Program Files\Java\jdk-15.0.2\bin\javaw.exe
```

```
1 101 Robert   123-456-789 78  22.00
2 105 James    321-654-987 80  25.50
```

```
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 28.31$
        The Total Deduction is 413.56$


        The Total Net Amount is 1302.44$

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 491.64$


        The Total Net Amount is 1548.36$

All Employee Total Income Amount is 3,756.00$
All Employee Total Net Amount is 2,850.80$
All Employee Total Deduction Amount is 905.20$
```

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*.

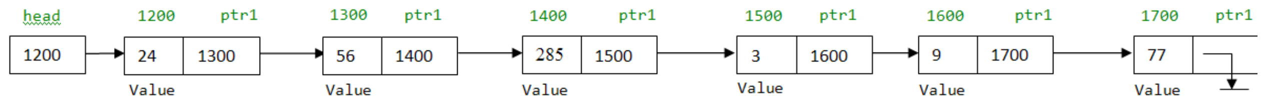
The total Net Amount (*Net_Amount*) is calculated according to the following formula:
***Net_Amount* = *Total_Income* - Deductions**

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 user-defined linked list.

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

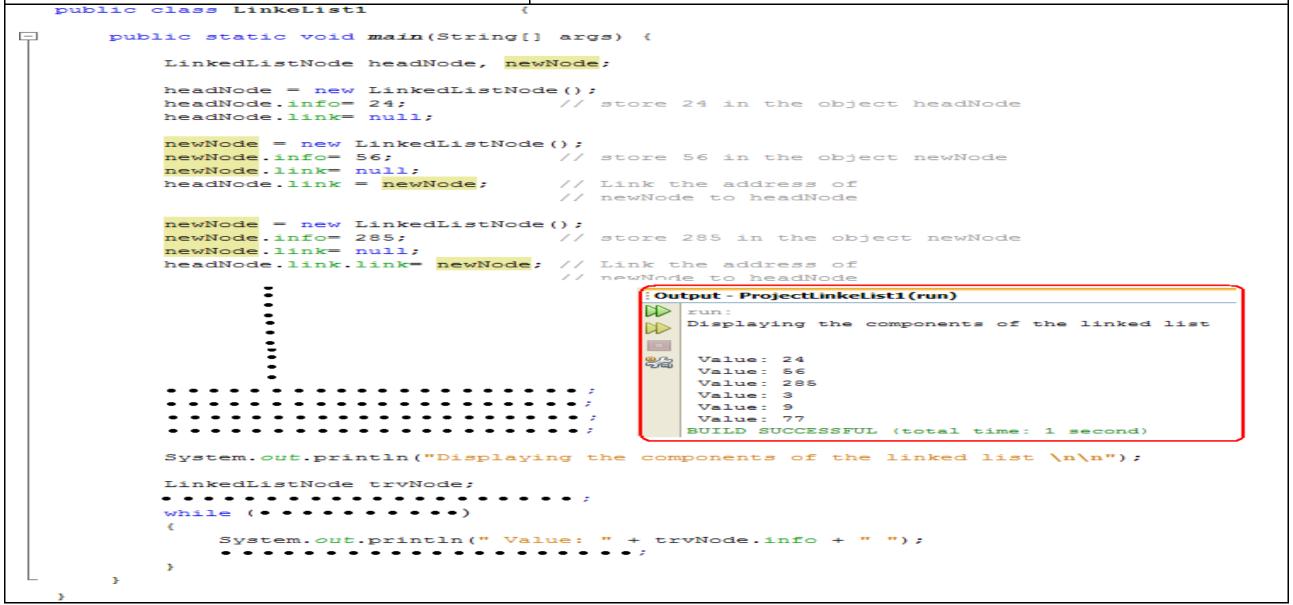


Linked list class

```
public class LinkedListNode {

    public int info;
    public LinkedListNode link;

}
```

Linked list construction

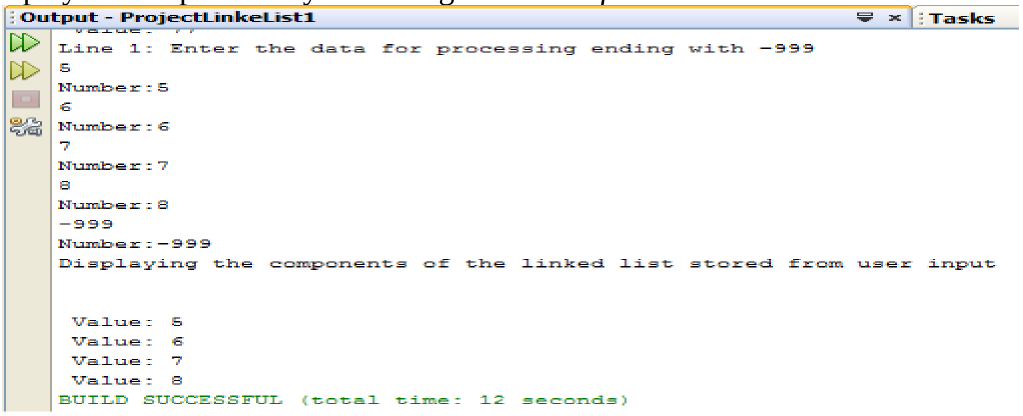
```
LinkedListNode headNode, newNode;

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
```



```
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 + " ");
            . . . . . . . . . . . . . . . . . . . . ;
        }
    }
}
```

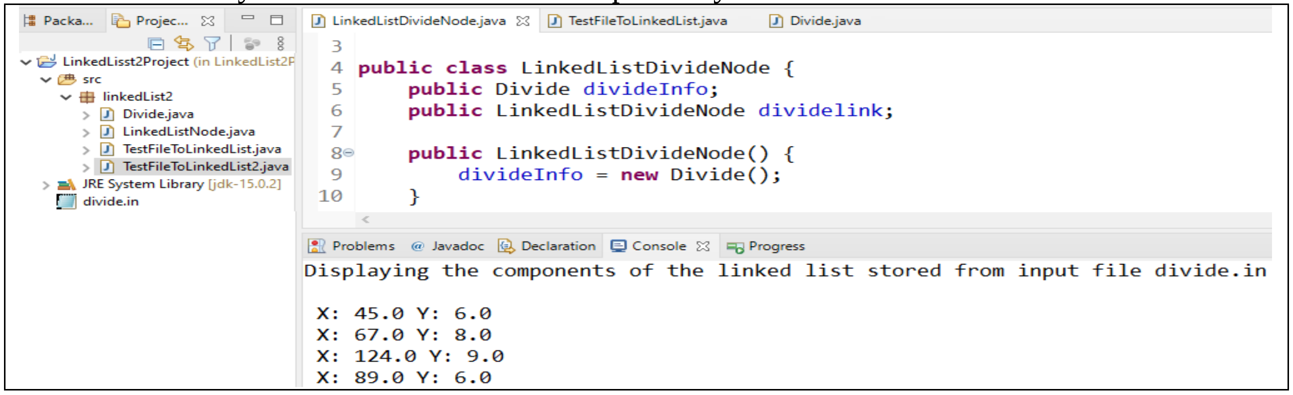
```
Output - ProjectLinkeList1(run)
run:
Displaying the components of the linked list

 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 *user-defined linked list* (build in *forward* manner), and display its components by traversing the link in *forward* manner.



```
Output - ProjectLinkeList1                    Tasks
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/ TestFileToLinkedList2.java* to read from input file *divide.in* and storing its content into user-defined and system-defined Linked list respectively.

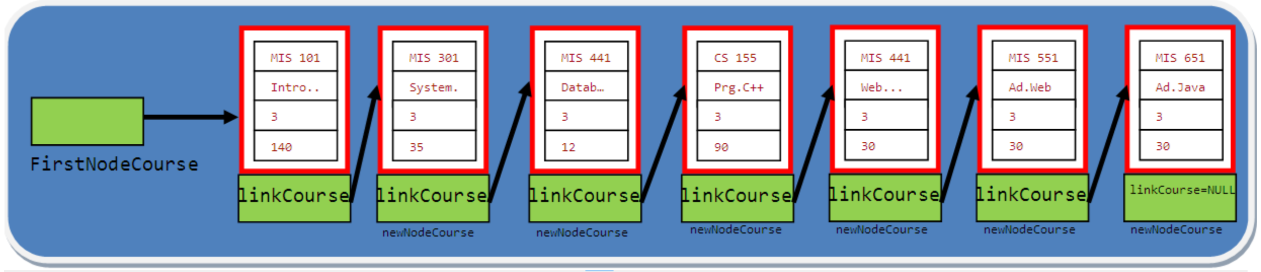

Packa...
Projec...
LinkedLisst2Project (in LinkedList2P
src
linkedList2
Divide.java
LinkedListNode.java
TestFileToLinkedList.java
TestFileToLinkedList2.java
JRE System Library [jdk-15.0.2]
divide.in
LinkedListDivideNode.java
TestFileToLinkedList.java
Divide.java
Problems
Javadoc
Declaration
Console
Progress

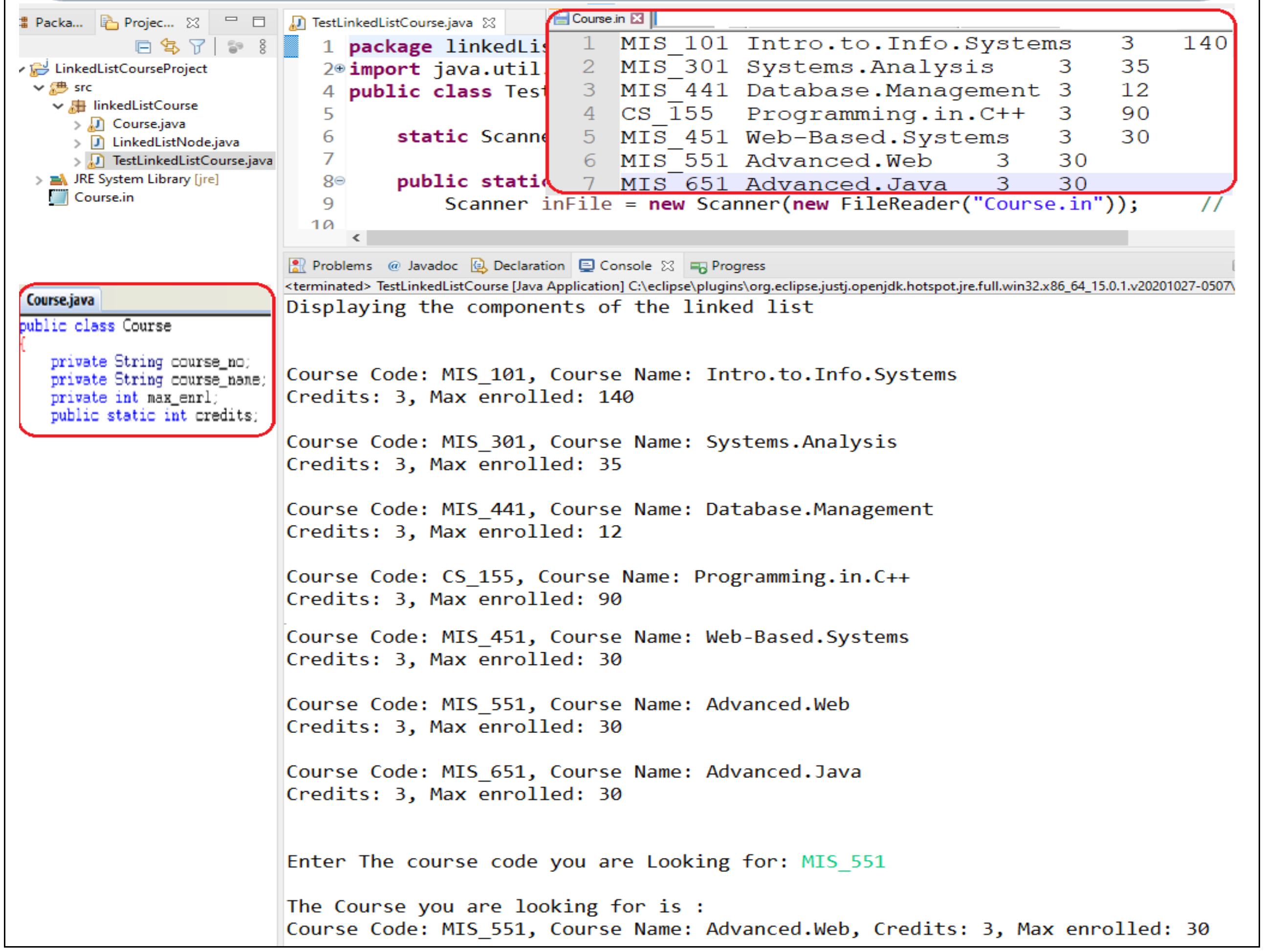
```
public class LinkedListDivideNode {
    public Divide divideInfo;
    public LinkedListDivideNode dividelink;

    public LinkedListDivideNode() {
        divideInfo = new Divide();
    }
```

```
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 *user-defined linked list.*





```
package linkedLis
import java.util
public class Tes

    static Scann

    public stati
        Scanner inFile = new Scanner(new FileReader("Course.in"));     //
```

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

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

```
<terminated> TestLinkedListCourse [Java Application] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.1.v20201027-0507\
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.

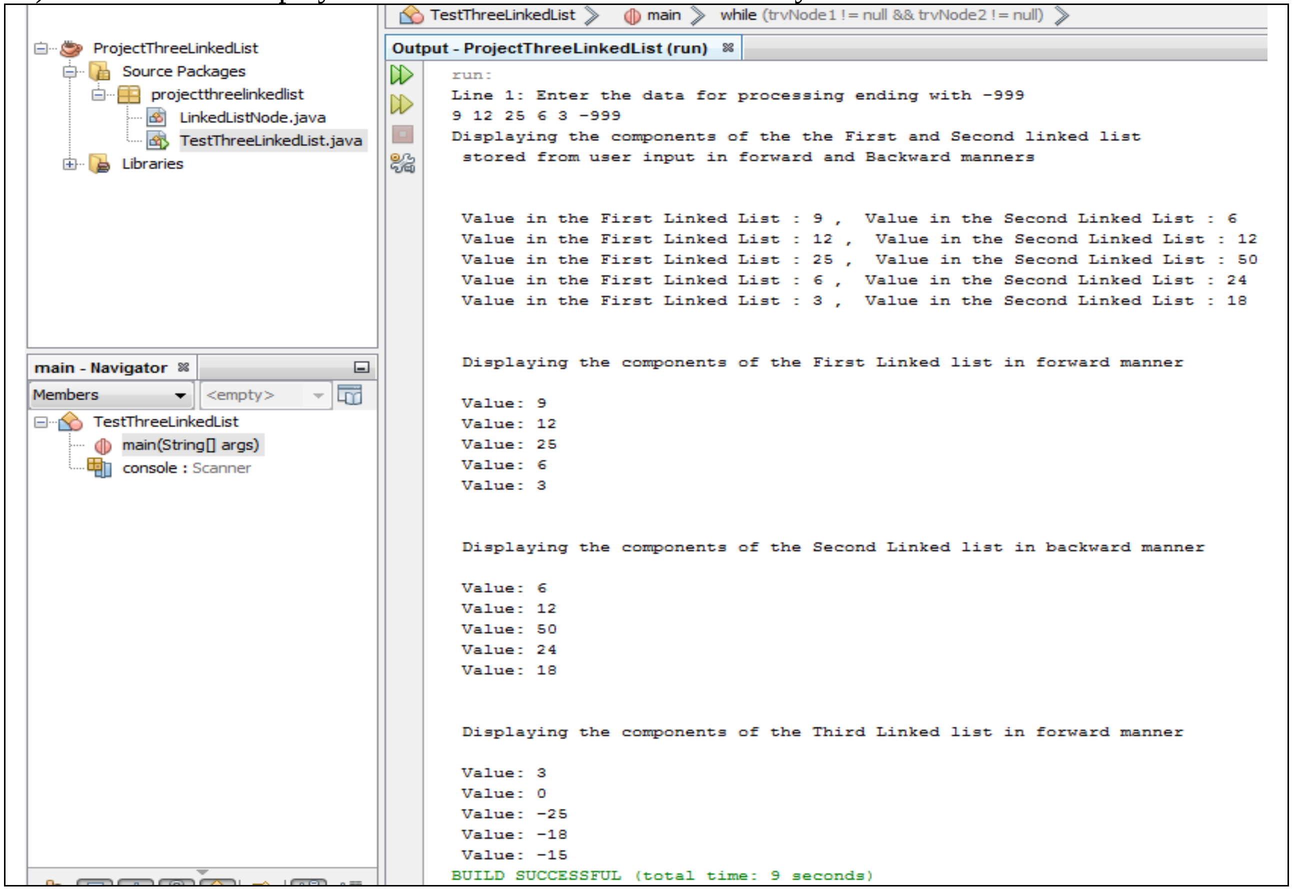
4. Linked List Project: *ProjectThreeLinkedList*

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 *user-defined linked list* build in *forward* manner referenced by *headNode1* and create the second *user-defined linked* 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 user-defined 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.



```
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)
```