**SINGLY LINKED LIST**

**Exercise 1.** Write a Java program to implement a singly linked list for the Person object, including the following requirements:

1. **Create 4 classes**: Main, MyList, Node, and Person:
   * Main: To run the program.
   * Person: To define the Person object with 2 attributes: String name and int age.
   * Node: To define a Node with 2 attributes: info and next.
   * MyList: To define the methods.
2. **Complete the following program requirements**:

public class MyList

{Node head, tail;

MyList() {head=tail=null;}

boolean isEmpty() {return(head==null);}

void clear() {head=tail=null;}

//(0)-------------------------------------------------------------

void displayYourInfo(){

- Full name:

- Class Nam:

- ID Student:

}

// (1) ----------------------------------------------------------

void addLast(Person x)

{Node q = new Node(x);

if(isEmpty()) {head=tail=q;return;}

tail.next = q;

tail = q;

}

//---------------------------------------------------------------------------

void visit(Node p) {if(p!=null) System.out.print(p.info);}

void traverse()

{Node p=head;

while(p!=null)

{visit(p);

p=p.next;

}

System.out.println();

}

void addMany(String [] a, int [] b)

{int n,i; n=a.length;

for(i=0;i<n;i++) addLast(new Person(a[i],b[i]));

}

// (2)----------------------------------------------------------------------

Node searchByName(String xName)

{Node p=head;

while(p!=null)

{if(p.info.name.equals(xName)) return(p);

p=p.next;

}

return(null);

}

// (3)----------------------------------------------------------------

void addFirst(Person x)

{

}

// (4)-----------------------------------------------------------------

void insertAfter(Node q, Person x)

{

}

// (5)------------------------------------------------------------------

void insertBefore(Node q, Person x)

{

}

// (6)------------------------------------------------------------------------

void remove(Node q)

{

}

// (7)------------------------------------------------------------------------

void remove(String xName)

{

}

// (8)--------------------------------------------------------------------------

void remove(int xAge)

{

}

// (9)---------------------------------------------------------------------------

void removeAll(int xAge)

{

}

// (10)-------------------------------------------------------------------------

Node pos(int k)

{return(null);

}

// (11)-------------------------------------------------------------------------

void removePos(int k)

{

}

// (12)--------------------------------------------------------------------

void sortByName()

{

}

// (13)----------------------------------------------------------------------

void sortByAge()

{

}

// (14)-------------------------------------------------------------------------

int size()

{return(0);

}

// (15)--------------------------------------------------------------------------

Person [] toArray()

{return(null);

}

// (16)-------------------------------------------------------------------------

void reverse()

{

}

// (17)---------------------------------------------------------------------------

Node findMaxAge()

{return(null);

}

// (18)-----------------------------------------------------------------

Node findMinAge()

{return(null);

}

// (19)--------------------------------------------------------------------

void setData(Node p, Person x)

{

}

// (20)-----------------------------------------------------------------------------

void sortByAge(int k, int h) // Sort from position k to position h (the position of the first element is 0)

{

}

// (21) -------------------------------------------------------------------------------

void reverse(int k, int h) // reverse from position k to position h (the position of the first element is 0)

{

}

}