EXAMPLE (STACK)

Example of Stack with Object Student

CLASS STUDENT

```
public class Student
{ ..... }
```

CLASS NODE

```
public class Node
{ ....  }
```

CLASS LINKEDLIST

```
public class LinkedList
{ ....  }
```

CLASS STACK

```
public class Stack extends LinkedList
{
            public Stack() { }          // constructor

            public void push (Object elem)
            { insertAtFront(elem);  }

            public Object pop ( )
            { return removeFromFront();  }

            public Object peek()
            { return getFirst(); }
} // end Stack
```

CLASS APPLICATION

```
import javax.swing.JOptionPane;
public class StackApp2
{
    public static void main(String [] args)
    {
        Stack theStack = new Stack();                 // original stack
        Stack tempStack = new Stack();                // temporary stack

        for (int i=0; i<5; i++) // to input 5 students into the list
        {  String sIdStd = JOptionPane.showInputDialog("Enter student id");
            String nameStd = JOptionPane.showInputDialog("Enter name");
            String sPart = JOptionPane.showInputDialog("Enter part");
            String sCgpa = JOptionPane.showInputDialog("Enter cgpa");
            int iIdStd = Integer.parseInt(sIdStd);
            int iPart = Integer.parseInt(sPart);
            double dCgpa = Double.parseDouble(sCgpa);
            Student stud = new Student(iIdStd, nameStd, iPart, dCgpa);
            theStack.push(stud);  }        //insert data
```

```
// to display all the students in the stack
Object data;
Student S;

while (!theStack.isEmpty())
{
    data = theStack.pop(); //delete first
    S = (Student) data; //casting
    System.out.println(S.toString()); //display
    tempStack.push(S);  // put into temporary stack
}

// restore; transfer all data from temporary stack to original stack
while (!tempStack.isEmpty())
{
   theStack.push(tempStack.pop());
}

// to demonstrate some possible operation on data in the stack
double max = -99999.99, min = 9999.99;
int part4 = 0, dList = 0, prob = 0;
Student bestStudent = null;
Student weakStudent = null;

while (!theStack.isEmpty())
{
    data = theStack.pop();   // pop from origional stack
    S = (Student) data;

    if(S.getCgpa()>max)     // find maximum cgpa
    {max = S.getCgpa();
    bestStudent = S;}

    if (S.getCgpa() < min)   // find minimum cgpa
       { min = S.getCgpa();
        weakStudent = S;}

    if (S.getCgpa() > 3.5)   // count dean's list student
        dList++;

    if (S.getPart()  == 4)   // count part 4 student
        part4++;

    if (S.getCgpa() < 1.8)   // count probation student
        prob++;

        tempStack.push(S); // store to temporary stack

}
```

```
        // display result
        System.out.println("The highest cgpa = " + max);
        System.out.println("The lowest cgpa = " + min);
        System.out.println("The number of dean's list student = "  + dList);
        System.out.println("The number of part 4 student = " +  part4);
        System.out.println("The number of probation student = " +  prob);
        System.out.println("BEST STUDENT:");
        System.out.println(bestStudent.toString());

        // restore; transfer all data from temporary stack to original stack
        while (!tempStack.isEmpty())
        {
           theStack.push(tempStack.pop());
        }

     } // main
  }  // StackApp
```