Lucrarea nr. 7 - Structuri de date de tip stivă şi coadă implementate prin liste înlănțuite

Breviar teoretic

În cadrul lucrarii anterioare (Lucrarea nr.5) *Structuri de date de tip stivă şi coadă* a fost implementată o stivă prin utilizarea unui tablou. În acel exemplu, operațiile specifice stivei push şi pop au fost implementate prin operații cu elementele tabloului, de tipul:

`Tablou[++top] = data;` (echivalentul lui `push()`)
`data = arr[top--];` (echivalentul lui `pop()`)

Similar se poate implementa o stivă prin utilizarea listelor înlanțuite, folosind însă de această dată metode de tipul:

```
Lista.insereazaPrimul(data)
data = Lista.stergePrimul()
```

În mod similar procedurii descrise anterior se poate implementa şi o coadă utilizând liste înlănțuite.

Pentru clarificarea acestor notiuni inserați, rulati şi corectati (daca este cazul) următoarele aplicatii.

Aplicatie pentru implementarea unei stive utilizand liste simplu inlantuite: realizati si rulati urmatorul proiect ***LinkStackApp***:

```
import java.io.*;

class Link
{
public double dData;
public Link next;

public Link(double dd)
{ dData = dd; }

public void displayLink()
{ System.out.print(dData + " "); }
}

class LinkList
{
private Link first;

public LinkList() // constructor
{ first = null; }
```

```
public boolean isEmpty()
{ return (first==null); }

public void insertFirst(double dd)
{ Link newLink = new Link(dd);
  newLink.next = first;
  first = newLink;
  System.out.println(“A fost inserat pe prima pozitie elementul
”+first.dData);}

public double deleteFirst()
{ Link temp = first;
  System.out.println(“A fost extras de pe prima pozitie elementul
”+first.dData);
  first = first.next;
  return temp.dData;
}

public void displayList()
{ Link current = first;
  while(current != null)
      { current.displayLink();
       current = current.next;
      }
  System.out.println("");}
}

class LinkStack
{private LinkList theList;

public LinkStack()
{ theList = new LinkList();}

public void push(double j)
{ theList.insertFirst(j);}

public double pop()
{ return theList.deleteFirst();}

public boolean isEmpty()
{ return ( theList.isEmpty() );}

public void displayStack()
{ System.out.print("Continutul Stivei (Primul --> Ultimul): ");
 theList.displayList(); }
}

class LinkStackApp
{ public static void main(String[] args) throws IOException
```

```
 {    LinkStack theStack = new LinkStack();
      theStack.push(20);
      theStack.push(40);
      theStack.displayStack();
      theStack.push(60);
      theStack.push(80);
      theStack.displayStack();
      theStack.pop();
      theStack.pop();
      theStack.displayStack();
  }
}
```

Aplicatie pentru implementarea unei structuri de date de tip coada, utilizand liste simplu inlantuite: Introduceți, rulați şi corectati (dacă este cazul) următorul proiectul ***LinkQueue***:

```
import java.io.*;

class Link
{ public double dData;
  public Link next;

public Link(double d)
{ dData = d; }

public void displayLink()
{ System.out.print(dData + " "); }
}

class FirstLastList
{ private Link first;
  private Link last;

public FirstLastList()
{ first = null;
  last = null;
 }

public boolean isEmpty()
{ return first==null; }

public void insertLast(double dd)
{ Link newLink = new Link(dd);
  System.out.print(" Elementul introdus in coada implementata este "+
Link.dData);
  if(isEmpty())
     first = newLink;
  else
     last.next = newLink;
```

```
     last = newLink;
}

public double deleteFirst()
{ double temp = first.dData;
  System.out.print(" Elementul sters din coada implementata este
"+first.dData);
  if(first.next == null)
  last = null;
  first = first.next;
  return temp;
 }

public void displayList()
{Link current = first;
 while(current != null)
      { current.displayLink();
      current = current.next;
      }
 System.out.println("");
 }
}

class LinkQueue
{private FirstLastList theList;

public LinkQueue()
{ theList = new FirstLastList(); }

public boolean isEmpty()
{ return theList.isEmpty();}

public void insert(double j)
{ theList.insertLast(j); }

public double remove()
{ return theList.deleteFirst(); }

public void displayQueue()
{ System.out.print("Continutul cozii (Primul element --> Ultimul element):
");
theList.displayList();}
}

class LinkQueueApp
{ public static void main(String[] args) throws IOException

      {
      LinkQueue theQueue = new LinkQueue();
```

```
        theQueue.insert(20);
        theQueue.insert(40);
        theQueue.displayQueue();
        theQueue.insert(60);
        theQueue.insert(80);
        theQueue.displayQueue();
        theQueue.remove();
        theQueue.remove();
        theQueue.displayQueue();
        }
}
```

Probleme propuse

Pentru proiectul *LinkStackApp*:

1. Modificați programul astfel încât elementele stivei să fie String-uri!
2. Modificați programul astfel încât stiva să accepte atât şiruri de caractere (String), cât şi numere întregi. Dacă credeți că este nevoie puteți folosi selectori de tipul **`args[0]`** pentru programul java.

Pentru proiectul *LinkQueue*:

1. Modificați programul astfel încât elementele cozii să fie String-uri!
2. Realizați un program complex care să poate folosi listele inlănțuite atât ca structuri de tip coadă, cât şi ca structuri de tip stivă, prin existența în cadrul programului atât a funcțiilor de tip insert şi remove, dar şi a functiilor push, respectiv pop.