Lucrarea nr. 6 - Liste înlănțuite. Operaţii cu liste

Breviar teoretic

O listă inlanțuită este o structură de date în care elementele care o compun sunt aranjate într-o anumita ordine. Chiar dacă aparent lista este similară cu organizarea de tip tablou, ordinea elementelor dintr-o listă este determinata de un indicator, pointer, pe care fiecare element al listei îl contine în structura sa. Prin urmare elementele din listă trebuie sa fie obiecte. Listele inlantuite permit o implementare simpla şi flexibilă pentru structuri de date dinamice. În cadrul operatiilor care se pot realiza cu structurile de tip listă, putem enumera toate operatiile clasice: adaugă element, elimină element, caută element, etc.
Scopul acestei prime lucrări din seria celor care işi propun aprofundarea conceptului de tip listă, este lista simplu inlanțuită.

O listă poate însă fi de mai multe feluri:

- listă simplu înlanțuită
- lista dublu înlanțuită
- listă sortată
- listă liniară
- listă circulară.

Din punct de vedere principial, diferența dintre o listă simplu inlanțuită şi una dublu înlanțuită constă din numarul şi semnificația pointerilor - indicatorilor - care specifică caracteristicile de selectare ale elementelor listei.
Astfel o lista dublu înlanțuită dispune de 3 câmpuri :

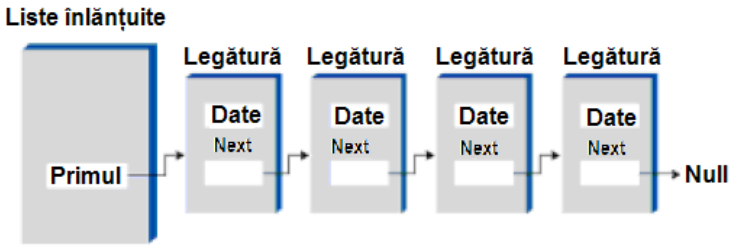
•back – referința către elementul anterior din listă
•value – valoarea elementului (cheie) de la poziția curentă în listă
•next – referința către urmatorul element din listă
Pentru o listă simplu înlanțuită se omite referința către elementul anterior(back), dispunând numai de informații referitoare doar la urmatorul element.

Dacă o listă este sortată crescator elementul din capul listei este elementul minim, iar elementul maxim ultimul din lista. Dacă o listă este nesortată cheile pot avea diferite valoari, nerespectând o ordine crescătoare sau descrescătoare.

O listă simplu înlanțuită este liniară dacă ultimul element din listă are referintă nulă către următorul element. O listă dublu înlanțuită este liniară dacă primul element are referința nulă către elementul precedent iar ultimul element din lista are referintă nulă către urmatorul element din listă.

O listă simplu înlanțuită este circulară dacă ultimul element din listă pointeaza către primul din lista. O listă dublu înlanțuită este circulară dacă primul element din listă pointează către ultimul element din listă, iar ultimul element din listă are referință către primul.

Aplicatii privind liste simplu înlănțuite: **Creați şi implementați proiectul *ListaAp1*:**



Următorul program permite inserarea unui element la începutul unei liste, ştergerea unui element de la începutul listei şi afişarea elementelor dintr-o listă de la sfarşitul listei la începutul acesteia.

```
class StructuraLista
{public int iDentif;
 public double dData;
 public StructuraLista next;
//urmatorul element din lista

public StructuraLista(int id, double dd)
//constructor
{ iDentif = id;
  dData = dd; }

public void afiseazaElement()
{ System.out.print("{" + iDentif + ", " + dData + "} "); }
                  }

class ElementLista
{ private StructuraLista Primul;

public void ElementLista()
{ Primul = null; }

public boolean ListaGoala()
{ return (Primul==null); }

public void insereazaPrimul(int id, double dd)
{
StructuraLista oLista = new StructuraLista(id, dd);
oLista.next = Primul;
Primul = oLista;
}

public  StructuraLista stergePrimul()
{StructuraLista temp = Primul;
 Primul = Primul.next;
 return temp;
}

public void afiseazaLista()
{
 System.out.print("Lista (de la Primul -->la Ultimul): ");
 StructuraLista curent = Primul;
```

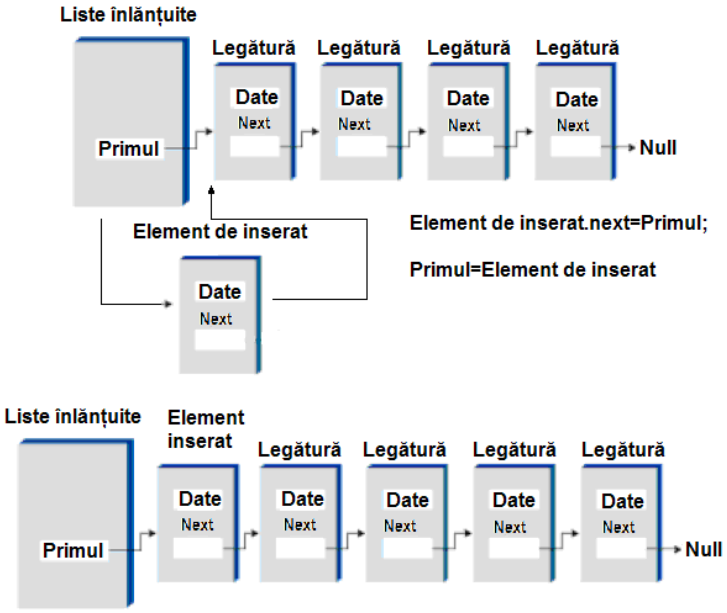
```
 while(curent != null)
  { curent.afiseazaElement();
    curent = curent.next;
  }
 System.out.println("");
 }
}

class ListaAp1
{public static void main(String[] args)
{ElementLista LegLista = new ElementLista();
 LegLista.insereazaPrimul(1,32.21);
 LegLista.insereazaPrimul(2,8.69);
 LegLista.insereazaPrimul(3,4.22);
 LegLista.insereazaPrimul(4,8.36);
 LegLista.afiseazaLista();

 while(!LegLista.ListaGoala())
   {
StructuraLista oLegatura = LegLista.stergePrimul();
      System.out.print("Sters ");
      oLegatura.afiseazaElement();
      System.out.println("");
    }
  LegLista.afiseazaLista();
 }
}
```

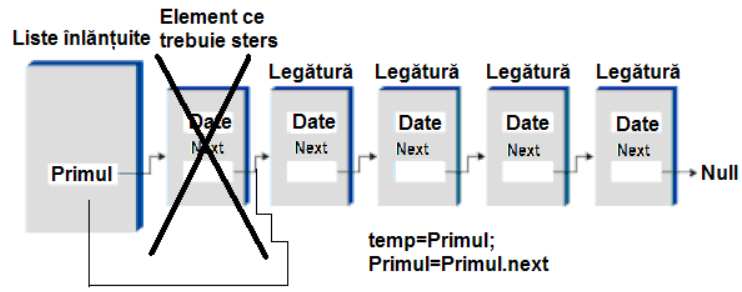
Observație:
Metoda **`insereazaPrimul()`** este cea mai simplă metodă de inserare, deoarece lista este poziționată deja la primul element. Metoda va trebui să inițieze formarea unui nou element al listei **`oLista`**, al cărui câmp **`next`** va trebui inițializat cu vechea referință **`Primul`** , în timp ce noua locație pentru elementul **`Primul`** (care este de fapt capătul listei create) va fi atribuită noului element introdus, **`oLista`**.

```
public void insereazaPrimul(int id, double dd)
{
StructuraLista oLista = new StructuraLista(id, dd);
oLista.next = Primul;
Primul = oLista;
}
```



Ştergerea primului Element – **`stergePrimul()`**
Se realizează în mod invers inserării în prima locație. În principiu va trebui declarat pe poziția **`Primul`**, elementul din listă care se află în referința **`next`** a Primului element din listă:

```
public  StructuraLista stergePrimul()
{
  StructuraLista temp = Primul;
  Primul = Primul.next;
  return temp;
}
```



Diferența dintre acest tip de organizare a datelor şi tipurile cunoscute: liste, stive, vectori se datorează faptului că localizarea informației nu se mai poate realize prin intermediul unui indice, ci prin poziția relativă pe care o ocupă față de un capăt al listei (Primul).

Identificarea şi ştergerea unei anumite înregistrări

Pe baza programului exemplu anterior, realizați următoarele modificări:

- introduceți metoda:

```
public StructuraLista Gaseste(int cheie)
{
 StructuraLista curent = Primul;
 while(curent.iDentif != cheie)
    { if(curent.next == null)
       {System.out.println("Nu exista un element cu
        cheia "+cheie);
        return null;}
      else
        {curent = curent.next;
         System.out.println("A fost gasit elemental  cu cheia "+cheie);}

     }
  return curent;
}
```

- introduceți metoda:

```
public StructuraLista sterge(int cheie)
{
 StructuraLista curent = Primul;
 StructuraLista anterior = Primul;

 while(curent.iDentif != cheie)
     { if(curent.next == null)
         return null;
       else
         {anterior = curent;
          curent = curent.next;
          }
      }
      if(curent == Primul)
        Primul = Primul.next;
      else
        anterior.next = curent.next;
        return curent;
 }
```

- în cadrul structurii programului principal introduceți următoarele linii:

```
StructuraLista f = LegLista.Gaseste(4);
if(f != null)
 System.out.println("A fost gasit elementul cu cheia  " + f.iDentif);
else
 System.out.println(" Nu exista o astfel de  inregistrare! ");
 StructuraLista d = LegLista.sterge(2);
if( d != null )
 System.out.println(" A fost stearsa inregistrarea cu cheia "  + d.iDentif);
else
 System.out.println(" Nu exista nici o astfel de inregistrare! ");
```

Aplicatii referitoare la structurile de date de tip lista simplu inlantuita: *Listă cu două capete.* Creați şi implementați proiectul *ListaDouaCapete*:


```
class StructuraLista
{ public double dData;
  public StructuraLista next;

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

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

class ElementLista
{ private StructuraLista Primul;
  private StructuraLista Ultimul;

public void ElementLista()
{ Primul = null;Ultimul=null; }

public boolean ListaGoala()
{ return (Primul==null); }

public void insereazaPrimul(double dd)
{StructuraLista oLista=new StructuraLista(dd);
If(ListaGoala())
Ultimul=oLista;
oLista.next = Primul;
Primul = oLista;  }

public void insereazaUltimul(double dd)
{StructuraLista oLista=new StructuraLista(dd);
 If(ListaGoala())
   Primul=oLista;
 else
   Ultimul.next=oLista;
   Ultimul = oLista;
 }


public  double stergePrimul()
{ double temp = Primul.dData;
  If(Primul.next==null)
    Ultimul=null;
    Primul = Primul.next;
    return temp;
 }
```

```
public void afiseazaLista()
{ System.out.print("Lista (de la Primul -->la Ultimul): ");
  StructuraLista curent = Primul;
  while(curent != null)
   { curent.afiseazaElement();
     curent = curent.next;
   }
  System.out.println("");
 }
}

class ListaDouaCapete
{ public static void main(String[] args)
{ ElementLista LegLista = new ElementLista();
  LegLista.insereazaPrimul(22);
  LegLista.insereazaPrimul(44);
  LegLista.insereazaPrimul(66);
  LegLista.insereazaUltimul(11);
  LegLista.insereazaUltimul(33);
  LegLista.insereazaUltimul(55);
  LegLista.afiseazaLista();
  LegLista.stergePrimul();
  LegLista.stergePrimul();
  LegLista.afiseazaLista();
}
}
```

Probleme propuse

Pentru proiectul *ListaAp1:*

1. Modificati programul astfel incat sa folosiți ca elemente de tip date şiruri de caractere!
2. Construiți o metodă care să afişeze elementul i din cadrul unei liste, acest index fiind preluat ca **args[0].**
3. Realizați următoarele modificări: preluați poziția de căutare ca **args[0],** respective poziția de ştergere ca **args[1].**
4. Modificați programul astfel încât cheia de căutare să fie localizată în zona de date a întregistrării.

Pentru proiectul *ListaDouaCapete:*

1. Realizați un program în care câmpul de date sa fie de tip String. Introduceți în structura înregistrării şi un identificator, alături de câmpul de tip data.
2. Cuplați cele două metode **`insertPrimul`** şi **`insertUltimul`** într-o singură metodă, **`insert`**, care să primească un argument suplimentar, prin care să se realizeze inserarea în prima sau ultima locație, în funcție de respectivul indicator. Folosiți această metodă într-un nou program.