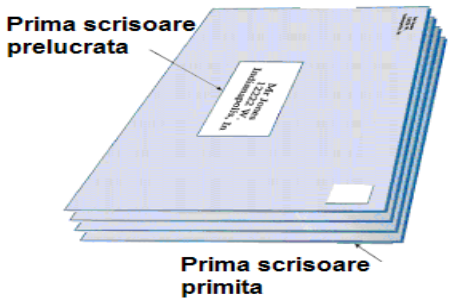
# Lucrarea nr. 5 - Structuri de date de tip stivă şi coadă

## Breviar teoretic

Pentru a intelege ce inseamna o stiva să ne gândim la o analogie cu prelucarea unui pachet de scrisori primite.

Fiecare scrisoare pe care o primim o aşezăm una peste cealaltă, urmând a le citi în clipa în care avem puțin timp liber.

Citirea teancului de scrisori se realizează de la ultima scrisoare primită, către prima, această abordare fiind de fapt o abordare de tip stivă: primul intrat, ultimul servit **– First In Last OUT – FILO.**




O altă posibilitate este în întoarcerea întregului set, invers, ca prima scrisoare primită, ca fiind scrisoarea cea mai apropiată şi parcurgerea astfel a setului de scrisori. Această abordare este de tip listă: primul intrat, primul servit – **First In First OUT – FIFO**.

Sigur că mai sunt posibile şi alte abordări, de pildă înainte de a realiza procesarea scrisorilor se va face o triere a lor pe baza importanței acordate, lucru care va duce la alcătuirea unei liste cu priorități.

***Concluzionand, putem defini Stiva drept o structura de date abstracta pentru care atat operatia de inserare a unui element cat si operatia de extragere a unui element se realizeaza de la un singur capat,denumit varful stivei.***

Operațiile de bază într-o listă sau stivă sunt:

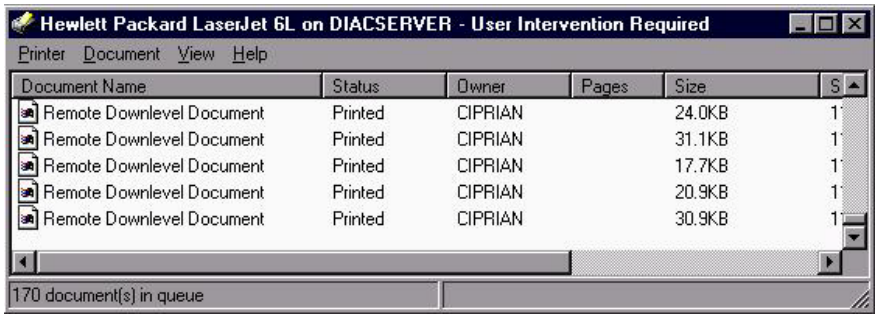
- Realizarea formației de tip coadă sau listă – alocarea zonei de memorie - new
- introducere element în formația stivă sau listă – push
- extragerea elementului din formația stivă sau listă – pop
- citirea informației din vârful listei,stivei fără a o extrage – peek
- identificarea dimensiunii stivei – Stack Size

**Coada**

***Principial Coada este definită tot ca o structura de date abstractă pentru care operatia de inserare a unui element se realizeaza la un capat, in timp ce operatia de extragere a unui element se realizează de la celalalt.***

Acest tip de structura are cea mai buna explicatie in viata cotidiana a fiecaruia: toata lumea a stat la coada. O situatie in care sunt mai multe cereri de acces la o resursa unica, implica formarea unei "linii de asteptare". Regula care trebuie respectata pentru deservirea acestui sistem de organizare este ca cererile

trebuie sa fie efectuate in ordinea sosirii. Prin urmare coada functionează dupa principiul FIFO(first in first out).




Retinand faptul că, în cadrul unei structuri de tip coadă informațiile trebuie prelucrate exact în ordinea in care au fost retinute, un bun exemplu de organizare de tip coada il reprezinta imprimanta,cand un utilizator face mai multe cereri de imprimare in acelasi timp. Cererile de tiparire sunt memorate intr-o coada de tiparire -Queue - in timp ce imprimanta le va tipari in ordinea in care vor aparea.
Coada dispune de următoarele metode:

- metoda de alocare a memoriei necesare implementării cozii – **`new`**
- metoda de inserare a unui element – **`insert()`**
- metoda de extragere a unui element din coadă – **`remove()`**
- metoda de vizualizare a primului element **`peekFront()`**

Pentru o completa fundamentare a acestor concepte, abordarea in cadrul acestei lucrari, va fi similara aboradarii anterioare - rularea de programe de tip Aplicatie si intelegerea modalitatii de lucru a acestora.

***Aplicatii pentru exemplificarea conceptului de stiva:*** Introduceți şi rulați proiectul StivaApp:

```
import java.io.*;

class Stiva
{ private int maxSize;
  private double[]StivaSir;
  private int top;

 public Stiva(int s)
   { maxSize = s;
     StivaSir = new double[maxSize];
     top = -1;  }

 public void push(double j)
   { StivaSir[++top] = j; }

 public double pop()
   { return StivaSir[top--]; }

 public double peek()
      { return StivaSir[top];}
```

```
 public boolean StivaGoala()
      { return (top == -1); }

 public boolean StivaPlina()
    { return (top == maxSize-1);}
}

class StivaApp
{ public static void main(String[] args)
  { Stiva OStiva = new Stiva(10);
    OStiva.push(20);
    OStiva.push(40);
    OStiva.push(60);
    OStiva.push(80);

    while( !OStiva.StivaGoala ())
      { double value = OStiva.pop();
        System.out.print(value);
        System.out.print(" ");
       }
    System.out.println("");
   }
}
```

***Aplicatie:*** *Exemplu de program care utilizează o stivă: programul primeşte de la consolă un şir de caractere, iar la apăsarea tastei Enter afişează şirul în ordine inversă.*
*Introduceți şi rulați proiectul InversareChar:*

```
import java.io.*;

class Stiva
{     private int maxSize;
      private char[] StivaSir;
      private int top;

      public Stiva(int max)
      {maxSize = max;
      StivaSir = new char[maxSize];
      top = -1;}

      public void push(char j)
      { StivaSir[++top] = j;}

      public char pop()
      { return StivaSir[top--];}

      public char peek()
```

```
      {return StivaSir[top];}

      public boolean StivaGoala()
      {return (top == -1);}
}

class inverseaza
{private String input;
 private String output;

public inverseaza(String in)
{ input = in; }

public String invers()
{int stackSize = input.length();
 Stiva OStiva = new Stiva(stackSize);
 for(int j=0; j<input.length(); j++)
      {char ch = input.charAt(j);
       OStiva.push(ch);
      }
 output = "";
 while(!OStiva.StivaGoala())
      { char ch = OStiva.pop();
        output = output + ch;
      }
 return output;
 }
}

class InversareChar
{ public static void main(String[] args) throws IOException
  { String input, output;
    while(true)
      { System.out.print("Introduceti sirul de
        caractere: ");
        System.out.flush();
        input = getString();
        if(input.equals(""))
        break;
    inverseaza theReverser = new inverseaza(input);
        output = theReverser.invers();
        System.out.println("Sirul inversat are
        forma: " + output);
       }
   }

public static String getString() throws IOException
 {InputStreamReader isr=new
  InputStreamReader(System.in);
```

```
 BufferedReader br = new BufferedReader(isr);
 String s = br.readLine();
 return s;
 }
}
```

***Aplicatie pentru exemplificarea conceptului de coada:*** Creați şi implementați proiectul *CoadaAp*:

```
import java.io.*;
class Coada
{
private int dimensiune;
private int[] CoadaSir;
private int primul;
private int ultimul;
private int nrCelule;

public Coada(int s)
{dimensiune = s;
CoadaSir = new int[dimensiune];
primul = 0;
ultimul = -1;
nrCelule = 0;}

public void insert(int j)
{
if(ultimul == dimensiune-1)
ultimul  = -1;
CoadaSir[++ultimul] = j;
nrCelule++;
}

public int remove()
{
int temp = CoadaSir[primul++];
if(primul == dimensiune)
primul = 0;
nrCelule--;
return temp;
}

public int peekFront()
{return CoadaSir[primul];}

public boolean CoadaGoala()
{ return (nrCelule==0); }

public boolean CoadaPlina()
{ return (nrCelule==dimensiune);}
```

```
public int size()
{return nrCelule;}
}

class CoadaAp
{
public static void main(String[] args)
{Coada OCoada = new Coada(5);
 OCoada.insert(10);
 OCoada.insert(20);
 OCoada.insert(30);
 OCoada.insert(40);
 OCoada.remove();
 OCoada.remove();
 OCoada.remove();
 OCoada.insert(50);
 OCoada.insert(60);
 OCoada.insert(70);
 OCoada.insert(80);

 while( ! OCoada.CoadaGoala() )
 { int n = OCoada.remove();
   System.out.print(n);
   System.out.print(" ");
  }
  System.out.println("");
 }
}
```

## Probleme propuse

**Pentru proiectul *StivaApp***

1. Sunt folosite toate metodele? Dacă nu, modificați programul astfel încât să folosiți toate metodele create.
2. Modificați programul astfel încât să realizați ordonarea crescătoare a elementelor şirului utilizând una dintre metodele studiate în cadrul laboratorului anterior. Stocarea se va realiza în stivă, urmând ca apoi extragerea şi afişarea elementelor să se realizeze utilizând structura program anterioară. (această structură poartă numele de stivă ordonată).

**Pentru proiectul *InversareChar***

1. Modificați programul astfel încât, la contorizarea a 3 caractere programul să afişeze automat şirul cu ordinea inversată.
2. Modificați programul astfel încât acesta să realizeze contorizarea caracterelor şirului introdus şi să îl afişeze.

**Pentru proiectul *CoadaAp***

1. Modificati comenzile de tip insert, remove şi peekFront prin inserarea unor mesaje text care să indice operatiile realizate în interiorul metodelor.

2. Modificați programul astfel încât la extragerea informației din coadă tot conținutul să coboare cu o locație.

3. Realizați un program care sa preia de la tastatură un şir de caractere şi apoi în funcție de selecția prin intermediul unui argumnet exterior **`(args[0])`** să realizeze fie folosirea stivei pentru afişarea inversă a informației, fie coada, pentru afişarea directă.

4. Realizați un program în care înlocuind metodele **`insert`** şi **remove**, să folosiți trei metode noi: **`insertLeft(),`** respectiv **`removeLeft(), removeRight().`**
   Ce structură implementează acum programul ?

5. Modificați programul astfel încât să realizați ordonarea crescătoare a elementelor şirului utilizând una dintre metodele studiate în cadrul laboratorului anterior. Stocarea se va realiza în coada, urmând ca apoi extragerea şi afişarea elementelor să se realizeze utilizând structura program anterioară (această structură poartă numele de coadă ordonată). Se va utiliza o metodă de ordonare care va fi introdusă în cadrul funcției insert.