

## Stack och Kö

- Implementering
- Tillämpningar



För utveckling av verksamhet, produkter och livskvalitet.



## ADS : STACK

- Det finns ett par vanligt förekommande ADT:er för samlingar av element som egentligen **är specialfall av listor**. En av dem är *Stack*
- **Definition:** En stack är en följd av element där borttagning av ett element alltid avser det element som senast satts in.
- Kallas även LIFO-lista, Last In First Out

För utveckling av verksamhet, produkter och livskvalitet.



## ADT: STACK

- En stack beskrivs oftast **lodrätt orienterad**.
- Operationer sker på **toppen av stacken**. Här läggs nytt element in ( **top** ) och vid borttagning avses alltid översta elementet ( **top** ).




För utveckling av verksamhet, produkter och livskvalitet.



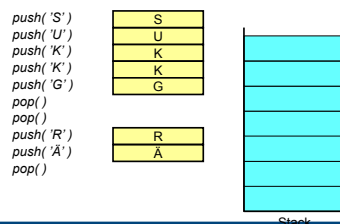
## Oftast förekommande metoder i Stack

```
public interface Stack {
    /** Lägg x "överst" på stacken */
    void push(Object x);
    /** Tag bort översta elementet från stacken */
    void pop();
    /** Tag reda på översta elementet på stacken */
    Object top();
    /** Tag reda på och tag bort översta elementet på stacken */
    Object topAndPop();
    /** Undersök om stacken är tom */
    boolean isEmpty();
    /** Töm stacken */
    void makeEmpty();
}
```

För utveckling av verksamhet, produkter och livskvalitet.



## Principen med en stack



För utveckling av verksamhet, produkter och livskvalitet.



## Användningsområden

- Datorteknik
  - Metod anrop, Kompilatorer
- Beräkningsmaskin
  - Omvänd polsk (postfix) notation (1950-talet)
  - Kräver "ett annat sorts tänkande..."
  - Kraftfull!

För utveckling av verksamhet, produkter och livskvalitet.



## Implementeringar

- Statisk implementation
  - Arraybaserad
- Dynamiska implementation
  - Länkade listor

För utveckling av verksamhet, produkter och livskvalitet.



## Med array...

- ToS – Top of Stack
  - Indexerar senast inlagda data
- Statisk maxstorlek
  - Dubblering möjlig men kostsam
- Fördel : Snabb

För utveckling av verksamhet, produkter och livskvalitet.



## Exempel

```
public class Stack <AnyType>
{
    private ...

    public Stack()
    {
    }

    public void push(AnyType data)
    {
    }

    public void pop()
    {
    }
}
```

För utveckling av verksamhet, produkter och livskvalitet.



## Tillstånd / Konstruktorn

```
private AnyType [] array;
private int ToS;

public Stack()
{
    array = (AnyType) new Object[50];
    ToS = -1;
}
```

För utveckling av verksamhet, produkter och livskvalitet.



## Metoden -- 'void push(AnyType data)'

```
public void push(AnyType data)
{
    if((ToS + 1) < array.length)
    {
        ToS++;
        array[ToS] = data;
    }
}
```

Tidskomplexiteten ??

För utveckling av verksamhet, produkter och livskvalitet.



## Metoden -- AnyType pop()'

```
public AnyType pop()
{
    AnyType value = null;
    if(ToS >= 0)
    {
        value = array[ToS];
        ToS--;
    }
    return value;
}
```

Tidskomplexiteten ??

För utveckling av verksamhet, produkter och livskvalitet.



## Med en länkad lista...

- Arbetar bara mot länken närmast huvudet
  - Läger in först i listan
  - Tar ut först ur listan
- Nackdel : **Långsam**
  - Kräver repeterade minnesallokering

För utveckling av verksamhet, produkter och livskvalitet.



## Exempel

```
public class Stack
{
private ...
public Stack()
{
}
public void push(Object x)
{
}
public Object pop()
{
}
}
```

För utveckling av verksamhet, produkter och livskvalitet.



## Tillstånd / Konstruktorn

```
private ListNode topofstack;
public Stack()
{
topofstack = null;
}
```

För utveckling av verksamhet, produkter och livskvalitet.



## Metoden -- 'void push(Object data)'

```
public void push(Object x)
{
topofstack=new ListNode(x,topofstack);
}
```

För utveckling av verksamhet, produkter och livskvalitet.



## Metoden -- 'Object pop()'

```
public Object pop()
{
    ListNode temp=topofstack;
    topofstack=topofstack.next;
    return temp.element;
}
```

För utveckling av verksamhet, produkter och livskvalitet.



## Implementationsjämförelse

- Statisk implementation, med array
  - (Kan ha) mycket utnyttjat minne
  - Snabb
- Dynamisk implementation, med länkad lista
  - Minimalt utnyttjat minne
  - Långsam

För utveckling av verksamhet, produkter och livskvalitet.



## Tillämpning-kompilator och miniräknare –

- Beräkning
  - aritmetiska uttryck i infix notation :  $1 + 2 * 3 - 4$
  - aritmetiska uttryck i postfix notation:  $1 2 3 * + 4 -$
- Kompilatorer, kontrollerar om koden är rätt parentiserad:  
rätt: { ( ) } fel: { ( ) }

För utveckling av verksamhet, produkter och livskvalitet.



## ADT: Kö - Kallas även FIFO-lista, First In First Out

- En annan modell, som också är specialfall av lista:
- **Definition:** En kö är en följd av element där
  - insättning alltid sker i **slutet av följen**
  - borttagning alltid avser följdens **första element**



För utveckling av verksamhet, produkter och livskvalitet.



## Användningsområden

- Printer köer,
- Simuleringar, av allt ...flyg, passersystem...
- Kommunikation mellan olika processer...
- Data buffring
- Datakommunikation / Internet

För utveckling av verksamhet, produkter och livskvalitet.



## Specifikation av ADT -Queue

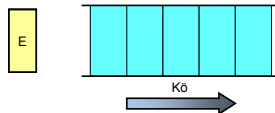
```
public interface Queue {  
    /** Sätt in x sist i kön */  
    void enqueue(Object x);  
    /** Tag reda på första elementet i kön */  
    Object getFront();  
    /** Tag reda på och tag bort första elementet i kön */  
    Object dequeue();  
    /** Undersök om kön är tom */  
    boolean isEmpty();  
    /** Töm kön */  
    void makeEmpty();  
}
```

För utveckling av verksamhet, produkter och livskvalitet.



## Principen med en kö

```
enqueue('A')  
enqueue('B')  
enqueue('C')  
enqueue('D')  
dequeue()  
dequeue()  
enqueue('E')  
dequeue()
```



För utveckling av verksamhet, produkter och livskvalitet.



## Implementeringar

- Statisk implementation
  - Arraybaserad
- Dynamiska implementation
  - Länkade listor

För utveckling av verksamhet, produkter och livskvalitet.



## Med en array...

- "Wraparound"
  - Slipper flytta data
  - Kräver start och slut index
  - Snabbt
- Statisk maxstorlek
  - Dubbling möjlig men kostsam

För utveckling av verksamhet, produkter och livskvalitet.



## Exempel

```
public class Queue
{
    private ...

    public Queue ()
    {
    }

    public void enqueue(Object data)
    {
    }

    public Object dequeue ()
    {
    }
}
```

För utveckling av verksamhet, produkter och livskvalitet.



## Tillstånd / Konstruktorn

```
private Object [] array;
private int back, front;
private int size;

public Queue ()
{
    array = new Object[50];
    back = -1;
    front = 0;
    size = 0;
}
```

För utveckling av verksamhet, produkter och livskvalitet.



## Metoden -- 'void enqueue(Object data)'

```
public void enqueue(Object data)
{
    if(size < array.length)
    {
        back = (back + 1)
        array[back] = data;
        size++;
    }
}
```

Tidskomplexiteten ??

För utveckling av verksamhet, produkter och livskvalitet.



## Metoden -- 'void enqueue(Object data)'

med Wraparound

```
public void enqueue(Object data)
{
    if(size == array.length)
        doubleQueue();
    back = increment(back);
    array[back] = data;
    size++;
}
```

Tidskomplexiteten ??

För utveckling av verksamhet, produkter och livskvalitet.



## Metoden -- 'int increment(int x)'

med Wraparound

```
private int increment(int x)
{
    if(++x == array.length)
        x=0;
    return x;
}
```

För utveckling av verksamhet, produkter och livskvalitet.



## Exempel

```
public class Queue
{
    private ...

    public Queue()
    {
    }

    public void enqueue(Object data)
    {
    }

    public Object dequeue()
    {
    }
}
```

För utveckling av verksamhet, produkter och livskvalitet.



## Metoden -- 'Object dequeue()'

```
public Object dequeue()
{
    Object value = null;
    if(size > 0)
    {
        value = array[front];
        front = front + 1;
        size--;
    }
    return value;
}
```

För utveckling av verksamhet, produkter och livskvalitet.





## Metoden -- 'Object dequeue()'

Med wraparound

```
public Object dequeue()
{
    Object value = null;

    if(size > 0)
    {
        value = array[front];
        front = increment(front);
        size--;
    }

    return value;
}
```

För utveckling av verksamhet, produkter och livskvalitet.



## Med en länkad lista...

- Arbetar bara mot ändarna på listan
- Läger in först i listan
- Tar ut sist ur listan
- Långsam
  - Minnesallokering
  - Behöver ingen "wrapping"
- Enklare implementation

För utveckling av verksamhet, produkter och livskvalitet.



## Tillstånd/ Konstruktör

```
private ListNode front;
private ListNode back;

public Queue()
{
    front=back=null;
}
```

För utveckling av verksamhet, produkter och livskvalitet.



## Metoden -- 'void enqueue(Object data)'

```
public void enqueue(Object data)
{
    if (isEmpty())
        back=front=new ListNode(data);
    else
        back=back.next=new ListNode(data);
}
```

För utveckling av verksamhet, produkter och livskvalitet.



## Metoden -- 'Object dequeue()'

```
public Object dequeue()
{
    if (isEmpty())
        throw new UnderflowException("Tom kö");
    Object value=front.element;
    front=front.next;
    return value;
}
```

För utveckling av verksamhet, produkter och livskvalitet.



## Implementationsjämförelse

- Statisk implementation
  - (Kan ha) mycket utnyttjat minne
  - Snabb
- Dynamisk implementation
  - Minimalt utnyttjat minne
  - Långsam
- Konstant tid

För utveckling av verksamhet, produkter och livskvalitet.

