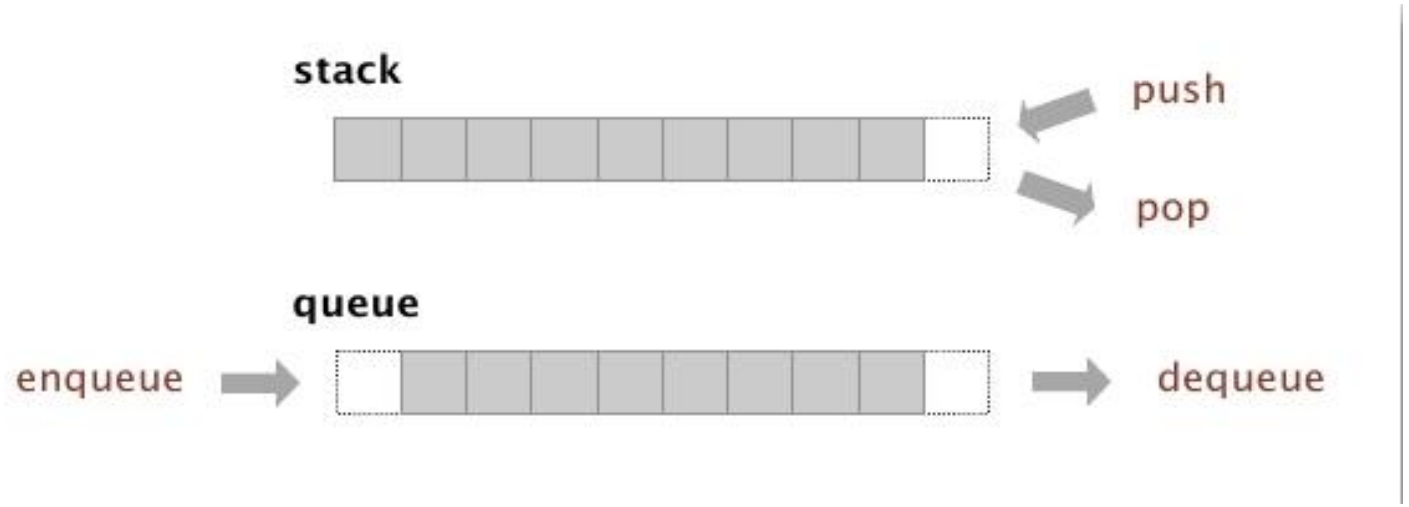
Стеки и очереди

- Фундаментальные структуры данных
  - Значения: коллекции объектов
  - Операции: вставка, удаление, итерации, проверка на пустоту

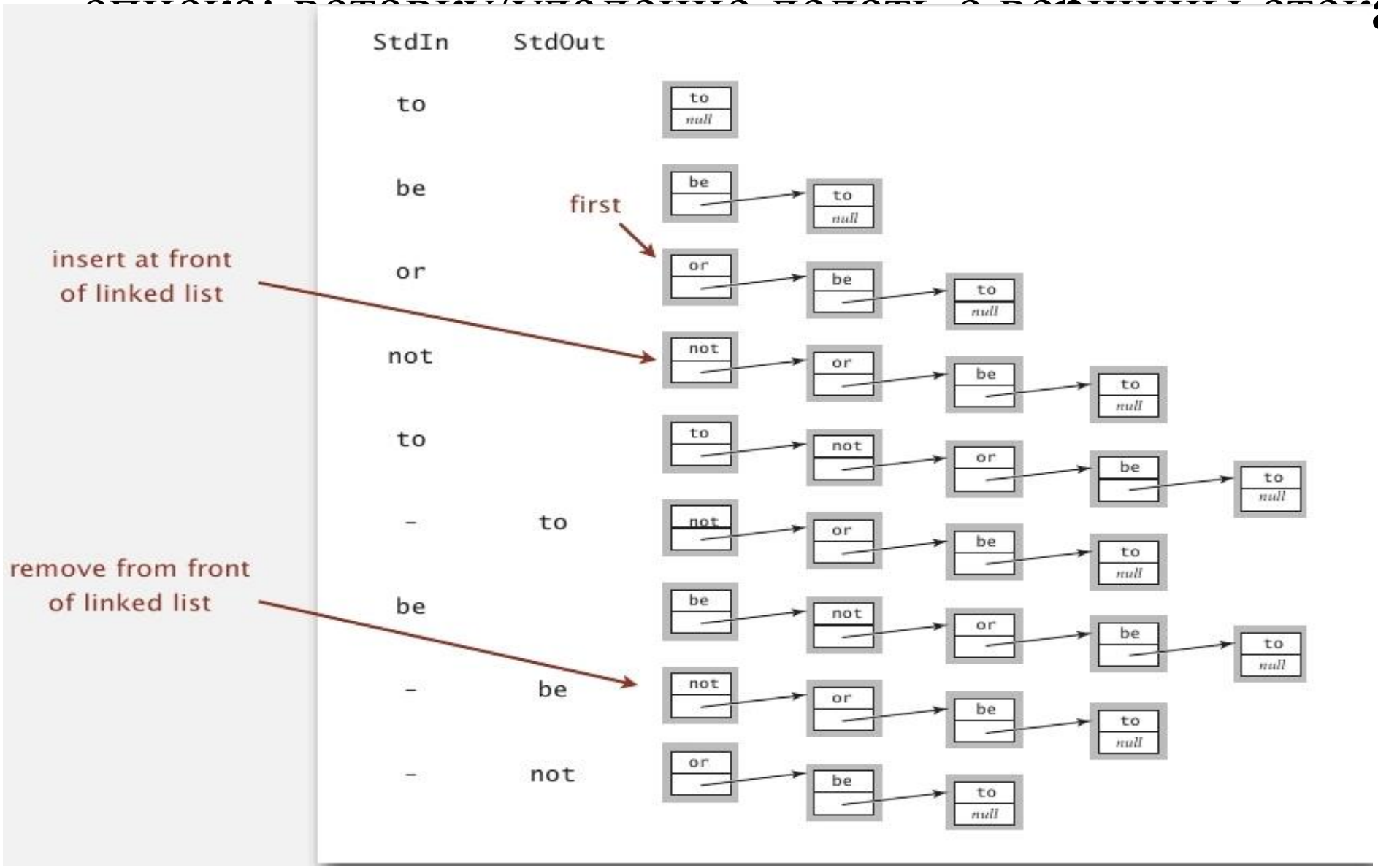


- Стек. LIFO
- Очередь. FIFO

Стеки

Стек. Связный список

- Хранить указатель на первый элемент связного списка; вставка/удаление делать с вершины стека



Изъять элемент из стека. Реализация с помощью связного списка

inner class

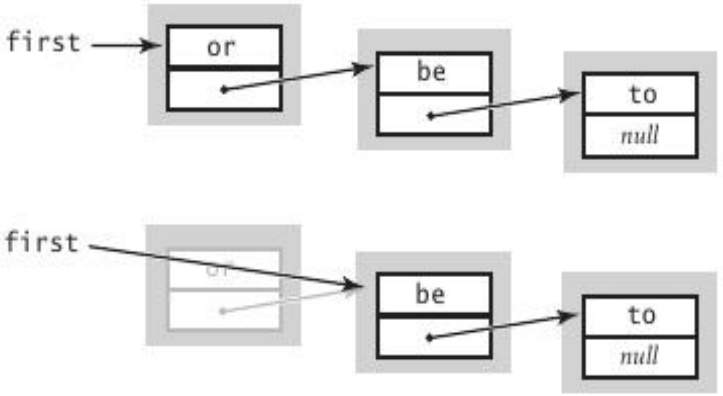
```
private class Node
{
   String item;
   Node next;
}
```

save item to return

```
String item = first.item;
```

delete first node

```
first = first.next;
```



return saved item

```
return item;
```

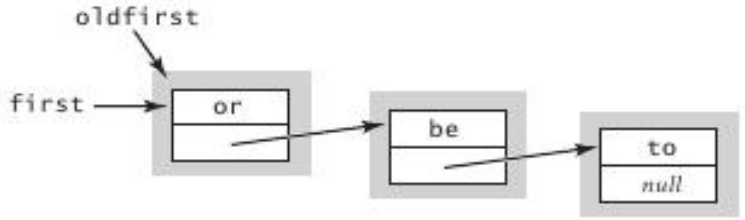
Добавить элемент в стек. Реализация с помощью связного списка

inner class

```
private class Node
{
   String item;
   Node next;
}
```

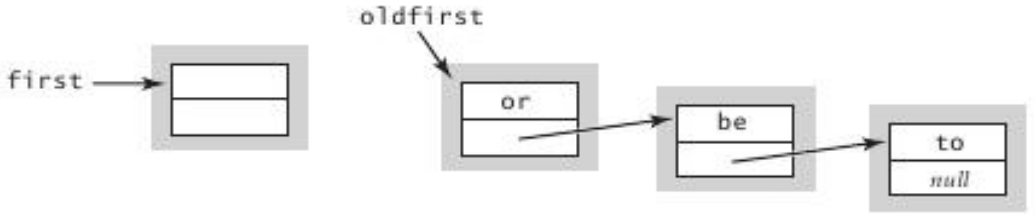
save a link to the list

```
Node oldfirst = first;
```



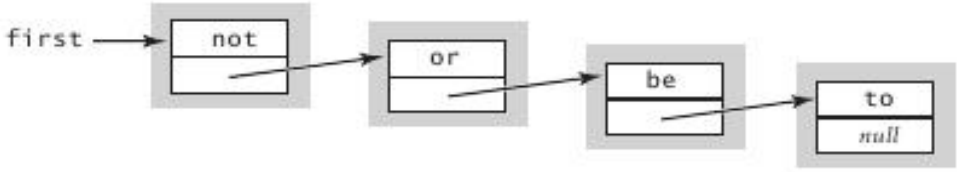
create a new node for the beginning

```
first = new Node();
```



set the instance variables in the new node

```
first.item = "not";
first.next = oldfirst;
```



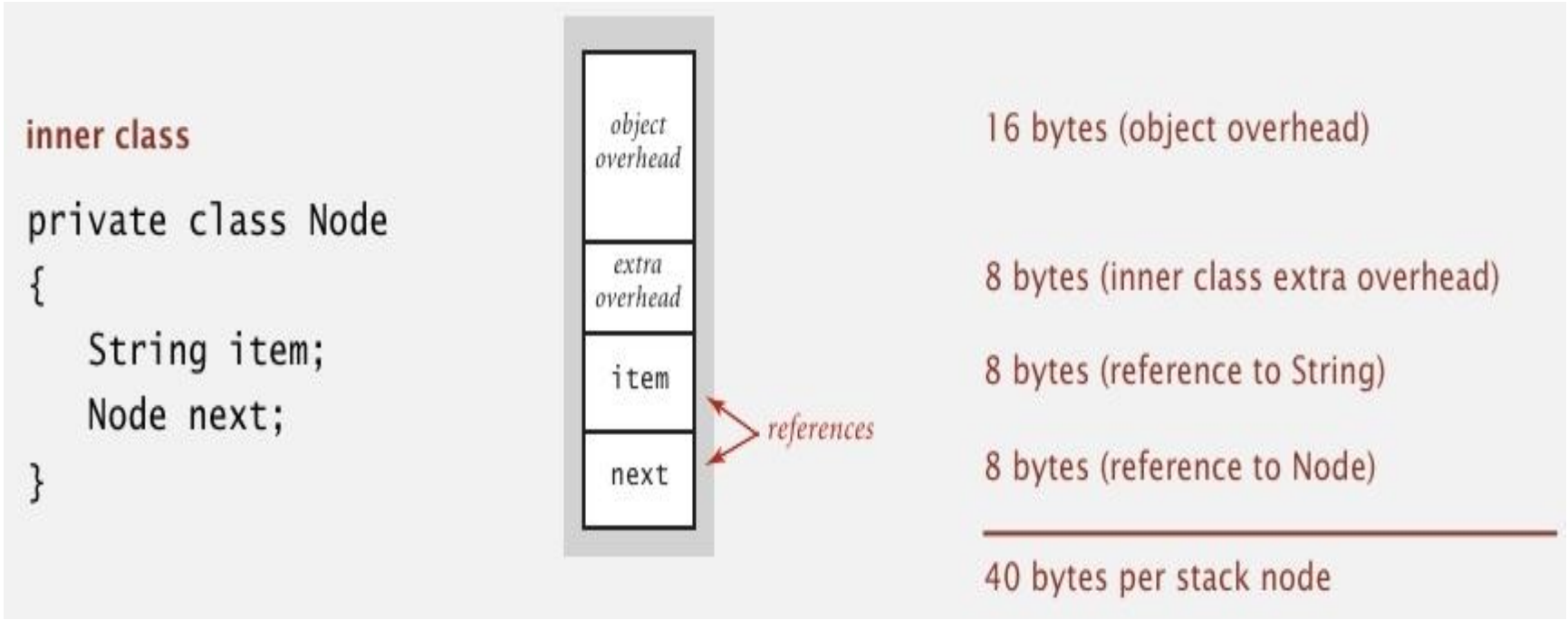
Стек. Реализация на Java

```
public class LinkedStackOfStrings
{
   private Node first = null;

   private class Node
   {
      String item;
      Node next;
   }

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

   public void push(String item)
   {
      Node oldfirst = first;
      first = new Node();
      first.item = item;
      first.next = oldfirst;
   }

   public String pop()
   {
      String item = first.item;
      first = first.next;
      return item;
   }
}
```

private inner class
(access modifiers don't matter)

Стек. Производительность реализации с помощью связного списка

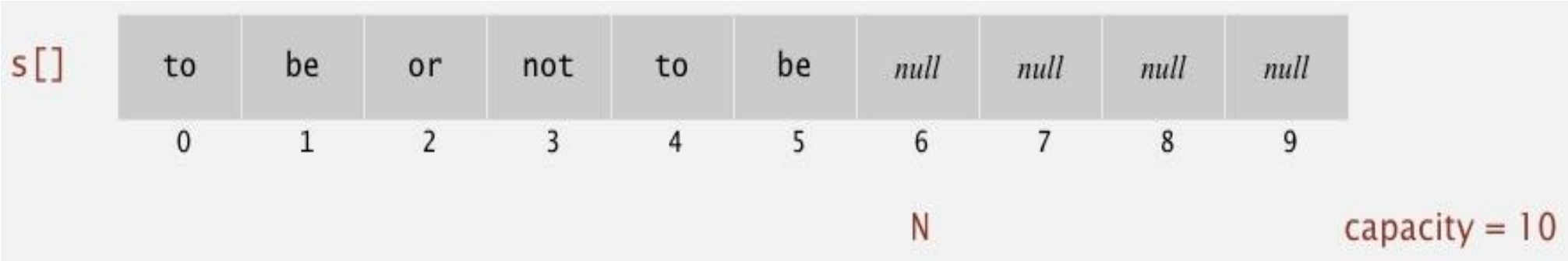
- Каждая операция производится за время равное константе
- Стек из N элементов испольщует ~ 40N байт



- Замечание: здесь учитывается сколько занимает сам стек. Память, которую занимают строки

Стек. Реализация с помощью массива

- Массив s[] для хранения N элементов стека
- push(): добавит новый элемент в s[N]
- pop(): изъять элемент из s[N-1]



- Дефект. Переполнение массива

Стек. Реализация с помощью массива

```
public class FixedCapacityStackOfStrings
{
   private String[] s;
   private int N = 0;

   public FixedCapacityStackOfStrings(int capacity)
   {  s = new String[capacity];  }

   public boolean isEmpty()
   {  return N == 0;  }

   public void push(String item)
   {  s[N++] = item;  }

   public String pop()
   {  return s[--N];  }
}
```

a cheat
(stay tuned)

use to index into array;
then increment N

decrement N;
then use to index into array

Стек. Некоторые соображения

- Переполнение и изъятие из пустого стека
  - Изъятие из пустого стека: исключительная ситуация
  - Переполнение: изменить размер массива
- null: свободное место в массиве
- Бесхозные ссылки: хранение ссылок на объекты, ко

```
public String pop()
{  return s[--N];  }
```

loitering

```
public String pop()
{
   String item = s[--N];
   s[N] = null;
   return item;
}
```

this version avoids "loitering":
garbage collector can reclaim memory
only if no outstanding references

Изменение размера массива

Стек: изменение размера массива

- Проблема. От клиента требуется указывать размер стека
- Как увеличивать и уменьшать размер массива?

- Первый подход
  - push(): увеличивать размер массива s[] на 1
  - pop(): уменьшать размер массива s[] на 1

- Стоимость
  - Требуется копировать все элементы в новый массив
  - Стоимость вставки первых N элементов 1 + 2 + 3 + ... + N

Стек: изменение размера массива

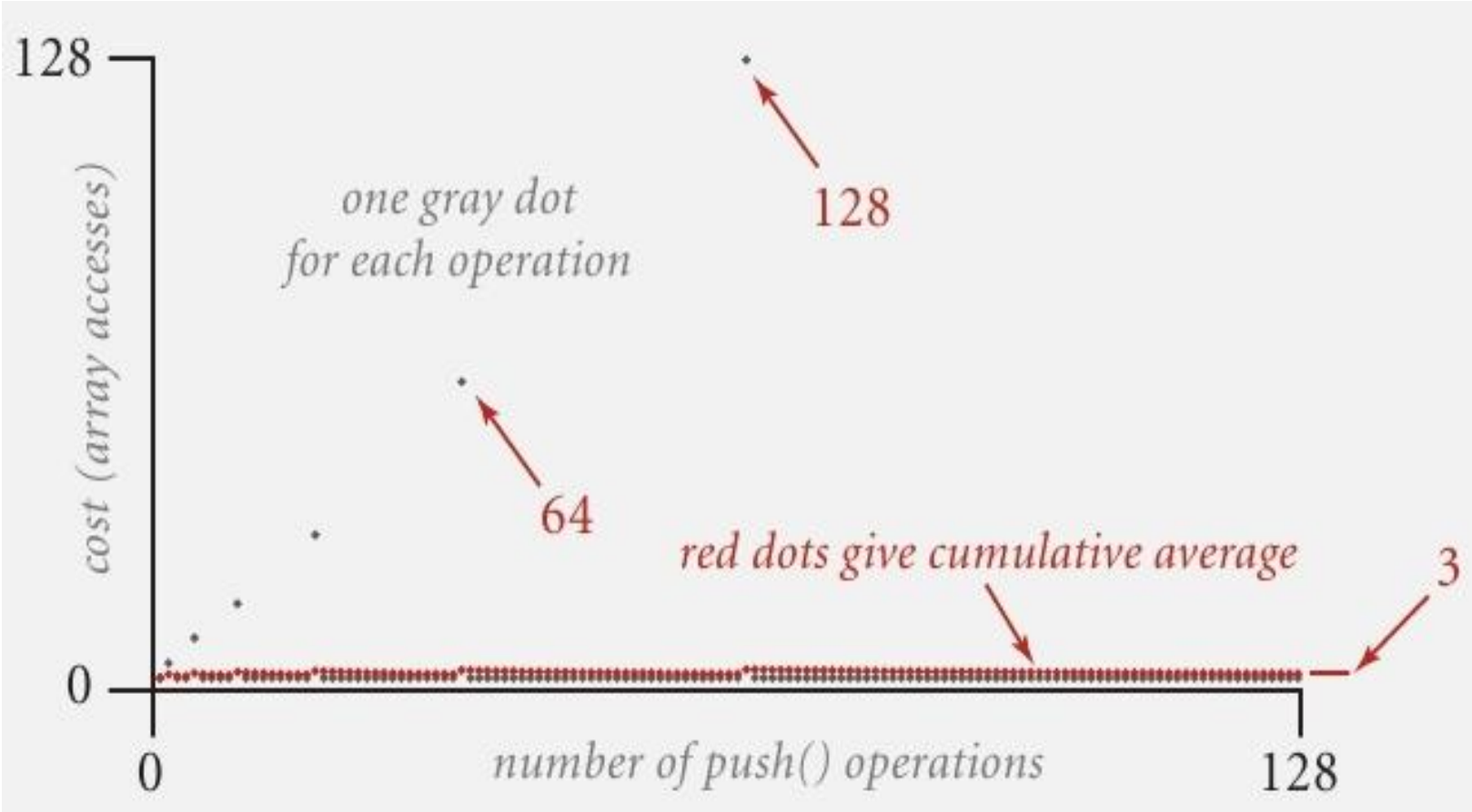
- Если массив полон, то создать новый массив в два раза больше и копировать элементы

```
public ResizingArrayStackOfStrings()
{  s = new String[1];  }

public void push(String item)
{
   if (N == s.length) resize(2 * s.length);
   s[N++] = item;
}

private void resize(int capacity)
{
   String[] copy = new String[capacity];
   for (int i = 0; i < N; i++)
      copy[i] = s[i];
   s = copy;
}
```

- Стоимость. Сложность вставки первых N элементов пропорциональна N

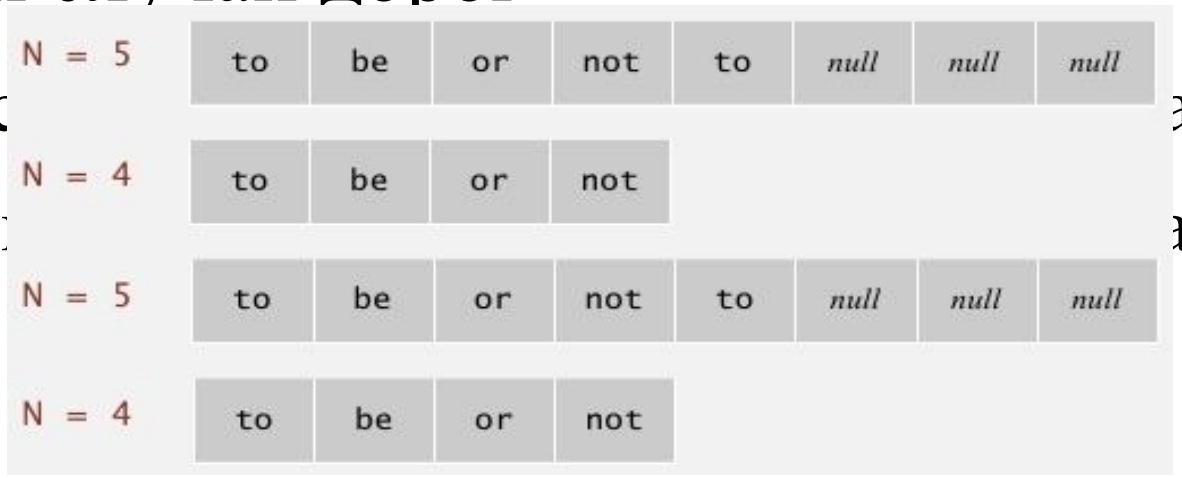
Стек: амортизированная стоимость добавления в стек

- Стоимость добавления первых N элементов: N + (2 + 4 + 8 + … + N) ~ 3N



Стек: изменение размера массива

- Как изменять размер массива?
- Первый подход
  - push(): увеличивать размер массива s[] в два раза, когда массив полон
  - pop(): уменьшать размер массива s[] в два раза, когда массив на половину пуст
- Худший случай дорог
  - Предс ассив полон
  - Слож альна N



Стек: изменение размера массива

- Эффективный подход
  - push(): увеличивать размер массива s[] в два раза, когда массив полон
  - pop(): уменьшать размер массива s[] в два раза, когда массив заполнен на четверть

```
public String pop()
{
   String item = s[--N];
   s[N] = null;
   if (N > 0 && N == s.length/4) resize(s.length/2);
   return item;
}
```

- Массив заполнен от 25% до 100%

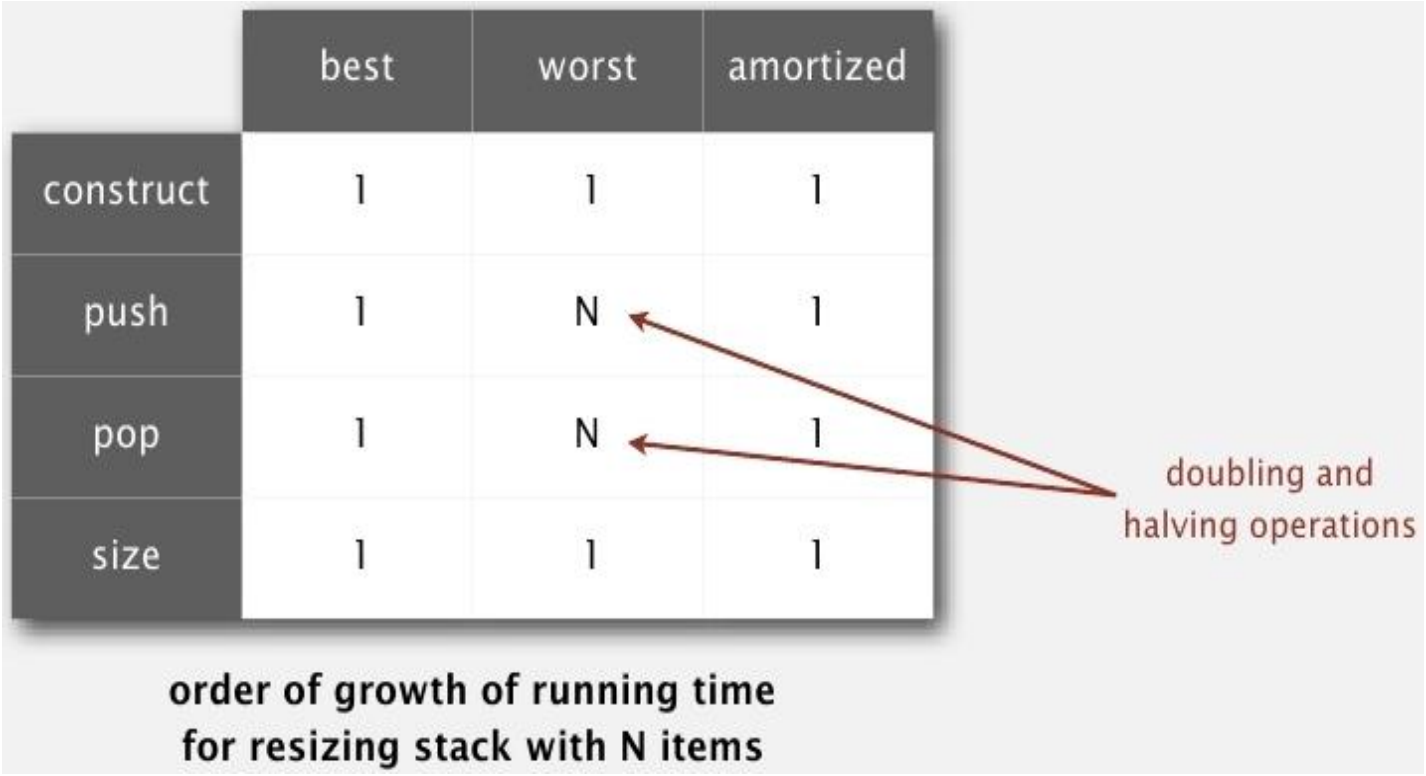
Стек: изменение размера массива

push()	pop()	N	a.length	a[] 0	1	2	3	4	5	6	7
		0	1	*null*							
to		1	1	to							
be		2	2	to	be						
or		3	4	to	be	or	*null*				
not		4	4	to	be	or	not				
to		5	8	to	be	or	not	to	*null*	*null*	*null*
-	to	4	8	to	be	or	not	*null*	*null*	*null*	*null*
be		5	8	to	be	or	not	be	*null*	*null*	*null*
-	be	4	8	to	be	or	not	*null*	*null*	*null*	*null*
-	not	3	8	to	be	or	*null*	*null*	*null*	*null*	*null*
that		4	8	to	be	or	that	*null*	*null*	*null*	*null*
-	that	3	8	to	be	or	*null*	*null*	*null*	*null*	*null*
-	or	2	4	to	be	*null*	*null*				
-	be	1	2	to	*null*						
is		2	2	to	is						

Trace of array resizing during a sequence of push() and pop() operations

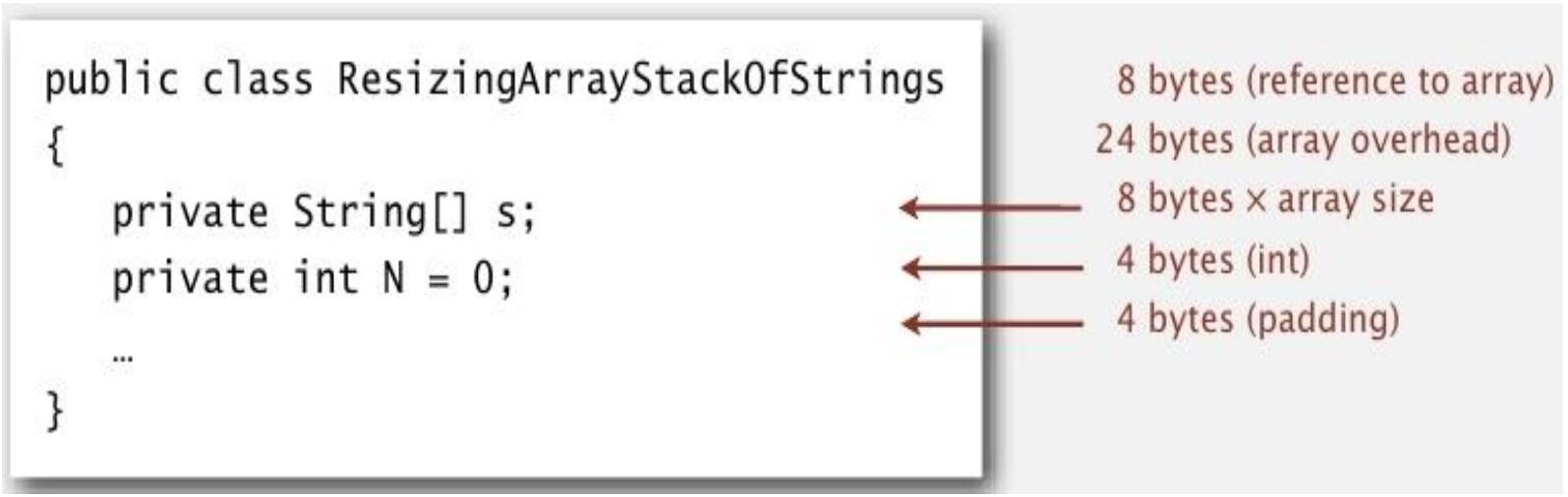
Стек: амортизированный анализ

- Предположение. Начиная с пустого стека, последовательность из M push/pop операций занимает время пропорциональное M



Стек: использование памяти

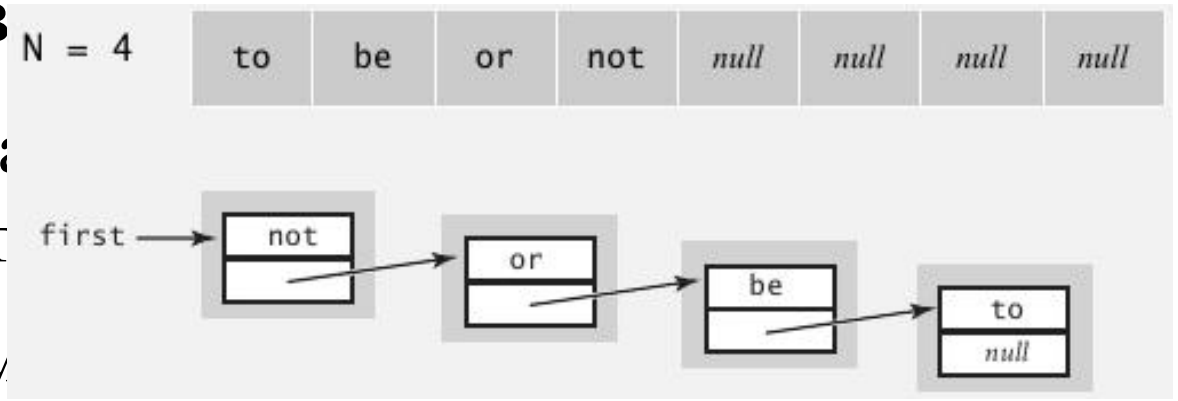
- Предположение. Используется от ~ 8N до ~ 32N байт для стека из N элементов
  - ~ 8N когда стек полон
  - ~ 32N когда стек заполнен на четверть



- Учитывается память, занимаемая самим стеком, но не данными

Реализация стек: массив и связный список

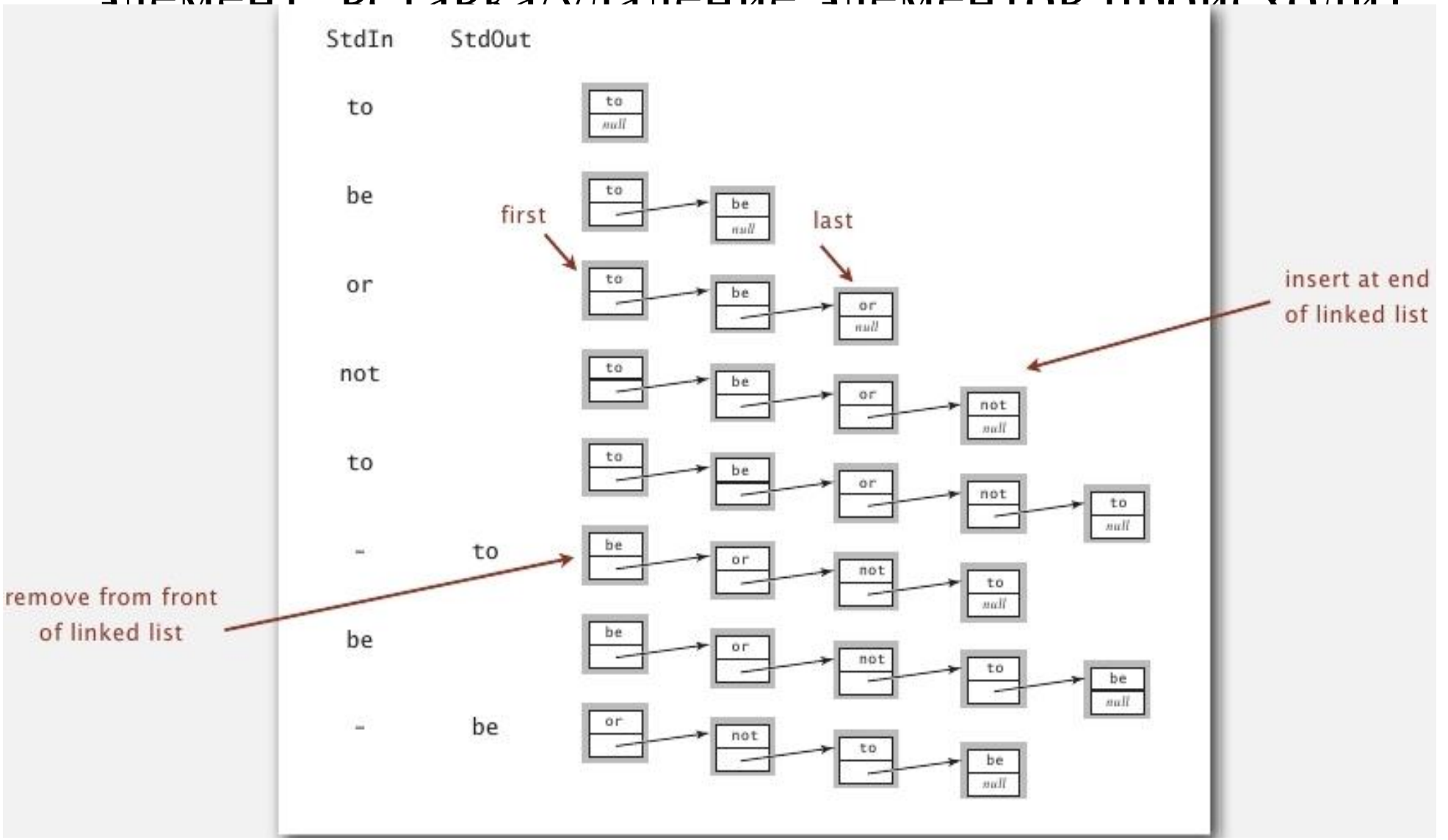
- Компромисс. Сделать две реализации стека и дать возможность клиенту выбрать.
- Связный список
  - Каждая операция занимает константное время в худшем случае
  - Использует дополнительное время и память для организации ссылок
- Массив
  - Кажда
    аморт
  - Заним



Очередь

Очередь: связный список

- Хранить указатели на первый и последний элемент; вставка/удаление элементов происходит



Очередь: удаление элемента

inner class

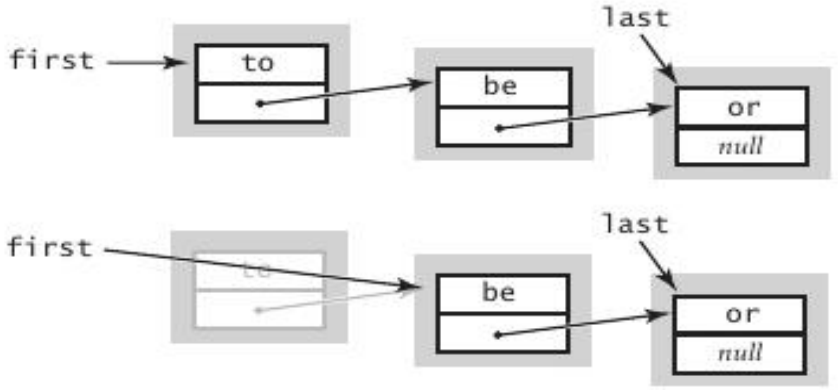
```
private class Node
{
   String item;
   Node next;
}
```

save item to return

```
String item = first.item;
```

delete first node

```
first = first.next;
```



return saved item

```
return item;
```

- Аналогична операции pop() для стека

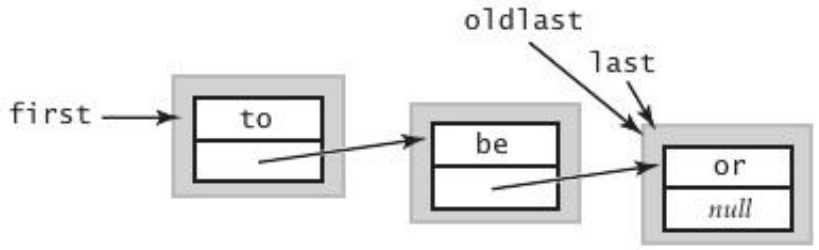
Очередь: вставка элемента

inner class

```
private class Node
{
   String item;
   Node next;
}
```

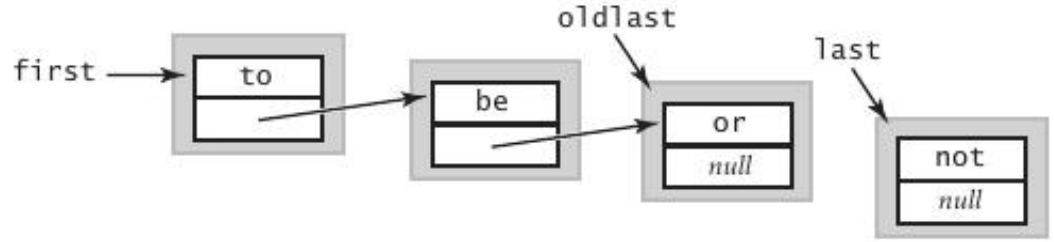
save a link to the last node

```
Node oldlast = last;
```



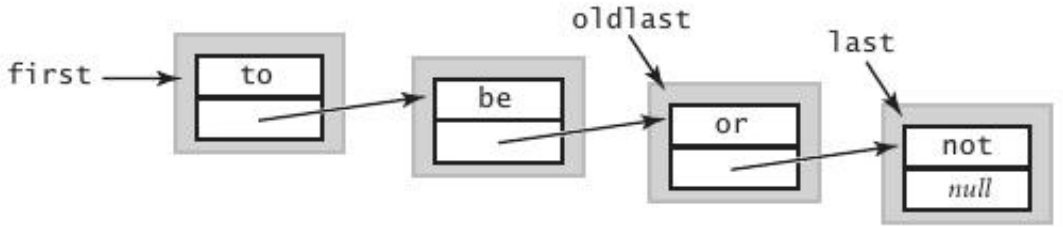
create a new node for the end

```
last = new Node();
last.item = "not";
```



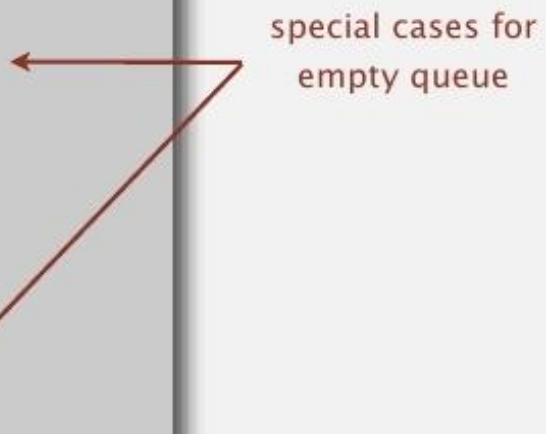
link the new node to the end of the list

```
oldlast.next = last;
```



Очередь: реализация на Java

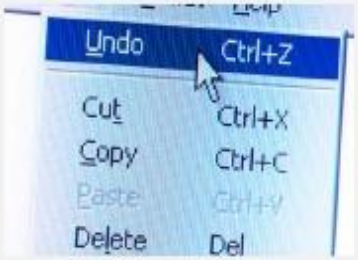
```
public class LinkedQueueOfStrings
{
   private Node first, last;

   private class Node
   {  /* same as in StackOfStrings */  }

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

   public void enqueue(String item)
   {
      Node oldlast = last;
      last = new Node();
      last.item = item;
      last.next = null;
      if (isEmpty()) first = last;
      else           oldlast.next = last;
   }

   public String dequeue()
   {
      String item = first.item;
      first       = first.next;
      if (isEmpty()) last = null;
      return item;
   }
}
```

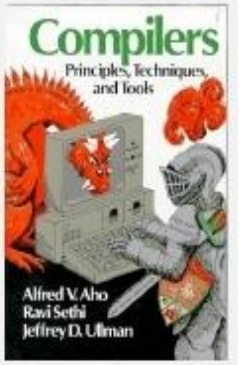


Применение очередей, контейнеров и стеков

Применение стека

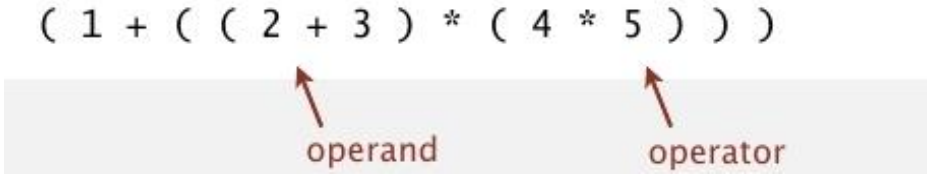
- Parsing in a compiler.
- Java virtual machine.
- Undo in a word processor.
- Back button in a Web browser.
- PostScript language for printers.
- Implementing function calls in a compiler.
- ...



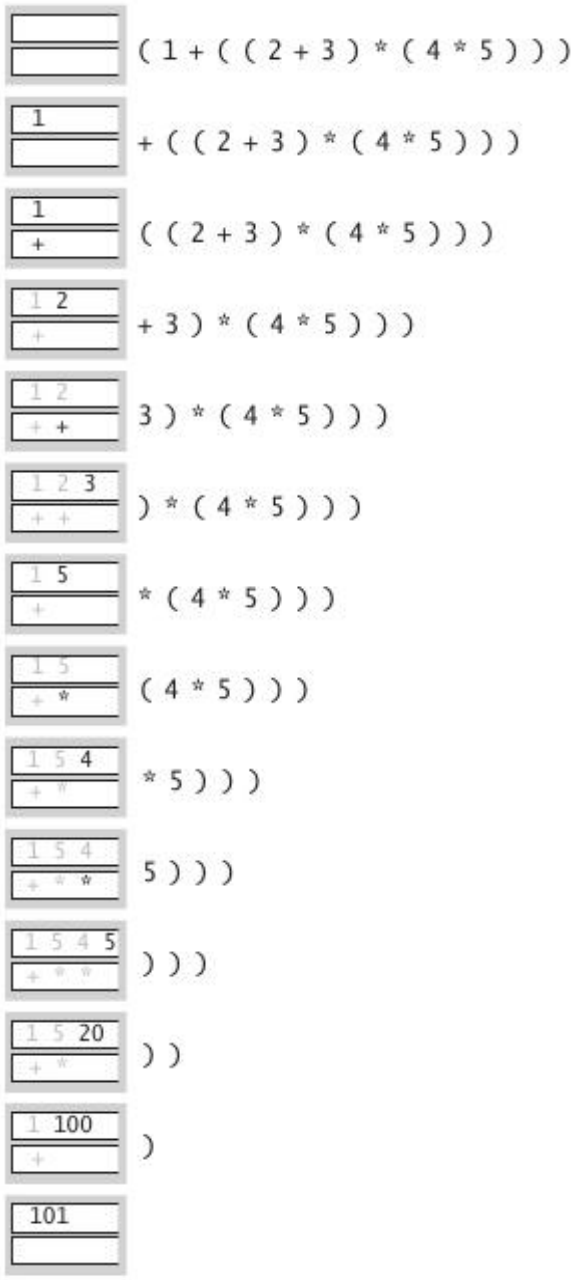


Вычисление арифметических выражений

- Цель. Вычислить выражение в инфиксной форме



- Двустековый алгоритм Дейкстры
  - Значение: занести в стек значений
  - Оператор: занести в стек оператор
  - Левая скобка: игнорируем
  - Правая скобка: выталкиваем оператор и два значения, выполняем операцию и заносим в стек значений
- Где используется?



Вычисление арифметических выражений

```
public class Evaluate
{
   public static void main(String[] args)
   {
      Stack<String> ops  = new Stack<String>();
      Stack<Double> vals = new Stack<Double>();
      while (!StdIn.isEmpty()) {
         String s = StdIn.readString();
         if      (s.equals("("))               ;
         else if (s.equals("+"))    ops.push(s);
         else if (s.equals("*"))    ops.push(s);
         else if (s.equals(")"))
         {
            String op = ops.pop();
            if      (op.equals("+")) vals.push(vals.pop() + vals.pop());
            else if (op.equals("*")) vals.push(vals.pop() * vals.pop());
         }
         else vals.push(Double.parseDouble(s));
      }
      StdOut.println(vals.pop());
   }
}
```

```
% java Evaluate
( 1 + ( ( 2 + 3 ) * ( 4 * 5 ) ) )
101.0
```