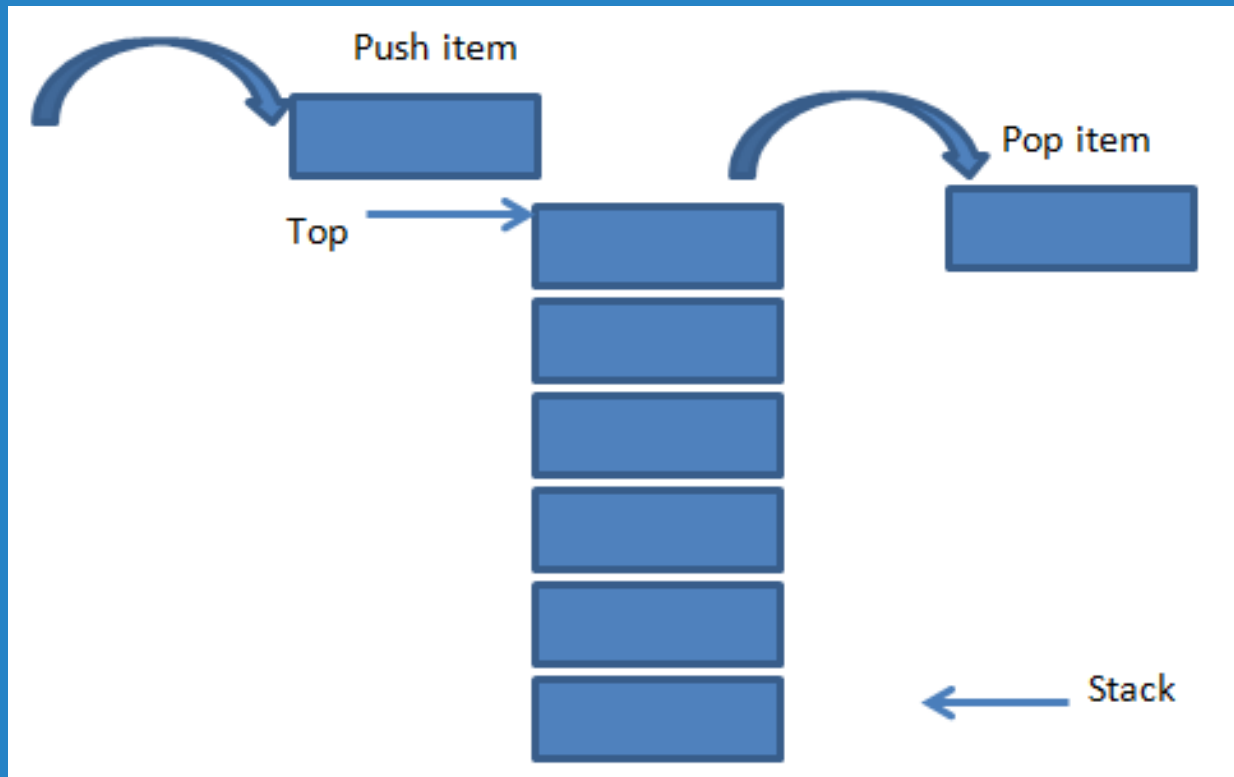
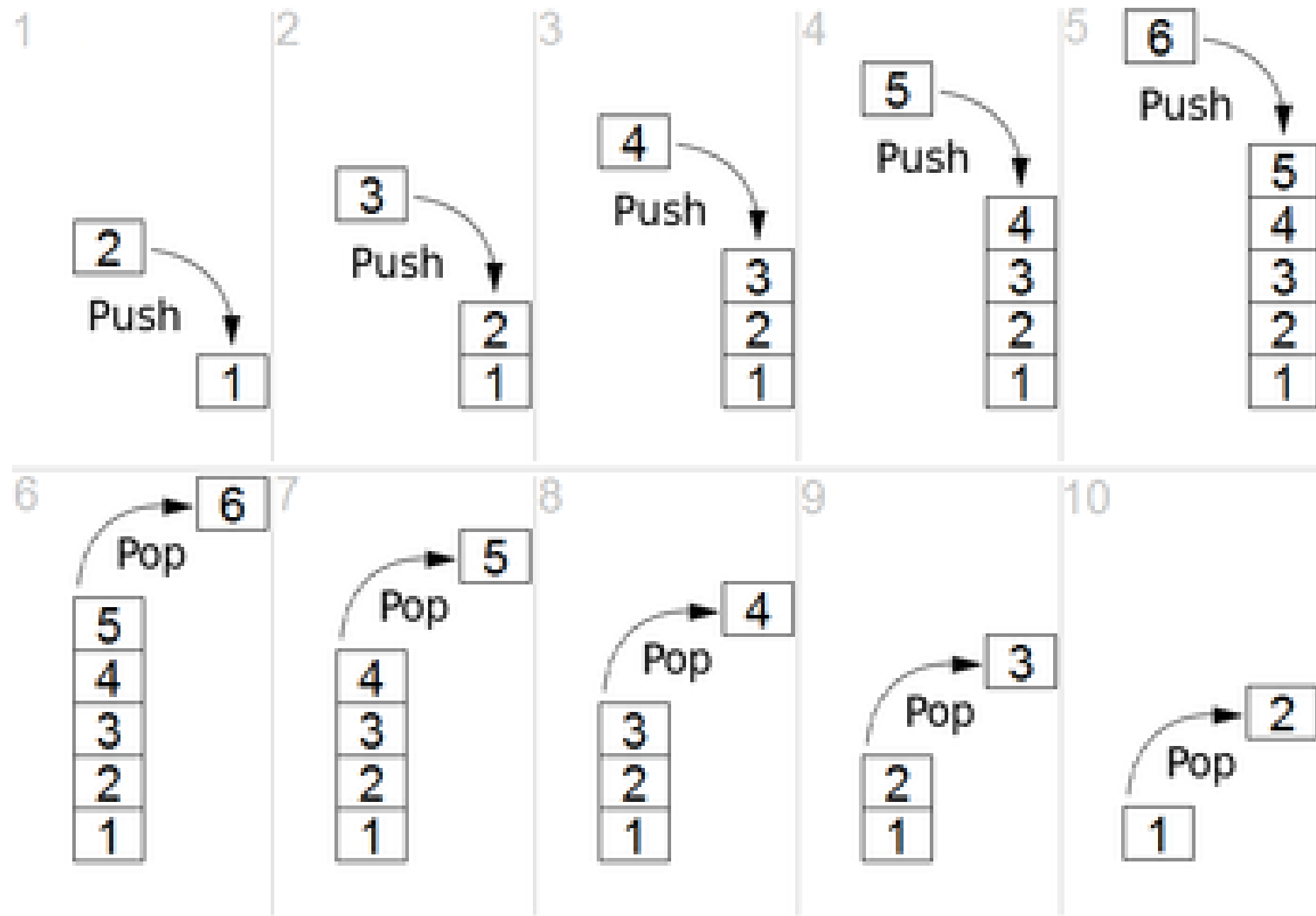


# Stack

В примери





# LIFO / FILO

Стек- линейна колекция от данни с две основни операции- операция за добавяне (`push`) и операция премахване (`pop`) на данна във върха на стека (**LIFO** структура).

Пример: стек от карти за игра, стек от кутии, стек от писма и пр.

## Операции с елементите на стека:

- Конструктор по подразбиране и копиране;
- Метод за проверка дали стека е празен;
- Методи **push** и **pop**
- Метод за разглеждане (**peek**) на данната във върха на стека
- Метод, връщащ стринг с данните в стека

## Реализация:

- Данните са елементи на масив
- Данните са във възли на свързан списък.

## Методи в Stack class

- Object push(*Object element*)
- Object pop()
- Object peek()
- boolean empty()
- int search(*Object element*)

# Пример за основните операции със STACK

# Добавяне на String елементи в Stack.

Initial Stack: [A, B, C, D, E]  
Final Stack: [A, B, C, D, E, F, G]

```
1 // Java code to illustrate push() method
2
3 import java.util.*;
4
5 public class StackDemo {
6     public static void main(String args[])
7     {
8         // Creating an empty Stack
9         Stack<String> STACK = new Stack<String>();
10
11         // Use push() to add elements into the Stack
12         STACK.push("A");
13         STACK.push("B");
14         STACK.push("C");
15         STACK.push("D");
16         STACK.push("E");
17
18         // Displaying the Stack |
19         System.out.println("Initial Stack: " + STACK);
20
21         // Pushing elements into the stack
22         STACK.push("F");
23         STACK.push("G");
24
25         // Displaying the final Stack
26         System.out.println("Final Stack: " + STACK);
27     }
28 }
29
```

# Добавяне на Integer елементи в Stack.

## Output:

Initial Stack: [10, 15, 30, 20, 5]  
Final Stack: [10, 15, 30, 20, 5, 1254, 4521]

```
// Java code to illustrate push() method
import java.util.*;

public class StackDemo {
    public static void main(String args[])
    {
        // Creating an empty Stack
        Stack<Integer> STACK = new Stack<Integer>();

        // Use push() to add elements into the Stack
        STACK.push(10);
        STACK.push(15);
        STACK.push(30);
        STACK.push(20);
        STACK.push(5);

        // Displaying the Stack
        System.out.println("Initial Stack: " + STACK);

        // Pushing elements into the Stack
        STACK.push(1254);
        STACK.push(4521);

        // Displaying the final Stack
        System.out.println("Final Stack: " + STACK);
    }
}
```

# Премахване на String елемент от върха на стека /pop/

Initial Stack: [A, B, C, D, E]  
Popped element: E  
Popped element: D  
Stack after pop peration [A, B, C]

Annie Ushanova

```
1 // Java code to illustrate pop()
2 import java.util.*;
3
4 public class StackDemo {
5     public static void main(String args[])
6     {
7         // Creating an empty Stack
8         Stack<String> STACK = new Stack<String>();
9
10        // Use add() method to add elements
11        STACK.push("A");
12        STACK.push("B");
13        STACK.push("C");
14        STACK.push("D");
15        STACK.push("E");
16
17        // Displaying the Stack
18        System.out.println("Initial Stack: " + STACK);
19
20        // Removing elements using pop() method
21        System.out.println("Popped element: " +
22                           STACK.pop());
23        System.out.println("Popped element: " +
24                           STACK.pop());
25
26        // Displaying the Stack after pop operation
27        System.out.println("Stack after pop peration "
28                           + STACK);
29    }
30 }
```





# Премахване на Integer елемент от върха на стека /pop/

## Output:

```
Initial Stack: [10, 15, 30, 20, 5]
Popped element: 5
Popped element: 20
Stack after pop operation [10, 15, 30]
```

Annie Ushanova

```
// Java code to illustrate pop()
import java.util.*;

public class StackDemo {
    public static void main(String args[])
    {
        // Creating an empty Stack
        Stack<Integer> STACK = new Stack<Integer>();

        // Use add() method to add elements
        STACK.push(10);
        STACK.push(15);
        STACK.push(30);
        STACK.push(20);
        STACK.push(5);

        // Displaying the Stack
        System.out.println("Initial Stack: " + STACK);

        // Removing elements using pop() method
        System.out.println("Popped element: " +
                           STACK.pop());
        System.out.println("Popped element: " +
                           STACK.pop());

        // Displaying the Stack after pop operation
        System.out.println("Stack after pop operation "
                           + STACK);
    }
}
```



Методът `java.util.Stack.peek()` в Java се използва за извличане на елемента, от върха на стека. Полученият елемент не се изтрива или премахва от стека

Initial Stack: [AA, BB, CC, DD, EE]  
The element at the top of the stack is: EE  
Final Stack: [AA, BB, CC, DD, EE]

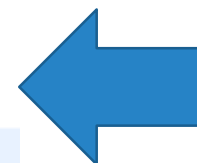
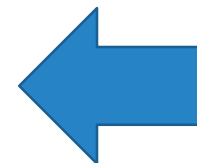
```
1 // Java code to illustrate peek() function
2
3 import java.util.*;
4
5 public class StackDemo {
6     public static void main(String args[])
7     {
8         // Creating an empty Stack
9         Stack<String> STACK = new Stack<String>();
10
11         // Use push() to add elements into the Stack
12         STACK.push("AA");
13         STACK.push("BB");
14         STACK.push("CC");
15         STACK.push("DD");
16         STACK.push("EE");
17
18         // Displaying the Stack
19         System.out.println("Initial Stack: " + STACK);
20
21         // Fetching the element at the head of the Stack
22         System.out.println("The element at the top of the"
23                             + " stack is: " + STACK.peek());
24
25         // Displaying the Stack after the Operation
26         System.out.println("Final Stack: " + STACK);
27     }
28 }
```

# Проверка дали стека е празен

```
The stack is: [A, B, C, D, E]
Is the stack empty? false
Is the stack empty? true
```

Annie Ushanova

```
1 // Java code to demonstrate empty() method
2 import java.util.*;
3
4 public class Stack_Demo {
5     public static void main(String[] args)
6     {
7
8         // Creating an empty stack
9         Stack<String> STACK = new Stack<String>();
10
11        // Stacking strings
12        STACK.push("A");
13        STACK.push("B");
14        STACK.push("C");
15        STACK.push("D");
16        STACK.push("E");
17
18        // Displaying the stack
19        System.out.println("The stack is: " + STACK);
20
21        // Checking for the emptiness of stack
22        System.out.println("Is the stack empty? " +
23                           STACK.empty());
24
25        // Popping out all the elements
26        STACK.pop();
27        STACK.pop();
28        STACK.pop();
29        STACK.pop();
30        STACK.pop();
31
32        // Checking for the emptiness of stack
33        System.out.println("Is the stack empty? " +
34                           STACK.empty());
35    }
36 }
37
```



# Задача

- Създайте метод за добавяне на числата от 0 до 4 в стек  
**stack\_push(Stack<Integer> stack);**
- Създайте метод за последователното премахване на елементите от върха на стека и отпечатването им на екрана  
**stack\_pop(Stack<Integer> stack);**
- Създайте метод за отпечатване на елемента от върха на стека  
**stack\_peek(Stack<Integer> stack);**
- Създайте метод за търсене на елемент в стек  
**stack\_search(Stack<Integer> stack, int element)**
- Имплементирайте стек за целочислени данни като използвате горните методи.

1

- **PUSH** –

Добавя  
елемент на  
върха на стека

- **POP** –

Премахване  
елемент от  
върха на стека

```
// Java code for stack implementation
```

```
import java.io.*;
import java.util.*;
```

```
class Test
{
```

```
    // Pushing element on the top of the stack
```

```
    static void stack_push(Stack<Integer> stack)
```

```
    {
```

```
        for(int i = 0; i < 5; i++)
```

```
        {
```

```
            stack.push(i);
```

```
        }
```

```
    }
```

```
    // Popping element from the top of the stack
```

```
    static void stack_pop(Stack<Integer> stack)
```

```
    {
```

```
        System.out.println("Pop :");
```

```
        for(int i = 0; i < 5; i++)
```

```
        {
```

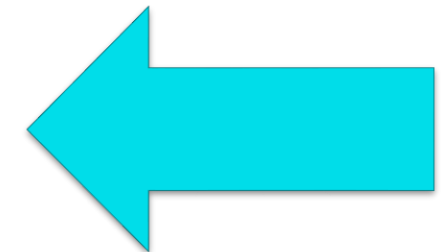
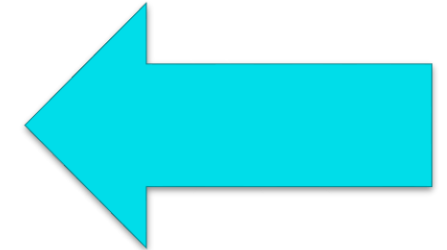
```
            Integer y = (Integer) stack.pop();
```

```
            System.out.println(y);
```

```
        }
```

```
    }
```

Anne Ushanova



2

- **PEEK** - Показва елемента от върха на стека
- **SEARCH** – търси елемент в стека.  
Извежда -1, ако не присъства такъв елемент или индекса му, ако го има в стека

```
// Displaying element on the top of the stack
static void stack_peek(Stack<Integer> stack)
{
    Integer element = (Integer) stack.peek();
    System.out.println("Element on stack top : " + element);
}

// Searching element in the stack
static void stack_search(Stack<Integer> stack, int element)
{
    Integer pos = (Integer) stack.search(element);
    if(pos == -1)
        System.out.println("Element not found");
    else
        System.out.println("Element is found at position " + pos);
}
```

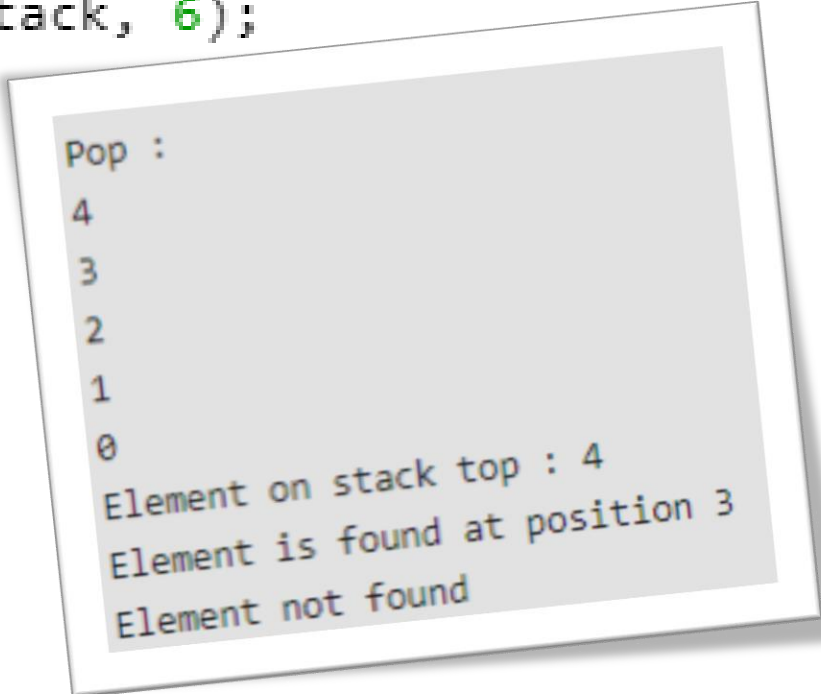
Търсене на  
елемент в стек

- Създаване на стек;
- Инициализация на стека;
- Извеждане на елементите му;
- Повторна инициализация;
- Извеждане на елемента от върха на стека;
- Търсене на елемент

```
public static void main (String[] args)
{
    Stack<Integer> stack = new Stack<Integer>();

    stack_push(stack);
    stack_pop(stack);
    stack_push(stack);
    stack_peek(stack);
    stack_search(stack, 2);
    stack_search(stack, 6);
}
```

3



```
Pop :
4
3
2
1
0
Element on stack top : 4
Element is found at position 3
Element not found
```

# Stack empty() Method in Java



# empty()

- Методът `java.util.Stack.empty ()` в Java се използва за проверка дали стека е празен или не.
- Методът е от `boolean type` и връща `true`, ако стекът е празен, а друг `false`.
- Syntax:

```
STACK.empty()
```

## Пример 1 за STACK.empty()

```
1 // Java code to demonstrate empty() method
2 import java.util.*;
3
4 public class Stack_Demo {
5     public static void main(String[] args)
6     {
7
8         // Creating an empty Stack
9         Stack<String> STACK = new Stack<String>();
10
11        // Stacking strings
12        STACK.push("SOFIA");
13        STACK.push("VOCATIONAL");
14        STACK.push("SCHOOL");
15        STACK.push("OF");
16        STACK.push("ELECTRONICS");
17    }
```

1

## Пример 1 за STACK.empty()

```
17
18 // Displaying the Stack
19 System.out.println("The stack is: " + STACK);
20
21 // Checking for the emptiness of stack
22 System.out.println("Is the stack empty? " +
23                     STACK.empty());
24
25 // Popping out all the elements
26 STACK.pop();
27 STACK.pop();
28 STACK.pop();
29 STACK.pop();
30 STACK.pop();
31
32 // Checking for the emptiness of stack
33 System.out.println("Is the stack empty? " +
34                     STACK.empty());
35 }
36 }
37
```

The stack is: [SOFIA, VOCATIONAL, SCHOOL, OF, ELECTRONICS]  
Is the stack empty? false  
Is the stack empty? true

## Пример 2 за STACK.empty()

```
// Java code to demonstrate empty() method
import java.util.*;

public class Stack_Demo {
    public static void main(String[] args)
    {

        // Creating an empty Stack
        Stack<Integer> STACK = new Stack<Integer>();

        // Stacking int values
        STACK.push(8);
        STACK.push(5);
        STACK.push(9);
        STACK.push(2);
        STACK.push(4);

        // Displaying the Stack
        System.out.println("The stack is: " + STACK);

        // Checking for the emptiness of stack
        System.out.println("Is the stack empty? " +
                           STACK.empty());

    }
}
```

### Output:

The stack is: [8, 5, 9, 2, 4]  
Is the stack empty? false

# Задачи за домашно

## 1 задача REVERSE

- Напишете програма `Reverse.java`, който чете низове един по един от стандартен вход и ги отпечатва на стандартния изход в обратен ред.

# Примерно решение

```
public class Reverse {  
  
    public static void main(String[] args) {  
        Stack<String> stack = new Stack<String>();  
        while (!StdIn.isEmpty()) {  
            String s = StdIn.readString();  
            stack.push(s);  
        }  
        while (!stack.isEmpty()) {  
            String s = stack.pop();  
            StdOut.println(s);  
        }  
    }  
}
```

## 2 задача REVERSE

- Напишете програма ReverseArr.java, който чете елементите на едноредов целочислен масив, записва ги в стек и ги отпечатва на стандартния изход в обратен ред.



## 3 задача

- Какво ще отпечата следният фрагмент от код, когато  $n$  е 50? Дайте описание на това, което прави фрагментът от кода, когато е представен с положително цяло число  $n$ .

```
Stack stack = new Stack();  
while (n > 0) {  
    stack.push(n % 2);  
    n /= 2;  
}  
while (!stack.isEmpty())  
    StdOut.print(stack.pop());  
StdOut.println();
```

## 3 задача

/отпечатва  
двоичното  
представяне  
на n (110010,  
когато n е 50/

- Какво ще отпечатва следният фрагмент от код, когато n е 50?
- Дайте писмено описание на това, което прави фрагментът от кода, когато е представен с положително цяло число n.

```
Stack stack = new Stack();  
while (n > 0) {  
    stack.push(n % 2);  
    n /= 2;  
}  
while (!stack.isEmpty())  
    StdOut.print(stack.pop());  
StdOut.println();
```