

АБСТРАКТНИ ТИПОВЕ ДАННИ

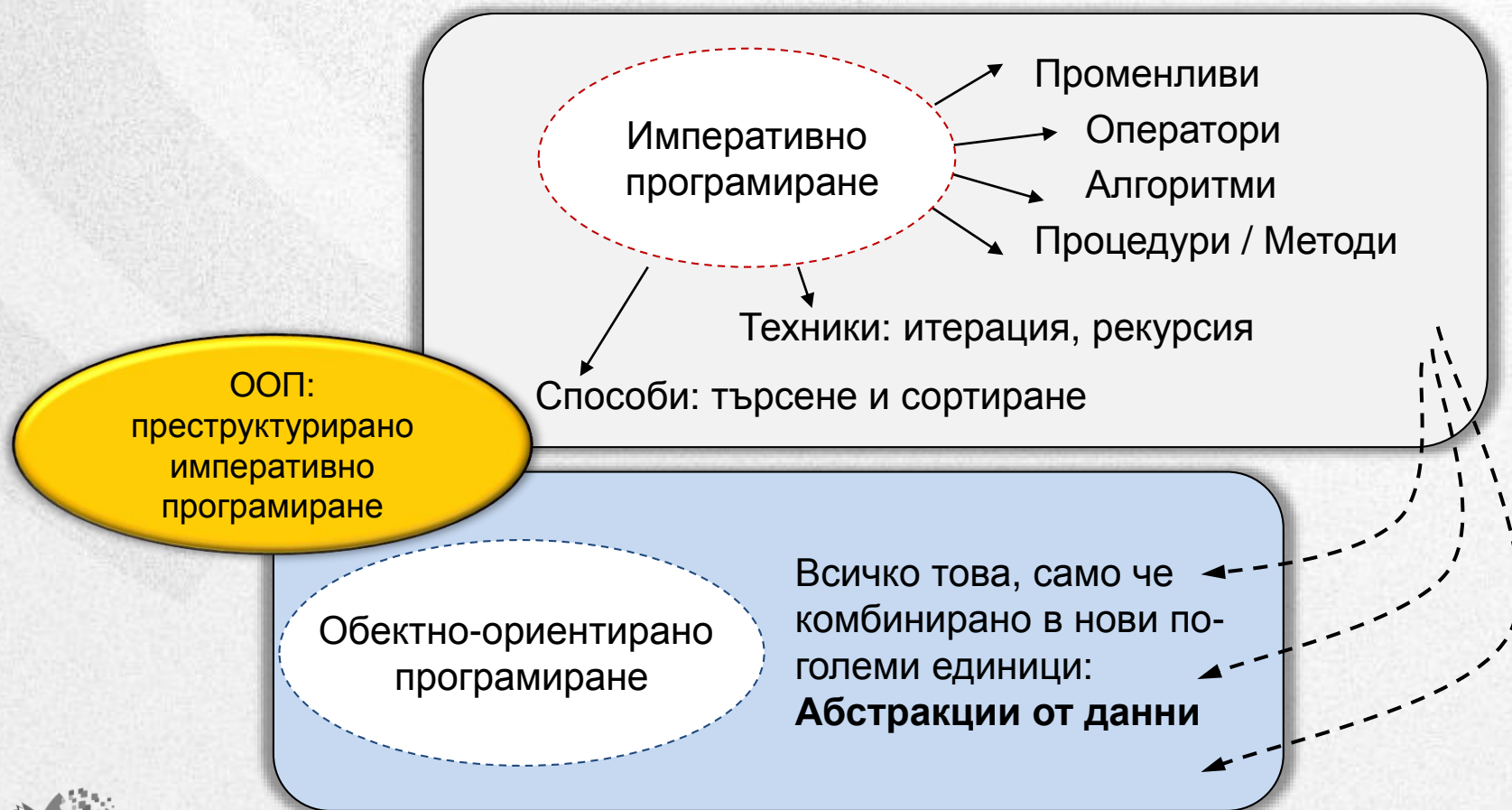
ЛЕКЦИОНЕН КУРС “ПРОГРАМИРАНЕ НА JAVA”



СТРУКТУРА НА ЛЕКЦИЯТА

- Парадигми: императивно и обектно-ориентирано програмиране
- Абстракции данни: абстрактни типове данни (ADT), класове, инстанции
- Променливи и методи на инстанции
- Пример: стек
- Обобщение

ИМПЕРАТИВНО ПРОГРАМИРАНЕ: ОСНОВНО ПОМОЩНО СРЕДСТВО НА ООП



СПРАВЯНЕ С КОМПЛЕКСНОСТТА НА ПРОГРАМИТЕ: ПОВТОРЕНИЕ

- **Декомпозиция:**

- разлагане на програмата на компоненти (модули)

- **Абстракция:**

- премахване на несъщественото за използването на един модул

АБСТРАКТНИ ТИПОВЕ ДАННИ

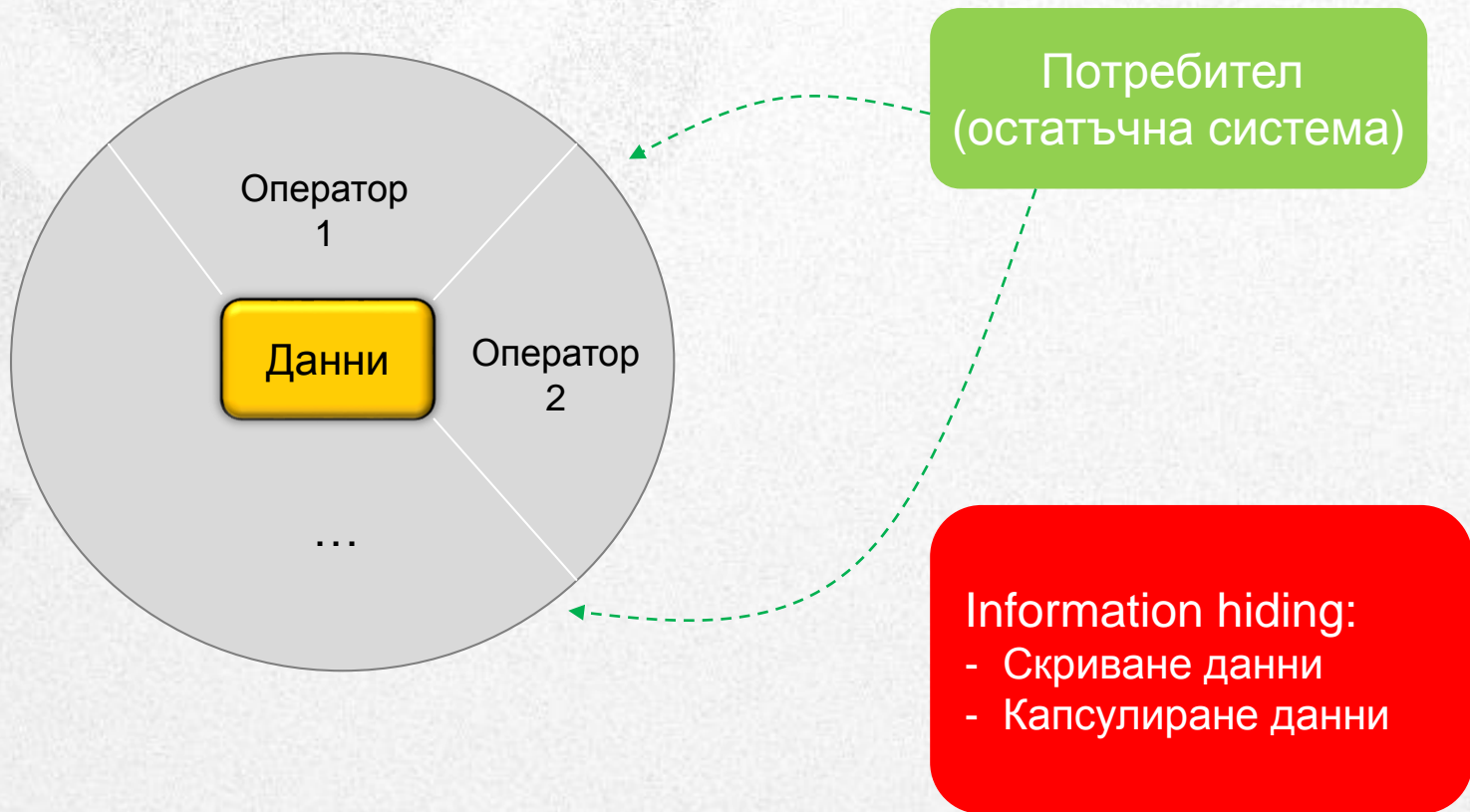
Единица от данни и операции

- **Операции:** служат за обработка на данни (инициализация, промяна, четене, изтриване)
- **Данни:** защитени/скрити от “външния свят”

Бележки:

- Основен принцип на разработването на софтуер: "information hiding"
- Сравнение с императивното програмиране: данни и алгоритми/операции са **разделени**

АБСТРАКТНИ ТИПОВЕ ДАННИ:(INFORMATION HIDING)



ТИПИЧЕН ПРИМЕР: АБСТРАКТЕН ТИП ДАННИ 'STACK'

1. Да се прочете един набор от данни (int/char) и да се изведе в обратен ред

Вход: a z d f g k

Изход: k g f d z a

2. Да се провери структурирането на скобите в една програма (появяване по двойки, без друг синтактичен анализ)

```
( a + ( x [ ( i + j ) ] % 12 ) {  
    z [ i ] ++ ;  
} ...
```

3. Да се провери Postfix формата на изрази

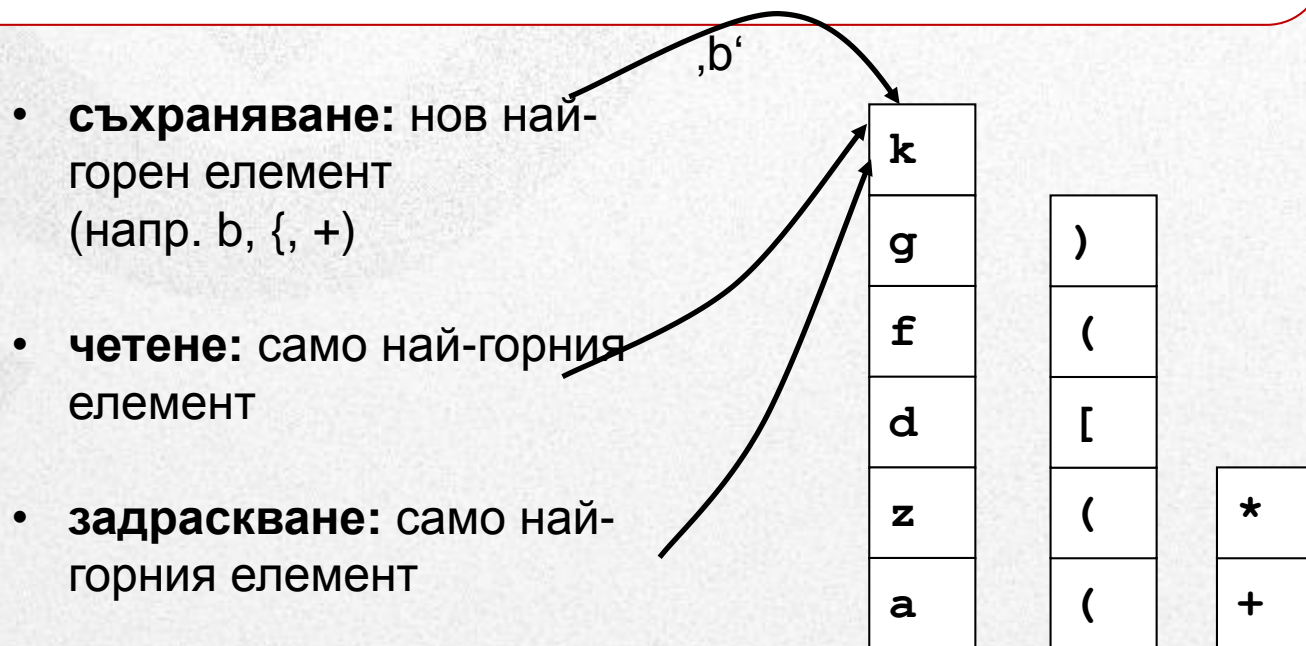
```
a + b * c -> a b c * +  
(a + b) * c -> a b + c *
```

4. Заместване на рекурсия с итерация

РЕШЕНИЕ: ИЗПОЛЗВАНЕ НА СТЕК

Стек (stack):

- Абстрактен тип данни;
- Последователност от елементи, които могат да бъдат обработвани (четене, съхраняване, задраскване), само от едната страна („отгоре“)
- LIFO-принцип (Last In First Out)



Опашка (queue): FIFO-принцип (First In First Out)

ПРЕДСТАВЯНЕ НА ДАННИ В СТЕКОВЕ

1. Масив с индекс за последния елемент

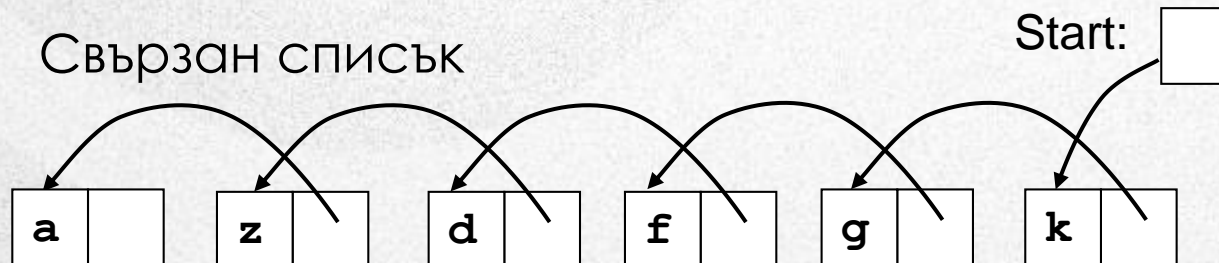
char [] stack:

int top:

5



2. Свързан списък



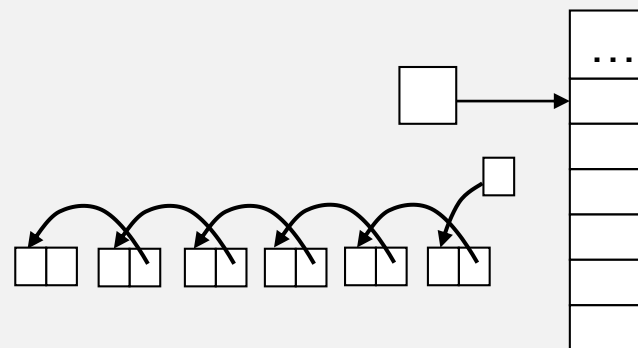
Използване на стек

- несъществено: представянето на данни
- съществено: операторите за достъп

ПОТРЕБИТЕЛСКИ АСПЕКТ

- **Несъществено:**

- представянето на данни (масиви или списъци)



- **Съществено:**

- операции за достъп или с какви оператори могат да бъдат обработвани данните?

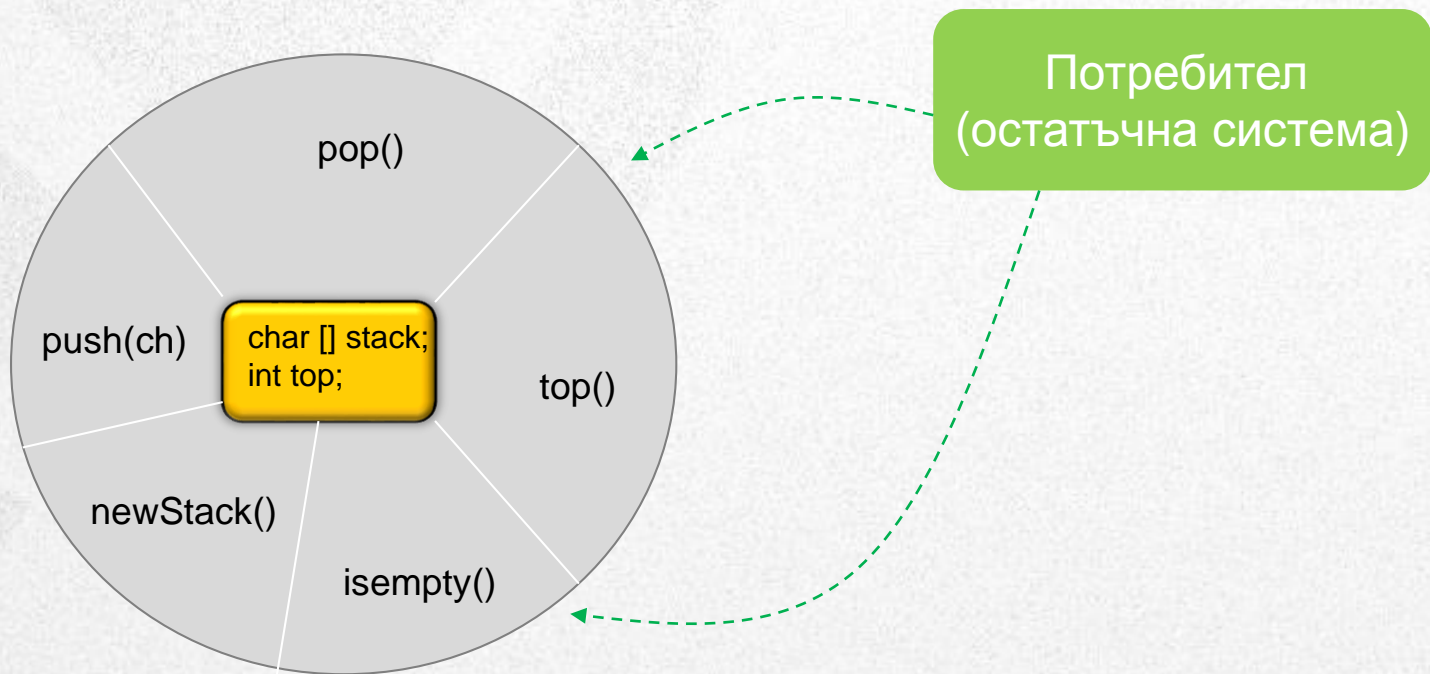
ОПЕРАЦИИ ЗА СТЕКОВЕ

- създава: празен стек (с дължина n)
- съхранява: нов елемент на върха
- чете: чете най-горния елемент
- задрасква: премахва най-горния елемент
- тества: празен ли е стека?

Сигнатура: (име, дефиниционна област, област на стойностите)

newStack:	int	→ stack
push:	stack x char	→ stack
top:	stack	→ char
pop:	stack	→ stack
isempty:	stack	→ boolean

АБСТРАКТНИ ТИПОВЕ ДАННИ:(INFORMATION HIDING)



ПРОГРАМА СТЕК

```
class Stack {  
    private char[] stackElements;  
    private int top; // shows the top element  
    public Stack(int n) {  
        stackElements = new char [n];  
        top = -1;  
    }  
    public boolean isempty() {  
        return top == -1;  
    }  
    public void push(char x) {  
        top++; stackElements[top] = x;  
    }  
    public char top() {  
        if (isempty()) {  
            System.out.println("Stack empty");  
            return ' ';  
        }  
        else  
            return stackElements [top];  
    }  
    public void pop() {  
        if (isempty())  
            System.out.println("Stack empty");  
        else  
            top--;  
    }  
}
```

ADT → Java-Class

АБСТРАКТНИ ТИПОВЕ ДАННИ В JAVA

Променливи: 'private' (скрити)

```
class Stack {  
    private char[] stackElements;  
    ...  
    public Stack(int n) {  
        stackElements = new char [n];  
        top = -1;  
    }  
    public void push(char x) {  
        ...  
    }  
}
```

Методи: 'public' (видими 'навън')

Променливи, методи: не повече 'static'
→ променливи, методи на инстанции

ЗА СРАВНЕНИЕ: ИМПЕРАТИВНА ПРОГРАМА

```
class TimePlan {  
  
    private static int hour, minute;  
  
    private static void addMinutes (int m)  
    private static int timeInMinutes ()  
    private static void printTime ()  
    private static void printTimeInMinutes ()  
    private static void includeNewEntry  
                        (int intervalInMinutes, String event)  
    public static void main (String[] args) {  
        ...  
        includeNewEntry(90, „L PiJ");  
        includeNewEntry(15, "Pause");  
        includeNewEntry(90, „L IiDB");  
        ...  
    }  
}
```

JAVA-КЛАСОВЕ: ДЕФИНИРАНЕ ADT

- Един клас дефинира нов (абстрактен) тип
- Type name = Class name
- Деклариране на променливи: от същия тип
- Стойностите на новия тип се наричат **Обекти** или **Инстанции** на класа

```
int i;  
int[] sorted;  
Stack s;
```

- Създаване на един обект:

Метод-конструктор (име на клас)

```
s = new Stack(n) ;
```

сравни Arrays: декларация + създаване

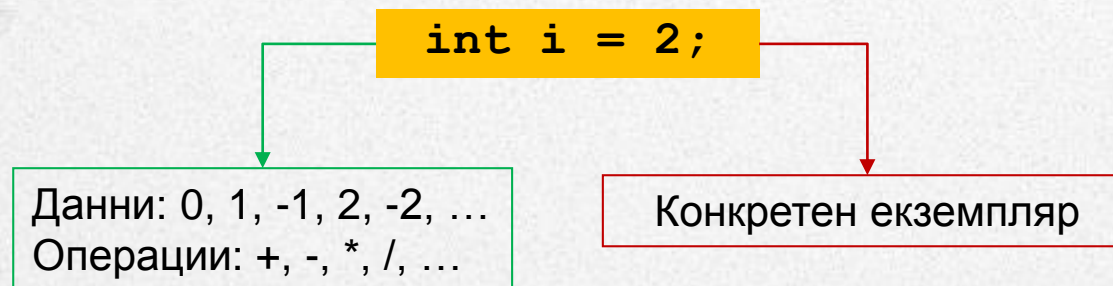
В променливата се съхранява само референция към обекта. Самият обект се пласира на специално място в паметта, наречено heap

КЛАСОВЕ И ОБЕКТИ

Клас = описание на новия тип (данни и операции)

Обект = конкретен екземпляр от типа (съществува физически в паметта)

Сравнение с предварително дефинирани типове данни:



КЛАСОВЕ ЗА ОПИСАНИЕ НА ADT

```
class Stack {  
  
    private char[] stackElements;  
    private int top;  
  
    public Stack(int n) {  
        ...  
    }  
  
    public void top() {  
        ...  
    }  
  
    public boolean isempty() ...  
  
}
```

Променлива на инстанция:
Описание на данни
(представяне)

**Създаване и
инициализиране:** Конструктор
(метод)

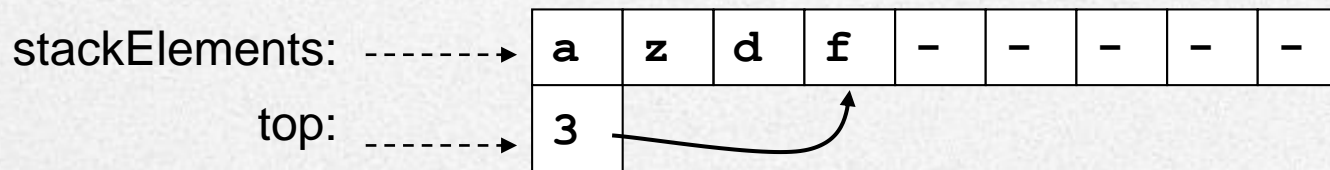
Методи на инстанция:
описание на оператори

ПРЕДСТАВЯНЕ НА ДАННИ В КЛАСА STACK

```
class Stack {  
    private char[] stackElements;  
    private int top;  
    ...  
}
```

Съответства на 1. вариант (виж горе):
масив с указател към последния
елемент

Пример за стек с дължина 4:

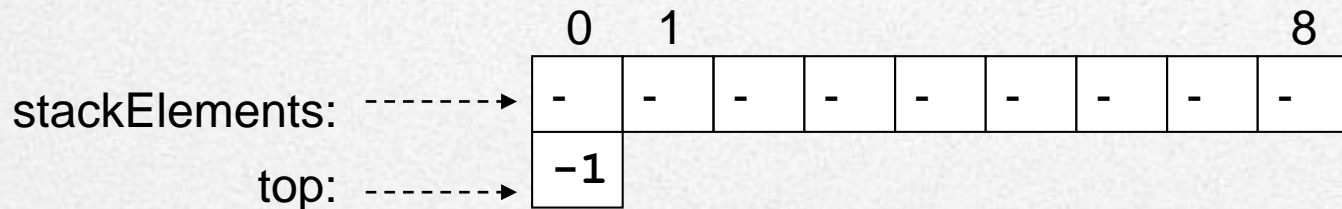


ИНИЦИАЛИЗИРАНЕ НА СТЕК

Конструктор Stack() създава един обект и инициализира променливите (на инстанцията)

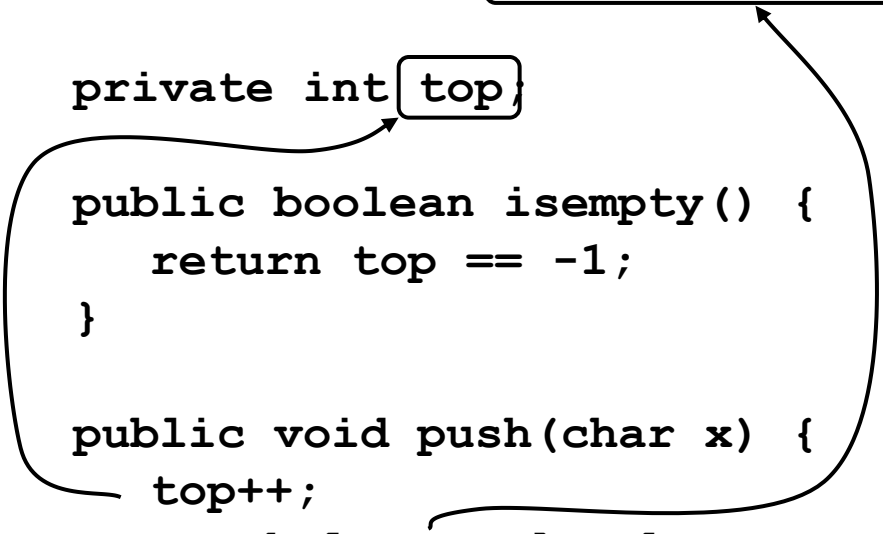
```
class Stack {  
    private char[] stackElements;  
  
    private int top;  
  
    public Stack(int n) {  
        stackElements = new char [n];  
        top = -1;  
    }  
    ...  
}
```

Пример за извикване: `s = new Stack(9);`



МЕТОДИ В СТЕКОВЕ

```
class Stack {  
    private char[] stackElements;  
  
    private int top;  
  
    public boolean isempty() {  
        return top == -1;  
    }  
  
    public void push(char x) {  
        top++;  
        stackElements[top] = x;  
    }  
    ...  
}
```



МЕТОДИ НА ИНСТАНЦИИ: ИЗВИКВАНЕ

Използване на обекти:

```
Variable.method ( Parameter )
```

Примери:

```
Stack s;  
s = new Stack(9);  
  
s.push( '+' );  
s.pop( );  
x = s.top( );
```

съхранява (push) символа '+' в
обекта, рефериран от
променливата s

ИЗПОЛЗВАНЕ НА СТЕК: ОБРЪЩАНЕ НА СИМВОЛНИ НИЗОВЕ

```
public class Turn {
    public static void main (String[] argv) {

        int n;
        char ch;
        Stack s;

        System.out.print("Stack lenght: ");
        n = Keyboard.readInt();
        s = new Stack(n);

        // read n elements and write in the stack
        System.out.println(" Enter min " + n + " symbols:");
        for (int i=0; i < n; i++) {
            ch = Keyboard.readChar();
            s.push(ch);
        }

        System.out.println
            (" Turned sequence of the first " + n + " symbol:");
        while (!s.isEmpty()) {                // while stack not empty:
            System.out.print(s.top());         // print and delete
            s.pop();                           // top element
        }
        System.out.println();
    }
}
```

Turn.java

ДЕЙСТВИЕ НА ОБРЪЩАНЕТО

```
% java Turn
```

```
Stack length: 5
```

```
Enter min 5 symbols:
```

```
abcde
```

```
Turned sequence of the first 5 symbols:
```

```
edcba
```


ОБРЪЩАНЕ НА СИМВОЛЕН НИЗ: СЪЗДАВАНЕ НА ОБЕКТ

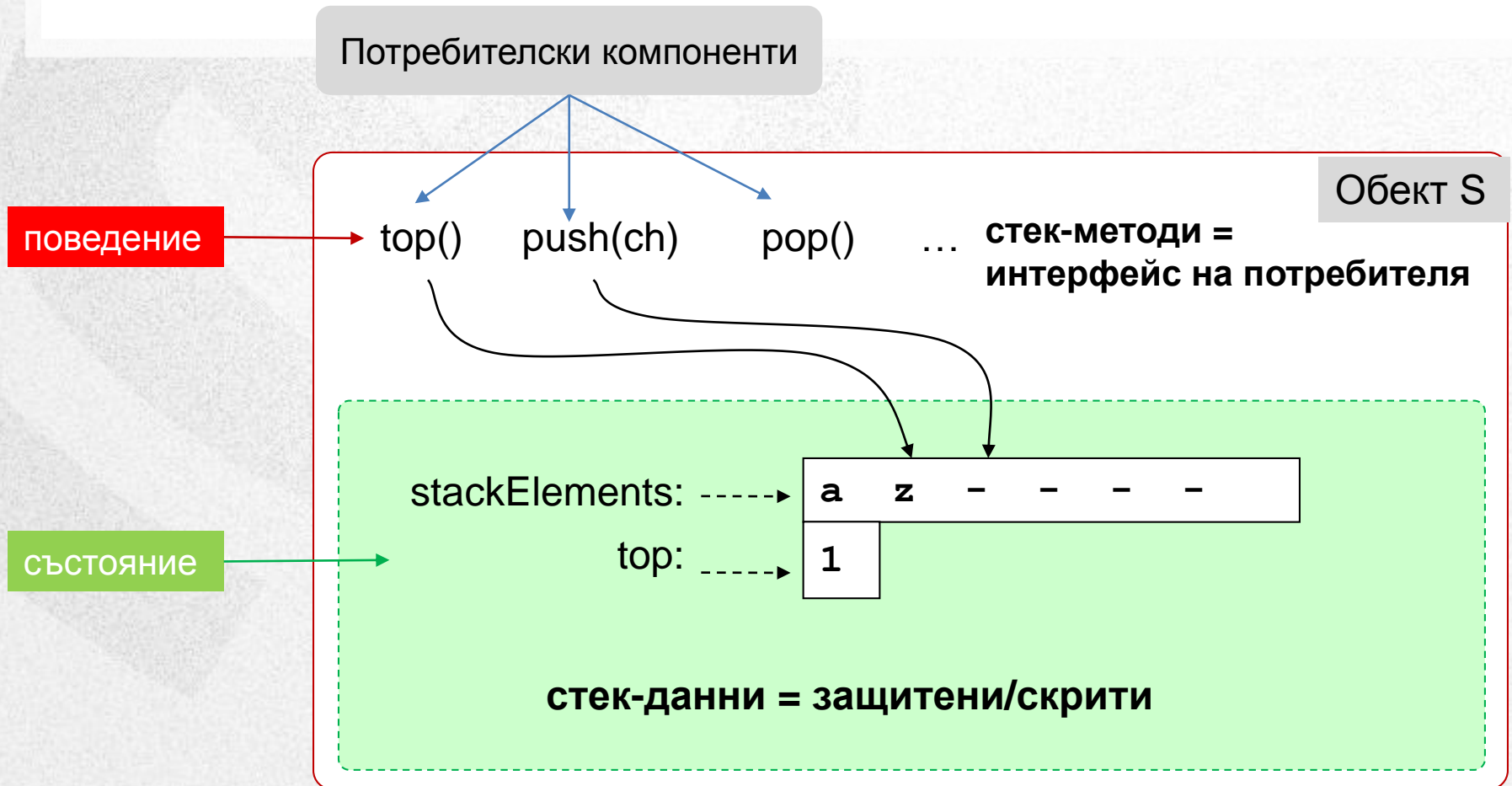
```
public static void main (String[] argv) {  
  
    int n;  
    char ch;  
    Stack s;  
  
    System.out.print("Stack lenght: ");  
    n = Keyboard.readInt();  
    s = new Stack(n);  
    ...  
}
```

ОБРЪЩАНЕ НА СИМВОЛЕН НИЗ: СЪХРАНЯВАНЕ И ИЗВЕЖДАНЕ НА ЕЛЕМЕНТИ

```
// n elements read and save in the stack
System.out.println("Enter ... :");
for (int i=0; i < n; i++) {
    ch = Keyboard.readChar();
    s.push(ch);
}

System.out.println("Turned sequence ... :");
while (!s.isEmpty()) {
    // while stack not empty:
    // print and delete the top element
    System.out.println(s.top());
    s.pop();
}
```

ОБЕКТИ = ЕДИНИЦИ ОТ СКРИТИ ДАННИ И ИНТЕРФЕЙСНИ МЕТОДИ




Booch: “един обект има идентичност, състояние и поведение”

ПРОМЯНА СЪСТОЯНИЕТО НА ОБЕКТИТЕ ЧРЕЗ ИЗВИКВАНЕ НА МЕТОДИ (1)


Декларация

```
Stack s;
```

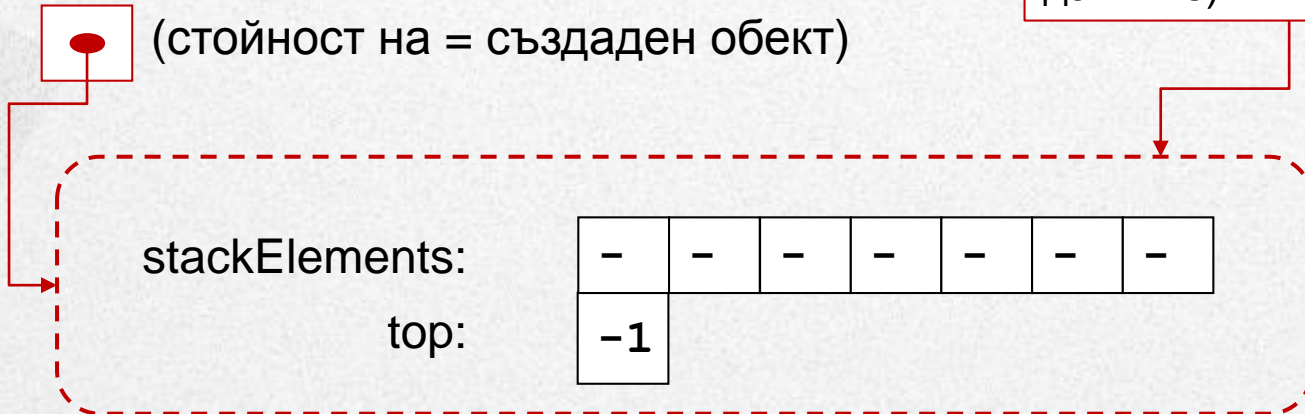
s :  (без стойност)

Създаване

```
s = new Stack(7);
```

s :  (стойност на = създаден обект)

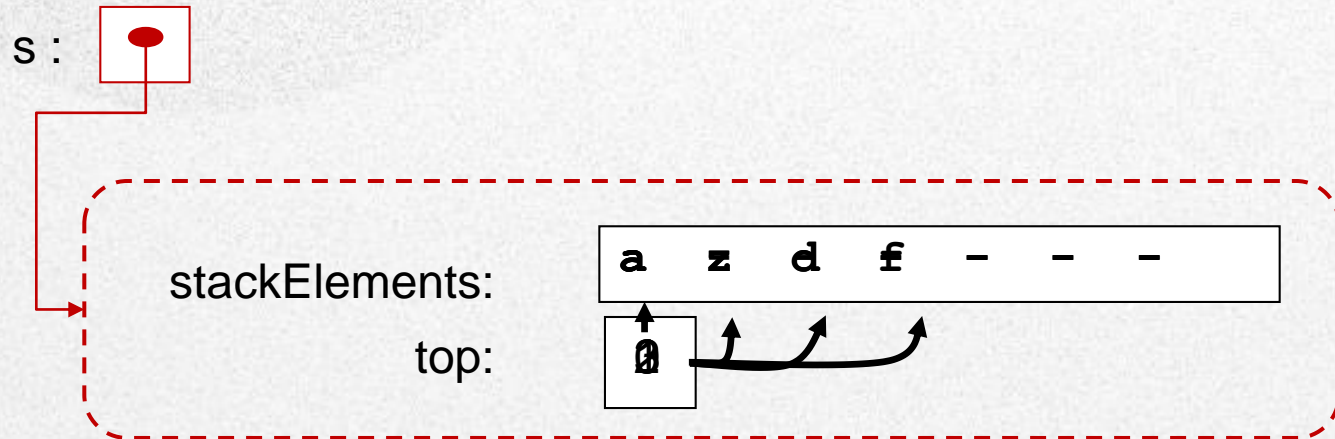
Състояние на обект
(актуални стойности на
данните)



ПРОМЯНА СЪСТОЯНИЕТО НА ОБЕКТТИТЕ ЧРЕЗ ИЗВИКВАНЕ НА МЕТОДИ (2)

Съхраняване на елементи:

```
s.push('a');  
s.push('z');  
s.push('d');  
s.push('f');
```



ПОВЕЧЕ ИНСТАНЦИИ НА КЛАС „STACK“

Задача:

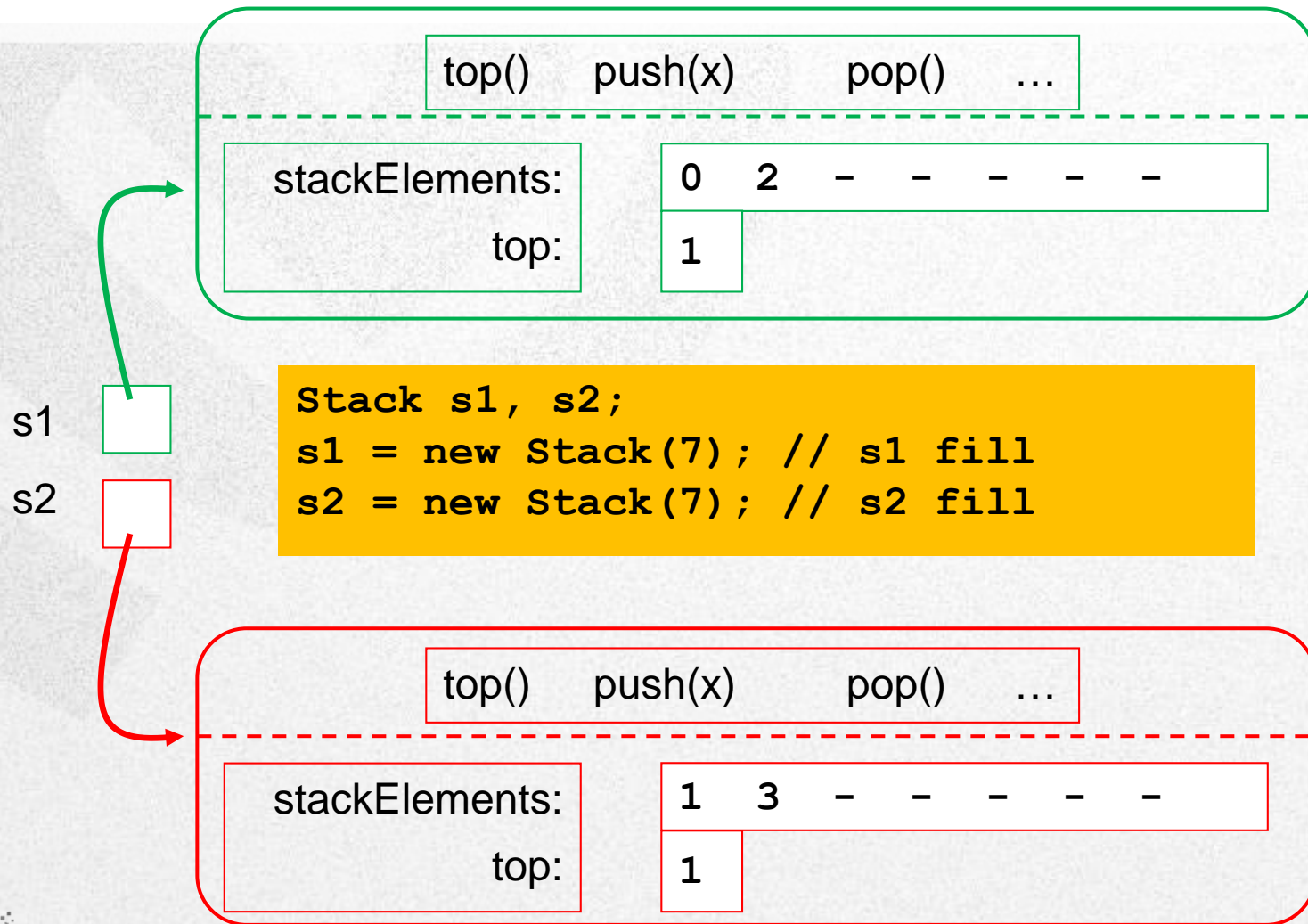
- Входните данни разделени на 2 части (четна/нечетна позиция)
- Всяка част се извежда в обратна последователност

Вход: 0123456789

Изход: 97531
86420

```
Stack s1, s2;  
...  
s1 = new Stack(n);  
s2 = new Stack(n);
```

ОБЕКТИ (ИНСТАНЦИИ): ВСЯКА СЪС СОБСТВЕНИ ПРОМЕНЛИВИ И МЕТОДИ



МЕТОДИТЕ, СВЪРЗАНИ С ИНСТАНЦИЯТА

```
Stack s1, s2;  
if (i % 2 == 0)  
    s1.push(ch);  
else  
    s2.push(ch);  
  
...  
  
while (!s2.isEmpty()) {  
    System.out.println(s2.top());  
    s2.pop();  
}
```

Метод push, свързан с
s1 съотв. s2

isEmpty() принадлежи тук
към s2

Допълнителен параметър на оператори:
Instance (Object) s1 / s2 ...
(вътрешен параметър)

ОБОБЩЕНИЕ

- Методите са основен механизъм за капсулиране на код
- Обикновено методите връщат някакъв резултат
 - return value
 - Могат да се използват в изрази
- Методите могат да не връщат резултат
 - return
 - Могат да се използват като самостоятелни оператори
- В терминологията на ООП извикване на метод се нарича “изпращане на съобщение към обект”

БЛАГОДАРЯ ЗА ВНИМАНИЕТО!

КРАЙ “АБСТРАКТНИ ТИПОВЕ ДАННИ”

