4. Условни оператори

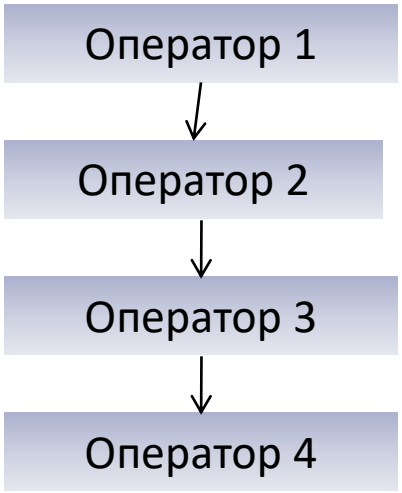
Лекционен курс “Програмиране на Java”
проф. д-р Станимир Стоянов

Структура на лекцията

- Контролен поток
- Оператори за избор
- “Висящ” else

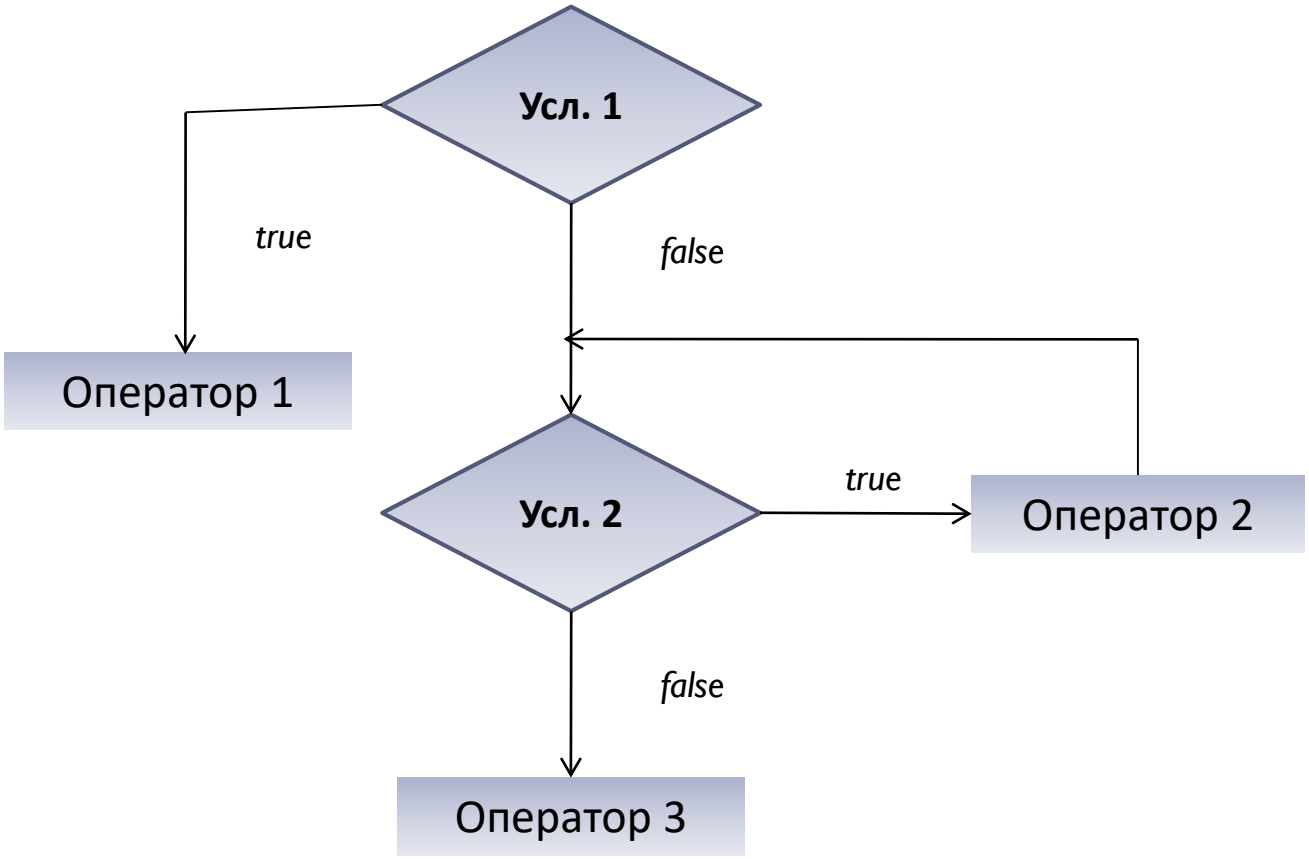
Контролен поток (control flow)

- Последователността на изпълняваните в програмата оператори
- Условия и цикли: позволяват ни да хореографираме контролния поток



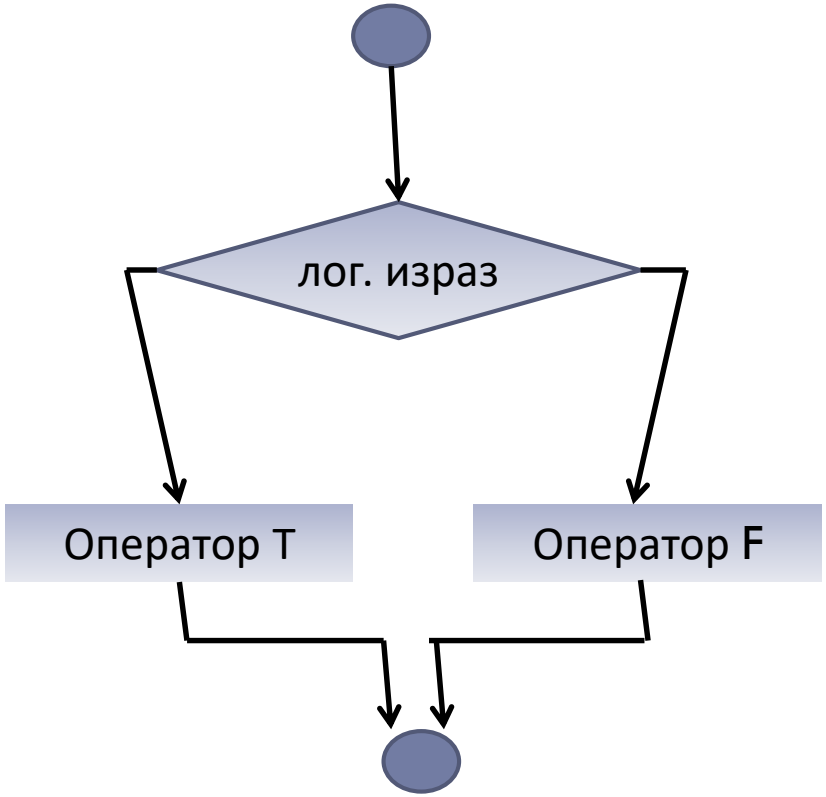
Последователен контролен поток

Контролен поток с условия и цикли



if оператор: разклоняваща структура

```
if (логически израз) {
    оператор T;
}
else {
    оператор F;
}
```



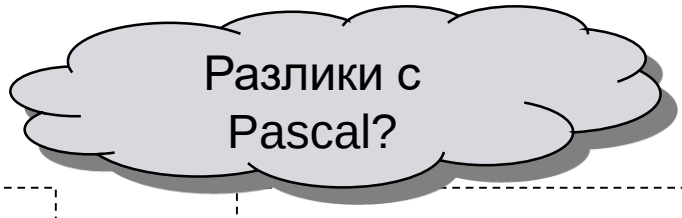
Синтаксис

- If-оператор: избор между две алтернативи

EBNF: **`if`** (израз) оператор **`else`** оператор

→ За всяка алтернатива: един оператор !

Примери:




```
if (x == 0)
   System.out.print(0);
else
   System.out.print(y/x);
```

```
if (x >= y)
   max = x;
else
   max = y;
```

If-оператор: повече оператори

EBNF:

if (израз) оператор **else** оператор

Повече отделни оператори?

→ с { ... } обединени като един

Пример:

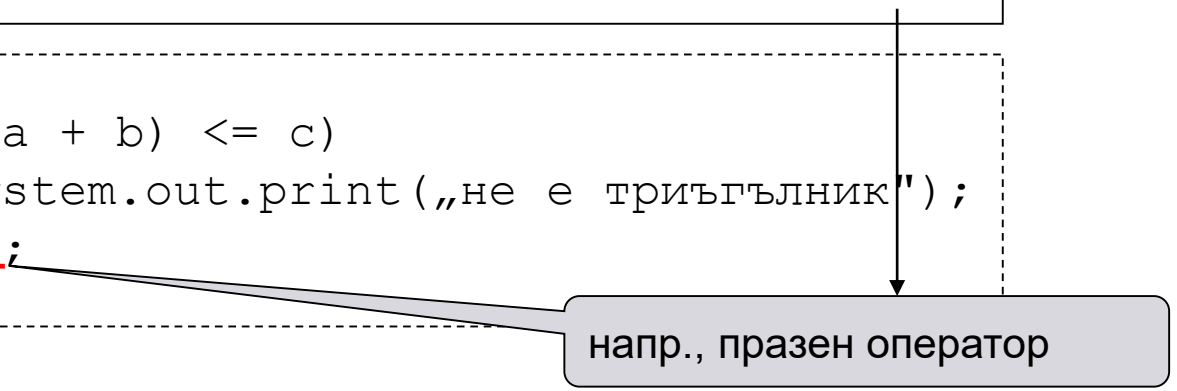
```
if (x == 0)    {    //Exception: Div by 0
    System.out.print(0);
    x = y;
}
else    {
    System.out.print(y/x);
    y = x;
}
```

If-оператор: кратка форма

EBNF: Пълна форма

```
if ( израз ) оператор else оператор
```

```
if ((a + b) <= c)
   System.out.print(„не е триъгълник");
else ;
```

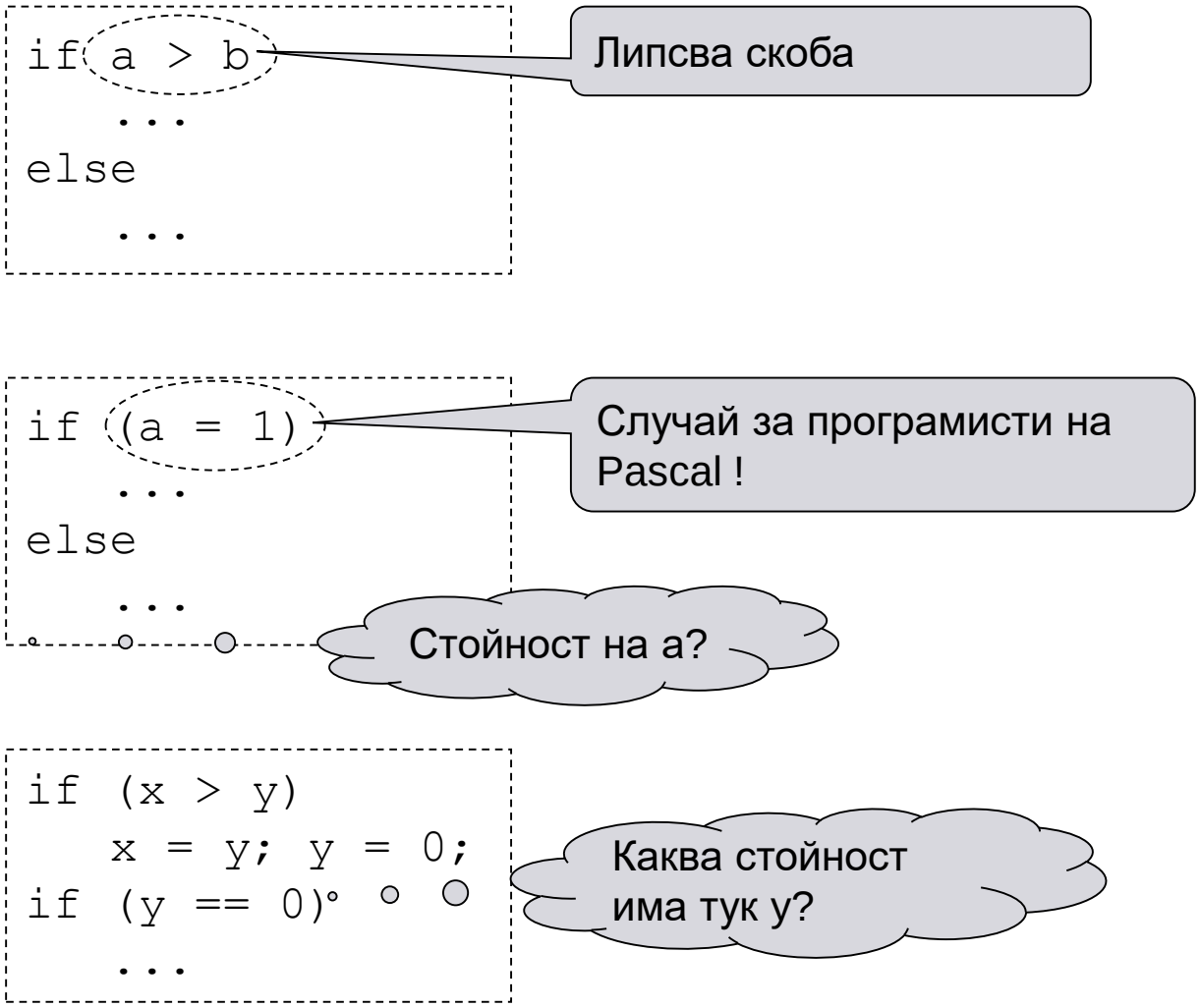


EBNF: Кратка форма = спец. случай на пълната форма

```
if ( израз ) оператор
```

```
if ((a + b) <= c)
   System.out.print(“не е триъгълник");
```

Типични грешки



“Висящ” else (dangling else)

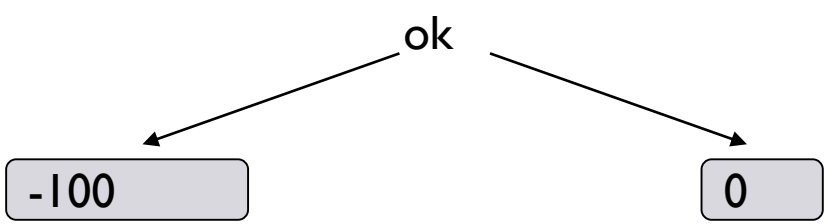
Входни стойности:

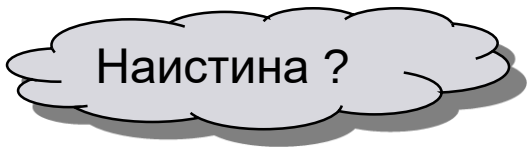
x	y	z	ok
2	1	100	-100

```
if  (x > y)
   if  (y > z)
       ok = 1;
else
   ok = 0;
```

```
if  (x > y)
   if  (y > z)
       ok = 1;
   else
       ok = 0;
```

Резултат :

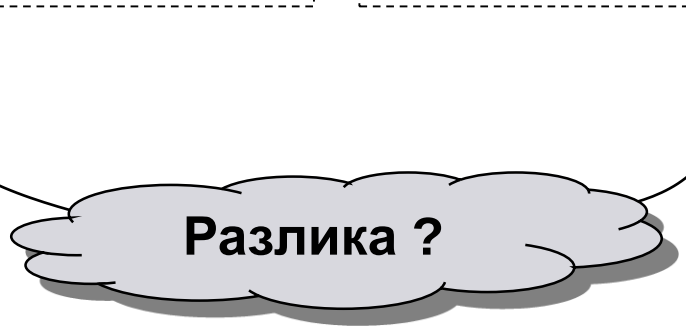




“Висящ” else: проблем

```
if   (a)
    if   (b)
        s1;
else
    s2;
```

```
if   (a)
    if   (b)
        s1;
    else
        s2;
```



Двата варианта: идентични

Само по различен начин изразен (Layout)

→ намерение на програмиста:

else s2; принадлежи към …

- 1. Форма: външен if
- 2. Форма: вътрешен if

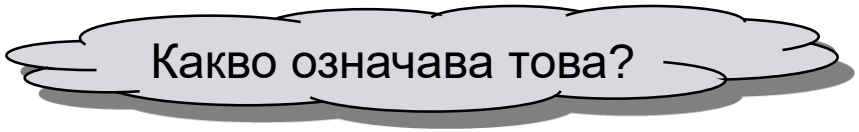
Значение на една програма: независимо от Layout

```
if (a)   if (b)  s1;  else s2;
```

➔ Констатация: 2. форма е обвързваща !

“Висящ”-else-проблем: причина

Граматиката на Java не е еднозначна !

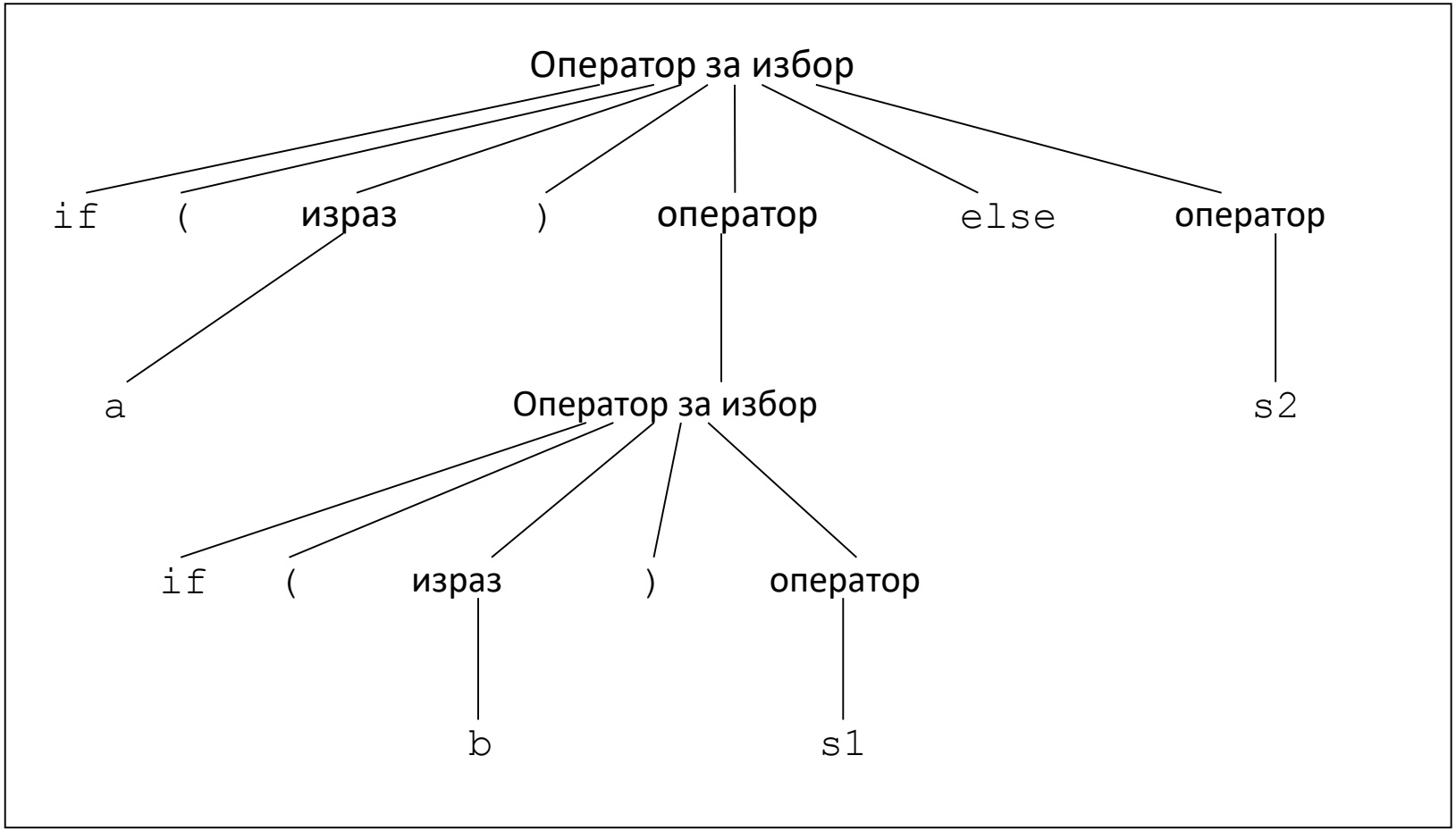


Оператор за избор ::=
    `if` (израз) оператор **`[ else`** оператор **`]`** |
    `switch ...`

3 правила

Оператор за избор ::=
    `if` (израз) оператор |
    `if` (израз) оператор **`else`** оператор |
    `switch ...`

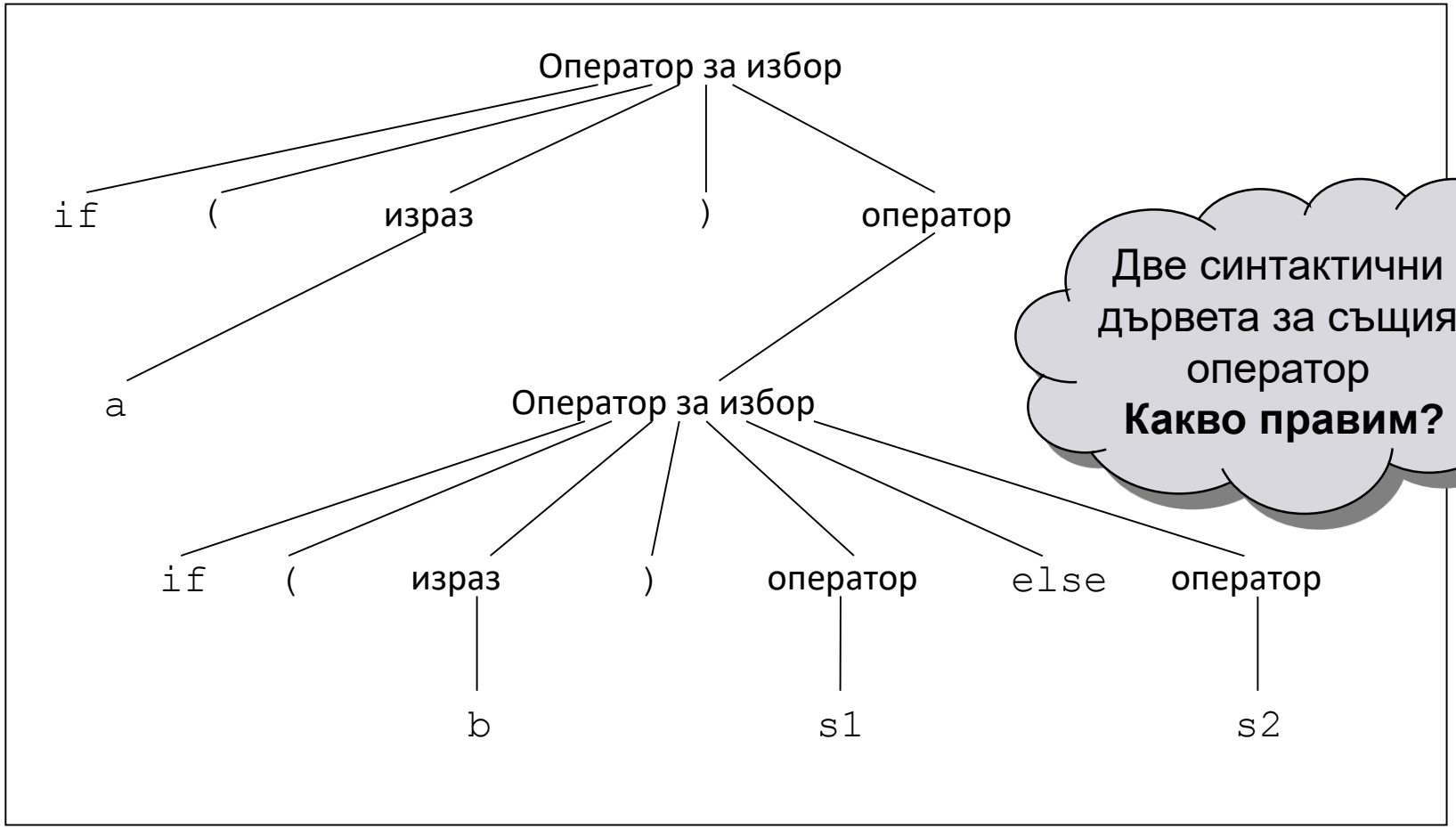
Синтактично дърво: Вариант 1



Изведен:

```
if ( a ) if ( b ) s1 else s2
```

Синтактично дърво: Вариант 2



Изведен: `if ( a ) if ( b ) s1 else s2`

Проблеми

- Какво правим: ако все пак искаме вариант I?

```
if (x > y)
    (y > z)
ok = 1;


= 0;
```

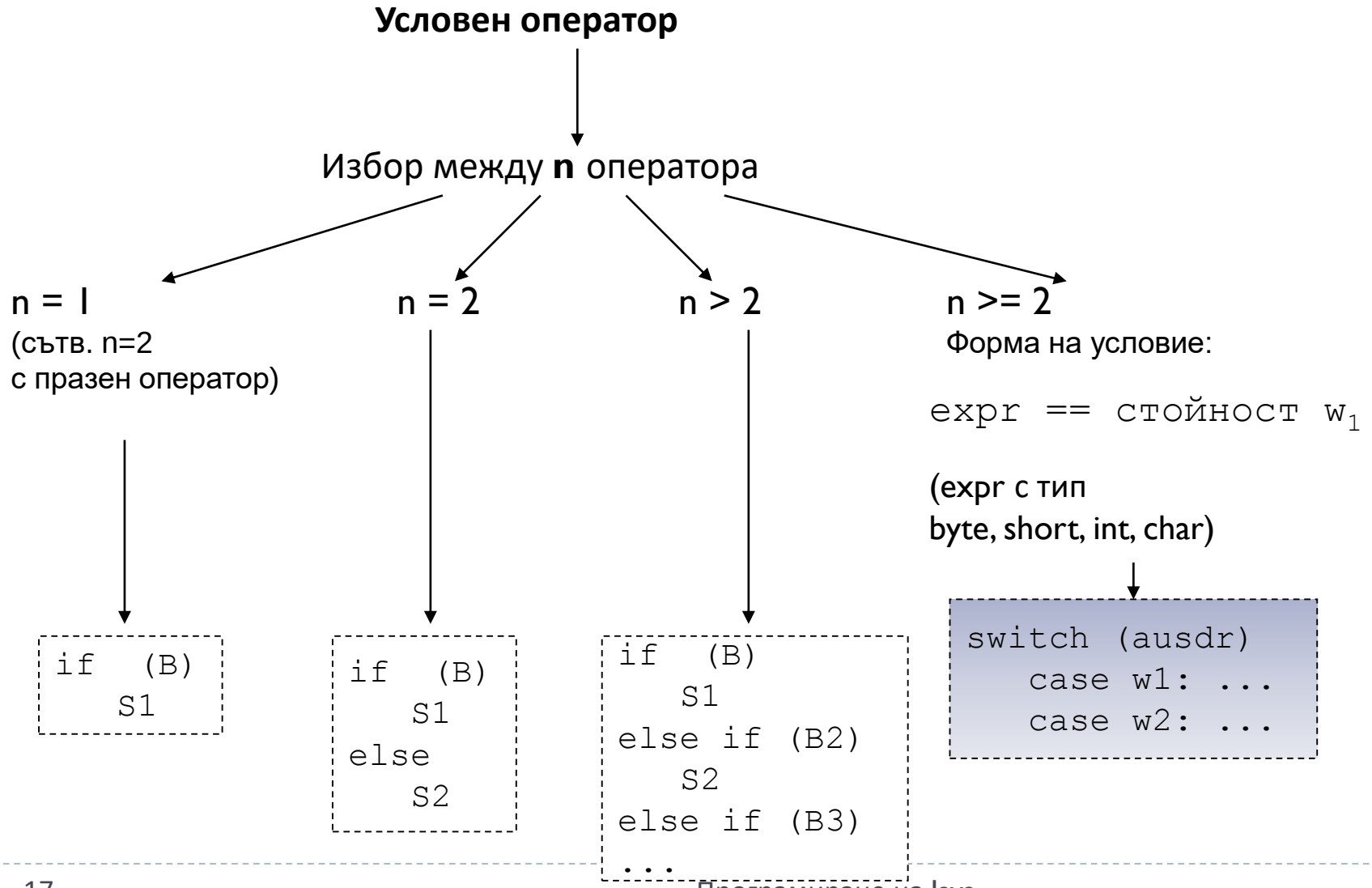
```
if (x > y) {
    if (y > z)
        ok = 1; }
else
    ok = 0;
```

```
if (x > y)
    if (y > z)
        ok = 1;
    else;
else
    ok = 0;
```

- Как би могъл да се направи синтаксисът еднозначен?

endif

Избор на условен оператор: switch



Switch-оператор: мотивация

```
if (n == 0)
   System.out.print(" нула ");
else if (n == 1)
   System.out.print(" едно ");
else if (n == 2)
   System.out.print("две");

. . . // else if до 9

else // n > 9
   System.out.print(" > девет
");
```

Стойност на израз (променлива n)

Повтарящо тестване при равенство с предварително зададени стойности (0, 1, 2 ...)

Switch-оператор:избор от повече варианти

```
switch (n)    {
   case 0:    System.out.print( "0" );
              break;
   case 1:    System.out.print( "1" );
              break;
   case 2:    System.out.print( "2" );
              break;
         ....
   default:   System.out.println( " >9" );
}
```

➔ **Семантиката се запазва**

Switch: пример

константа

```
final int jan = 1, feb = 2, mar = 3, ... dec = 12;
int month, year, numDays;
... // read month, year
switch (month) {
    case feb:
        if (((year % 4) == 0) && ((year % 100) != 0)
          || ((year % 400) == 0))  //high year test
            numDays = 29;
        else numDays = 28;
        break;
    case apr: case jun: case sep: case nov:
        numDays = 30;  break;
    default: numDays = 31;
}
```

инициализация

...and...

≠

...or...

Повече случаи едновременно

стандартен

Break-оператор

```
int n = 1;
switch (n)    {
   case 0:    System.out.print( "0" );
   case 1:    System.out.print( "1" );
   case 2:    System.out.print( "2" );
   case 3:    System.out.print( "3" );
}
System.out.println();
```

➔ Output ?

123

- Ако всеки оператор в switch завършва с break
  - Това е еквивалентно на вградена последователност от if оператори
- Предназначение на break
  - Завършва изпълнението на оператора switch в разклонението, където е даден
- По принцип всяка алтернатива на switch трябва да има break
  - При пропускане изпълнението на switch продължава до неговия край

Break-оператор

```
int n = 1;
switch (n)    {
   case 0:    System.out.print( "0" ); break;
   case 1:    System.out.print( "1" ); break;
   case 2:    System.out.print( "2" ); break;
   case 3:    System.out.print( "3" ); break;
}
System.out.println();
```

➔ Output ?

1

Благодаря за вниманието!

Край лекция 4. “Избор: условни оператори”