ИТЕРАЦИИ

ЛЕКЦИОНЕН КУРС "ПРОГРАМИРАНЕ НА JAVA"

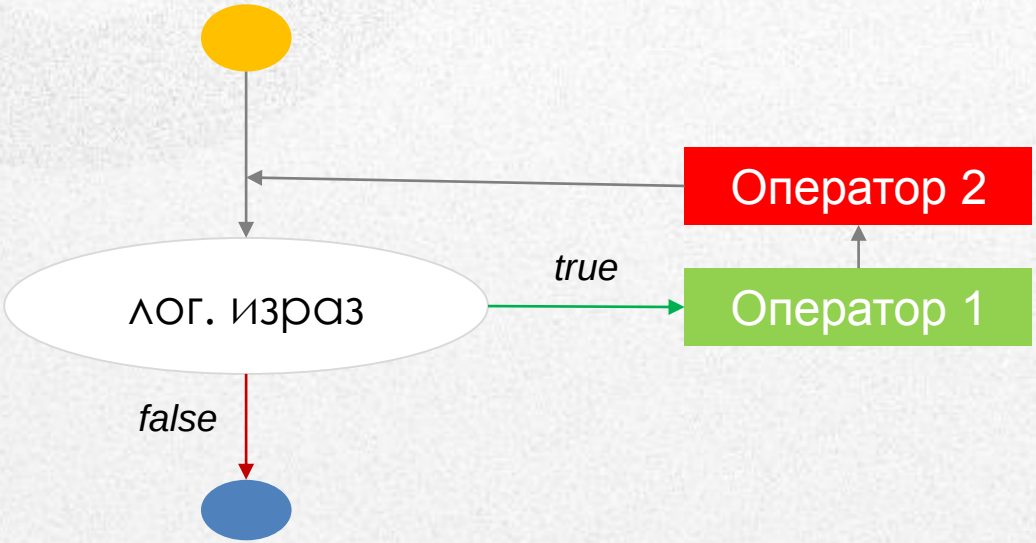
Distributed @learning center

СТРУКТУРА НА ЛЕКЦИЯТА

- Циклични контролни потоци
- Оператори
- Примери
- Класификация на операторите

WHILE ОПЕРАТОР: ЦИКЛИЧНА СТРУКТУРА

```
while (логически израз) {
   оператор 1;
   оператор 2;
}
```



СИНТАКСИС

EBNF: `while` (условие) оператор

Докато ‘условие’ е изпълнено, повтори ‘оператор’

Съществен специален случай:
операторът не се обработва

DO-WHILE-ОПЕРАТОР

1 Последователен ли е синтаксисът?

EBNF: `do оператор while ( условие ) ;`

Докато ‘условие' е изпълнено, повтори ‘оператор', при което условието се тества след изпълнение на оператора

Приложение:
Операторът се обработва поне веднъж

DO-WHILE-ОПЕРАТОР: ПРЕДСТАВЕН ЧРЕЗ WHILE-ОПЕРАТОР

1 Възможно ли е?

```
do
  оператор
while ( условие ) ;
```

Еквивалентен на:

```
оператор
while ( условие )
    оператор
```

РОЛЯ НА ';': РАЗДЕЛЯНЕ ИЛИ КРАЙ НА ОПЕРАТОРИ ?

Pascal, Modula-2, Ada, ... :

Разделяне на оператори

```
x := y ;
if x > y then
   begin x := y ; y := 0 end ;
d := 100
```

РОЛЯТА НА ';'

C, C++, **Java**:

- Край на оператори
  (синтактически – част от операторите)
- но: с много изключения
  (без ';' : while, if, ...)

→ “нечиста” езикова дефиниция (неединен принцип: причина за грешки)

```
x = y;
if (x > y) {
    x = y; y = 0;
}
d = 100;
```

ТЕСТ: ФУНКЦИЯТА “!”

n! = 1 * 2 * . . . * n

```
int n, x = 1,  fac = 1;
   ... // read n
while (x <= n) {
   fac = fac * x;
   x = x + 1;
}
```

напр. за n = 4:

- x получава стойностите: 1, 2, 3, 4, 5
  (при x = 5 не важи повече: x <= n)
- При това fac получава стойностите: 1, 1, 2, 6, 24

1 ½-ЦИКЪЛ-ПРОБЛЕМ

Break-оператор:
изход от цикъл в неговото тяло

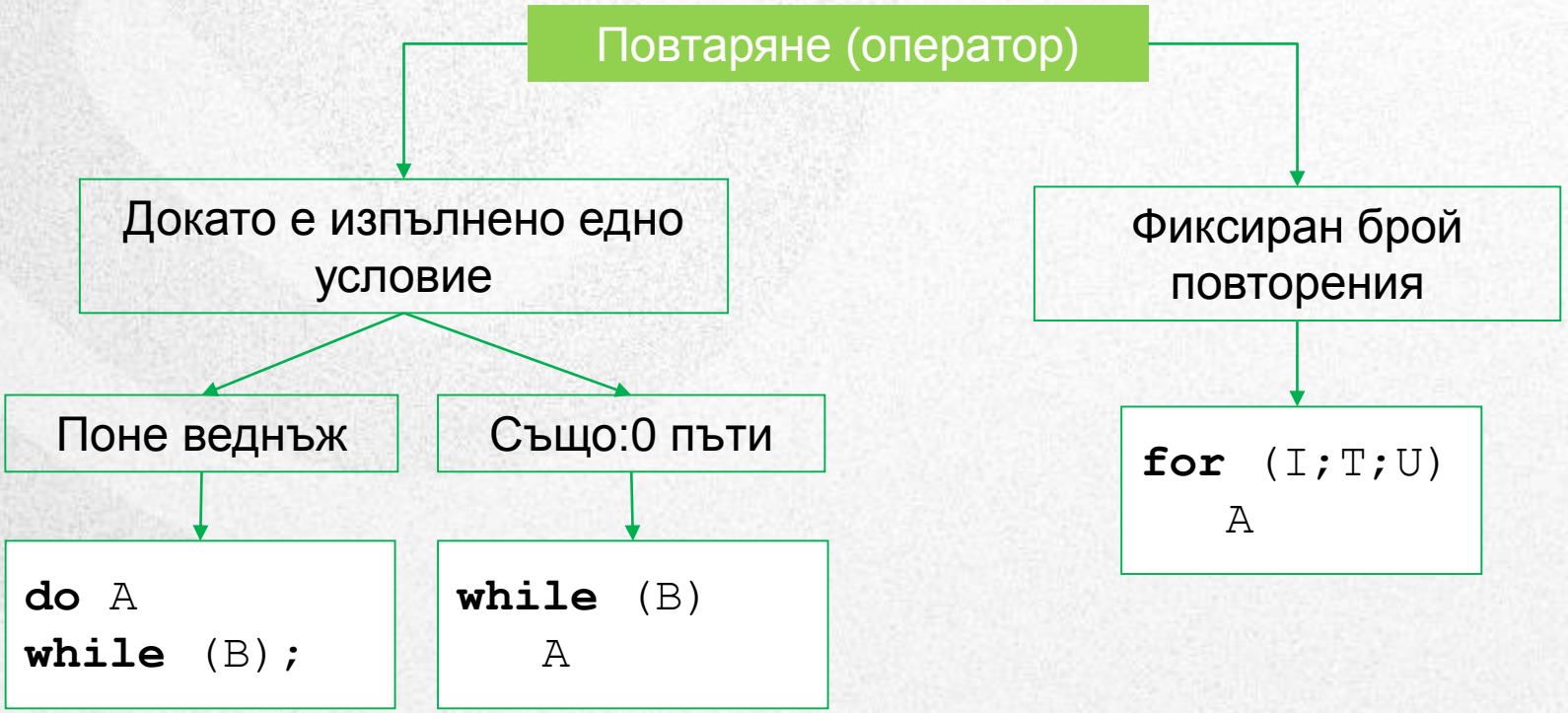
```
while (true) {
    ... четене ...
    if (Keyboard.eof())
        break;
    ... обработка входа ...
}
```

BREAK-ОПЕРАТОР: " 1 ½ - ЦИКЪЛ-ПРОБЛЕМ"

➔ Също без break:

```
... четене ...
while (! Keyboard.eof())
    ... Обработка входа ...
    ... четене ...
}
```

ИЗБОР НА ПОВТАРЯЕМИ ОПЕРАТОРИ



FOR-ЦИКЪЛ

EBNF: `for (инициализиране ; Test ; Update)оператор`

Еквивалентно на:

```
инициализиране ;
while ( Test )    {
    оператор ;
    Update ;
}
```

ПРИМЕР: ! С 'FOR'

Кратка форма: x++
Аналог на: x-- като x = x - 1

```
int n, x, fac = 1;
   ... // read n
for (x = 1; x <= n ; x = x + 1)
   fac = fac * x;
```

Приложение: броят на повторенията е познат (тук: n) !

ТЕСТ: ! С 'FOR'

1 Възможно ли е?

```
int n, x, fac = 1;
   ... // read n
for (x = 1; x <= n ; x = x + 1)
   fac = fac * x;
```

```
int n, fac = 1;
   ... // read n
for (int x = 1; x <= n ; x = x + 1)
   fac = fac * x;
```

В Java можем да разпръснем декларирането на променливите в рамките на целия блок, като ги дефинираме в момента, в който ни потрябват

ТЕСТ: ФУНКЦИЯТА “!”

1 Какъв е резултатът?

```
public static int faculty (int n) {
   int x, fac=1;

    for (x=1; x<=n; x++)
      fac = fac * x;

    return fac;
 }

 public static void main (String[] args) {

   int x, y;

   x = faculty(3);
   System.out.println("3! = " + x);
   y = faculty(5);
   System.out.println("5! = " + y);
  }
}
```

3! = 6
5! = 120

ПРИМЕР ЗА ЦИКЪЛ

Таблица: Трансформиране Celsius → Fahrenheit

Grad C	Grad F
-10.0	14.0
-9.0	15.8
-8.0	17.6
...	
10.0	50.0

→ Примерни програми с while и for оператор

С WHILE-ОПЕРАТОР

```
class TemperatureTable {
 // Table with C/F temperatures
   public static void main (String[] args) {
        final double
           LOW_TEMP = -10.0,
           HIGH_TEMP = 10.0;

        double
           cent, // degree Celsius
           fahr; // degree Fahrenheit

        System.out.println("\tGrad C\t\tGrad F");
        cent = LOW_TEMP;
        while (cent <= HIGH_TEMP) {
           fahr = (9.0/5.0) * cent + 32.0; // C -> F
           System.out.println("\t" + cent + "\t\t" + fahr);
           cent = cent + 1.0;
        }
   }
}
```

С FOR-ОПЕРАТОР

```
class TemperatureTable {
 // Table with C/F temperatures
   public static void main (String[] args) {
        final double
           LOW_TEMP = -10.0,
           HIGH_TEMP = 10.0;

        double
           cent, // Grad Celsius
           fahr; // Grad Fahrenheit

        System.out.println("\tGrad C\t\tGrad F");
        for(cent = LOW_TEMP; cent <=HIGH_TEMP; cent++) {
           fahr = (9.0/5.0) * cent + 32.0; // C -> F
           System.out.println("\t" + cent + "\t\t" + fahr);
        }
   }
}
```

ТЕСТ: FOR-ОПЕРАТОР

1 Коментар?

```
public class CommaOperator {
   public static void main(String[] args) {
      for( int i=1, j=i+10; i<5; i++, j=i*2) {
         System.out.println("i= " + i + " j= " + j);
      }
   }
}
```

Оператор „ , “ (последователност): има само едно приложение в Java – в управляващия израз на **for**

В управляващия блок на цикъла:

- Допуска се дефиниране на повече от една променлива, но те трябва да бъдат от един и същ тип
- Променливата се дефинира там, където се използва
- Областта на видимост – израз, управляващ **for**

ТЕСТ: FOR-ОПЕРАТОР

1 Какъв е резултатът?

```
public class CommaOperator {
   public static void main(String[] args) {
      for( int i=1, j=i+10; i<5; i++, j=i*2) {
         System.out.println("i= " + i + " j= " + j);
      }
   }
}
```

i= 1 j= 11
i= 2 j= 4
i= 3 j= 6
i= 4 j= 8

BREAK & CONTINUE

- В тялото на всяка една от управляващите конструкции можем да управляваме хода на цикъла посредством операторите break и continue
  - break – излиза от цикъла без да се изпълняват останалите конструкции в него
  - continue – спира изпълнението на текущата итерация и се връща в началото на цикъла, за да започне изпълнение на следващата итерация

ПРИМЕР

1. Коментар?
2. Какъв е резултатът?

```
public class BreeakAndContinue {
   public static void main(String[] args) {
      for(int i = 0; i < 100; i++) {
         if(i == 74) break; // Излиза от цикъла for
         if(i % 9 != 0) continue; // Следваща итерация
         System.out.println(i);
       }
       int i = 0;
       // "безкраен цикъл":
       while(true) {
          i++;
          int j = i * 27;
          if(j == 1269) break; // Излиза от цикъла
          if(i % 10 != 0) continue; // Връщане в началото на цикъла
          System.out.println(i);
        }
   }
}
```

```
0
9
18
27
36
45
54
63
72
```

```
10
20
30
40
```

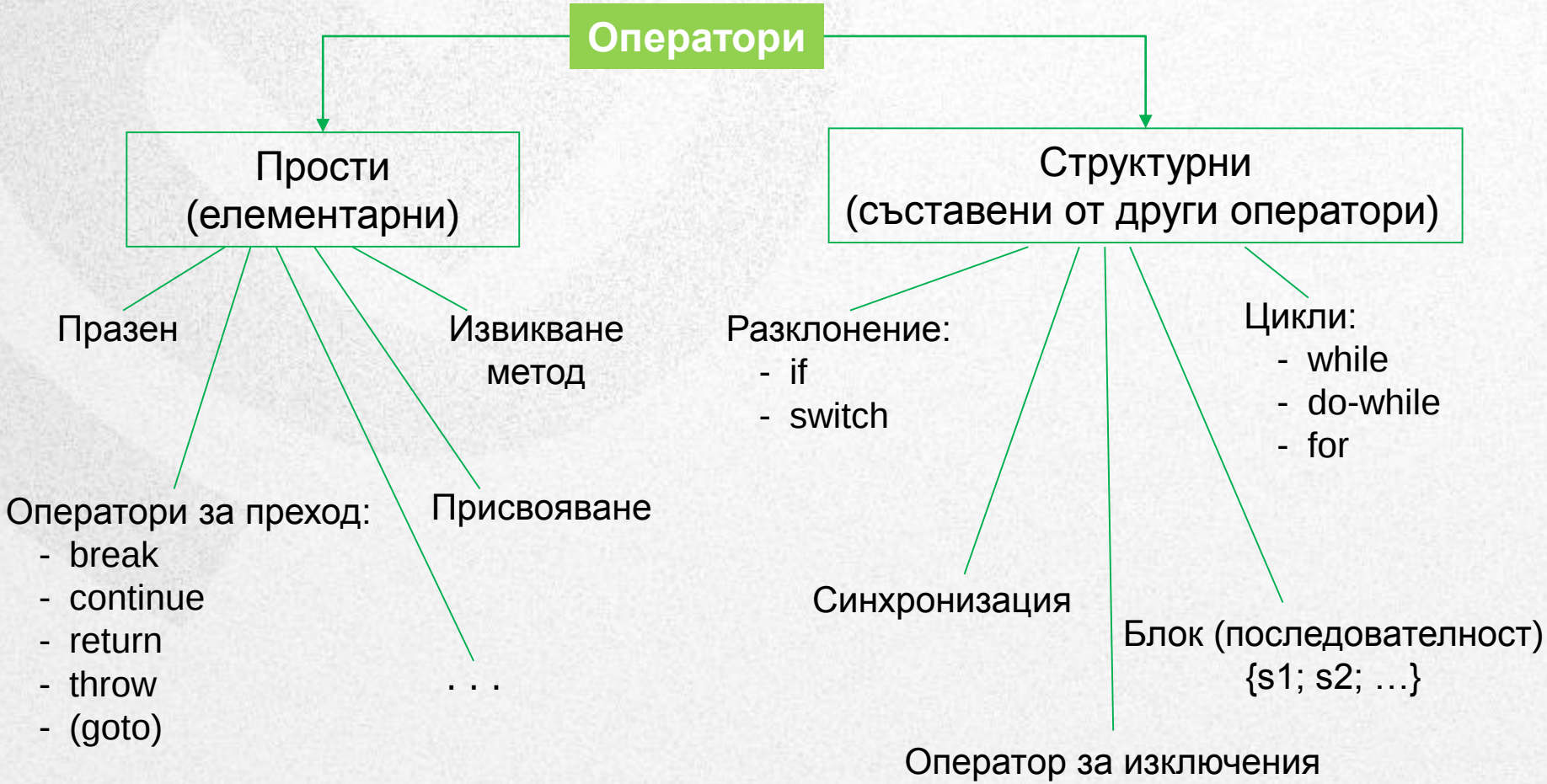
БЕЗКРАЙНИ ЦИКЛИ

1. Как се представят безкрайни цикли?

while(true)

for(;;)

СТРУКТУРНО ПРОГРАМИРАНЕ: ОПЕРАТОРИТЕ



БЛАГОДАРЯ ЗА ВНИМАНИЕТО!

КРАЙ “ИТЕРАЦИИ”