



ОНЛАЙН-ОБРАЗОВАНИЕ

# 16 – Using Loop Constructs (Часть 2)

**Дмитрий Коган**



## Как меня слышно и видно?



Если нет – напишите, если слышите – смайлик в чат.



## Цели :

- **Рассмотрим вложенные циклы**
- **Разберём способы выхода из круговорота**
- **Поймём, где и как использовать метки**





**Начинаем?**

# Темы экзамена

- ☐ Java Basics
- ☐ Working with Java Data Types
- ☐ Using Operators and Decision Constructs
- ☐ Creating and Using Arrays
- ☐ **Using Loop Constructs**
- ☐ Working with Methods and Encapsulation
- ☐ Working with Inheritance
- ☐ Handling Exceptions
- ☐ Working with Selected classes from the Java API

# Подтемы экзамена

## Using Loop Constructs

- Create and use while loops
- Create and use for loops including the enhanced for loop
- Create and use do/while loops
- Compare loop constructs
- Use break and continue



## Пролог



# Упражнение

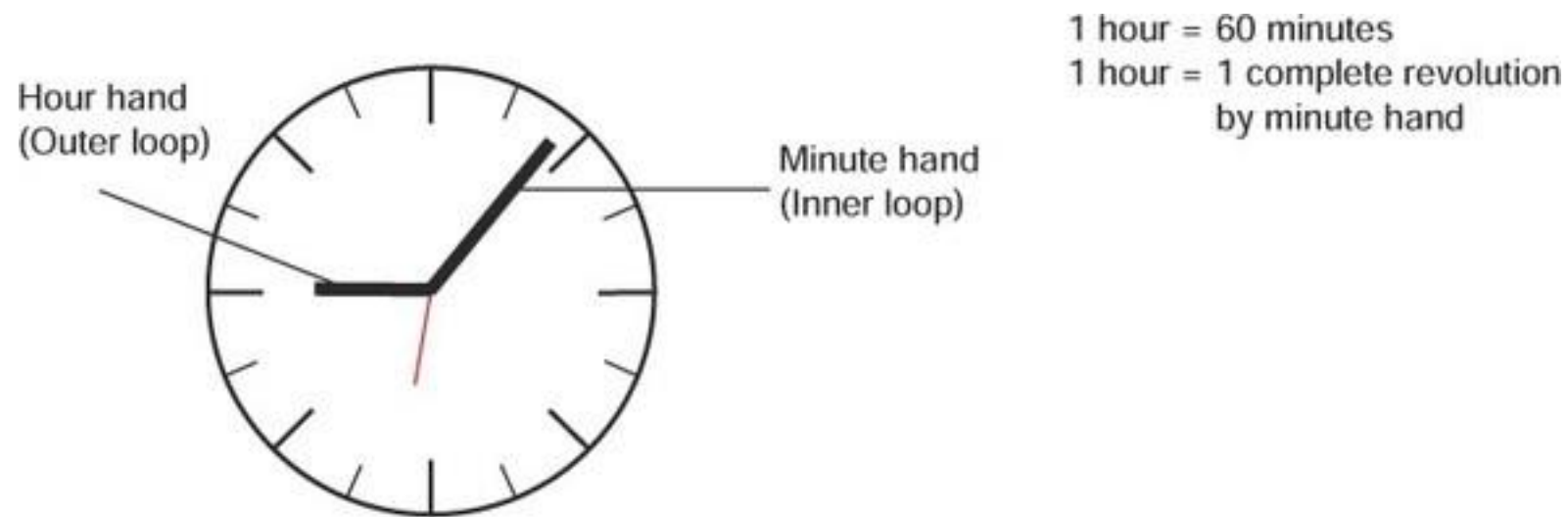
```
private static void run() {  
    System.out.print("Сайт курса:");  
    http://otus.ru  
    return;  
}
```

- A. Напечатает "Сайт курса:"
- B. Напечатает "Сайт курса:http://otus.ru"
- C. Зависнет
- D. Compile error



## Вложенные циклы

# Часы



```
for (int hrs = 1; hrs <= 6; hrs++) {  
    for (int min = 1; min <= 60; min++) {  
        System.out.println(hrs + ":" + min);  
    }  
}
```

Outer loop iterates for  
values 1 through 6

Inner loop iterates for  
values 1 through 60

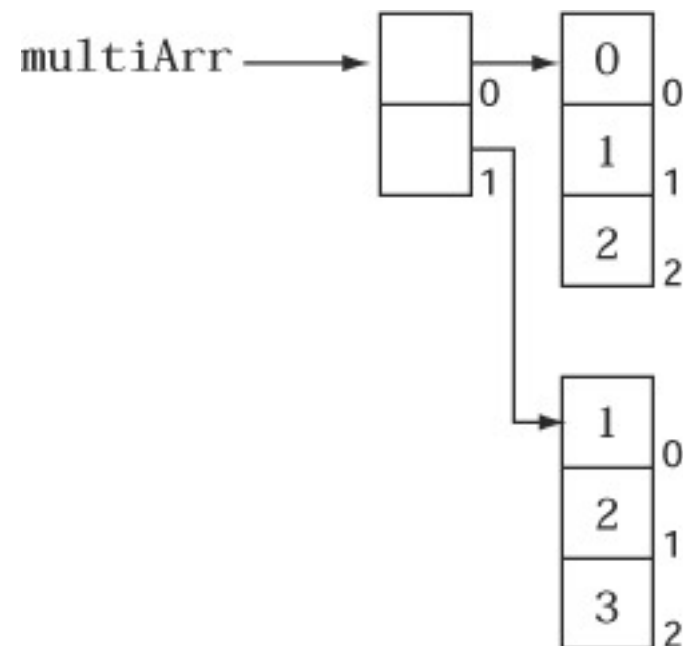
Executes  $6 \times 60$  times (total outer loop  
iterations  $\times$  total inner loop iterations)

# Обход массива

```
int multiArr[] [];  
multiArr = new int[2][3];  
for (int i = 0; i < multiArr.length; i++) {  
    for (int j = 0; j < multiArr[i].length; j++) {  
        multiArr[i][j] = i + j;  
    }  
}
```

Annotations for the code above:

- ① Array declaration (points to `int multiArr[] [];`)
- ② Array allocation (points to `multiArr = new int[2][3];`)
- ③ Outer for loop (points to the opening brace of the first `for` loop)
- ④ Inner for loop (points to the opening brace of the second `for` loop)
- Inner for loop ends (points to the closing brace of the second `for` loop)
- Outer for loop ends (points to the closing brace of the first `for` loop)

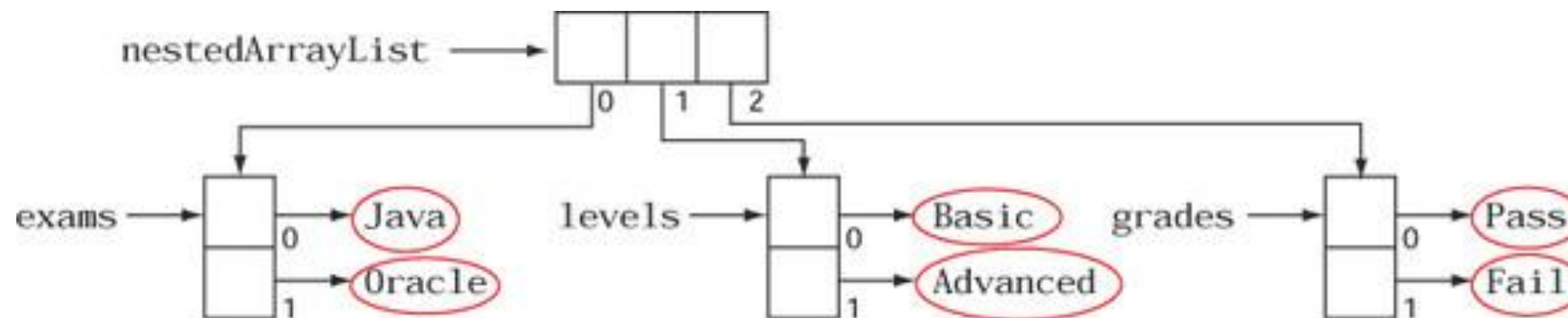


# Обход списка

```
ArrayList<ArrayList<String>> nestedArrayList =  
    new ArrayList< ArrayList<String>>();  
nestedArrayList.add(exams);  
nestedArrayList.add(levels);  
nestedArrayList.add(grades);
```

ArrayList of ArrayList  
←

Add object of ArrayList to nestedArrayList



```
for (ArrayList<String> nestedListElement : nestedArrayList)  
    for (String element : nestedListElement)  
        System.out.println(element);
```

Java  
Oracle  
Basic  
Advanced  
Pass  
Fail

# Вложенный цикл с for

```
int[][] myComplexArray = {{5,2,1,3},{3,9,8,9},{5,7,12,7}};  
  
for(int[] mySimpleArray : myComplexArray) {  
    for(int i=0; i<mySimpleArray.length; i++) {  
        System.out.print(mySimpleArray[i]+"\\t");  
    }  
    System.out.println();  
}
```

5	2	1	3
3	9	8	9
5	7	12	7

# Вложенный цикл с while

```
int hungryHippopotamus = 8;
while(hungryHippopotamus>0) {
    do {
        hungryHippopotamus -= 2;
    } while (hungryHippopotamus>5);
    hungryHippopotamus--;
    System.out.print(hungryHippopotamus+" ", " ");
}
```

3, 0,

# Для надёжности

```
int s = 5;  
for(int i = 0; i < 3; i++){  
    s += i;  
    System.out.print(s);  
}
```

```
i 0  
s 5  
p 5 (== "печатает 5")  
i 1  
s 6  
p 6  
i 2  
s 8  
p 8
```

лучше,  
чем

```
i 0 1 2  
s 5 6 8  
p 5 6 8
```



# Вложенные циклы

- ❑ Многоуровневые вложенные циклы могут выглядеть устрашающе, но есть все шансы, что до исполнения кое-каких уровней дело вообще не дойдет.

**Памятка:** Чем «мудреней» цикл, тем больше вероятность, что задача содержит ловушку где-то в другом месте, причем по какому-то довольно простому аспекту.

# Упражнение

```
class LoopyDoopy{
    public static void main(String[] args) {
        int i = 0, j = 4;
//      line X
        for (i = 0; i < x; i++) {
            do {
                int k = 0;
                while (k < z) {
                    System.out.print(k + " ");
                    k++;
                }
                System.out.println("");
                j--;
            } while (y >= j);
            System.out.println("-----");
        }
    }
}
```

**Which statement, when inserted at line X, will make the code print the following table?**

```
0 1 2 3 4
-----
0 1 2 3 4
-----
0 1 2 3 4
-----
```

- A. `int x = 2, y = 3, z = 2;`
- B. `int x = 3, y = 2, z = 4;`
- C. `int x = 3, y = 0, z = 5;`
- D. `int x = 4, y = 3, z = 4;`



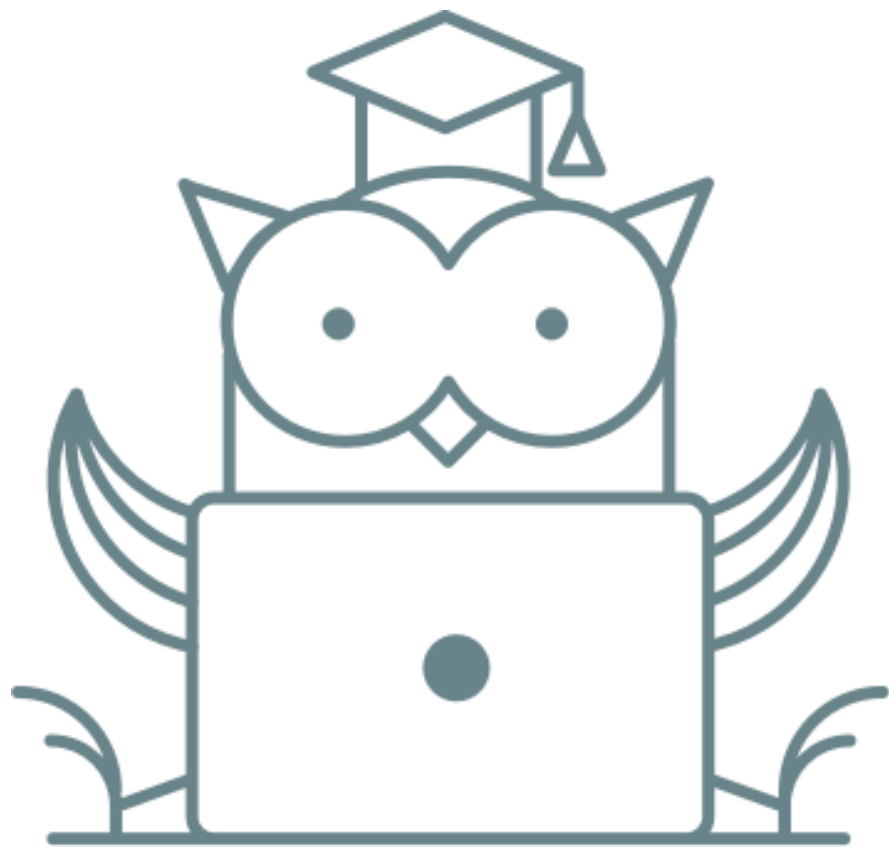
**Ответ: С**

# Правила пользования

- Вложенные циклы – Правила пользования:
  - чтобы выйти из внутреннего цикла целиком, используем просто `break`;
  - чтобы выскочить из текущей итерации, используем просто `continue`;
  - из внешнего цикла выходим через помеченный `break`;
  - из текущей итерации внешнего блока выходим через помеченный `continue`.



**Вопросы?**



**Метки**

# Добавляем метки

```
int[][] myComplexArray = {{5,2,1,3},{3,9,8,9},{5,7,12,7}};

OUTER_LOOP: for(int[] mySimpleArray : myComplexArray) {
    INNER_LOOP: for(int i=0; i<mySimpleArray.length; i++) {
        System.out.print(mySimpleArray[i]+"\\t");
    }
    System.out.println();
}
```

# Область применения

## □ Метки:

- можно добавлять к:
  - подблокам;
  - любым циклам `for`, `for-each`, `while` или `do-while`;
  - условным конструкциям типа `if` и `switch`;
  - выражениям и присвоениям;
  - операторам `return`;
  - TCF-конструкциям, и
  - операторам `throw`.
- нельзя добавлять к декларациям переменных.

```
int frog = 15;
BAD_IDEA: if(frog>10)
EVEN_WORSE_IDEA: {
    frog++;
}
```



# Примеры использования

Хотя помеченный `break` используется в первую очередь с циклами, его можно применять и в ряде других случаев (скажем, в `if`-конструкциях). При этом надо внимательно следить за областью действия этого оператора, иначе может возникнуть комперр `'unreachable statement'`.

Помеченный `break` нуждается в блоке или `if`-конструкции, где содержится как минимум одна операция `break label;`, например:

```
class Label {
    String label(){
//        label int a = 10;                // INVALID
        int a = 10;
//        label: if (true) break;          // INVALID
        label: if (true) break label;
        label: for (;true;) break label;
        label: switch(a) {}
        label: a = 42;
        label: a++;
        label: try{ label2: throw new NullPointerException(); }
                catch(NullPointerException npe){ break label; }
        label: return "Label!";
    }
}
```

# Возможные ошибки

Метки предназначены в первую очередь для блоков или циклов, и вне этих конструкций их область действия может оказаться недействительной → комперр.

**Следствие:** Увидев помеченный `continue`, надо убедиться, что он и впрямь стоит внутри того цикла, к которому привязана данная метка.

```
label: if (true)
    for (int j = 0; j < 10; j++)
    {
        continue label; // continue не может использовать метку от if
    }
```

Когда `if` не стоит внутри цикла или `switch`-блока, из него нельзя выскочить через `break` без метки:

```
// if(true) break; // INVALID
for(;;true;) if(true) break;
```

# УЧИМСЯ ИСПОЛЬЗОВАТЬ

```
L1: if (i > 0) {  
    L1: System.out.println(i);    // (1) Not OK. Label redeclared.  
}  
  
L1: while (i < 0) {                // (2) OK.  
    L2: System.out.println(i);  
}  
  
L1: {                             // (3) OK. Labeled block.  
    int j = 10;  
    System.out.println(j);  
}  
  
L1: try {                         // (4) OK. Labeled try-catch-finally block.  
    int j = 10, k = 0;  
    L2: System.out.println(j/k);  
} catch (ArithmeticException ae) {  
    L3: ae.printStackTrace();  
} finally {  
    L4: System.out.println("Finally done.");  
}  
  
LabelA: LabelB: System.out.println("Multiple labels. Use judiciously.");  
  
L0: int i = 0;                    // Compile-time error!
```

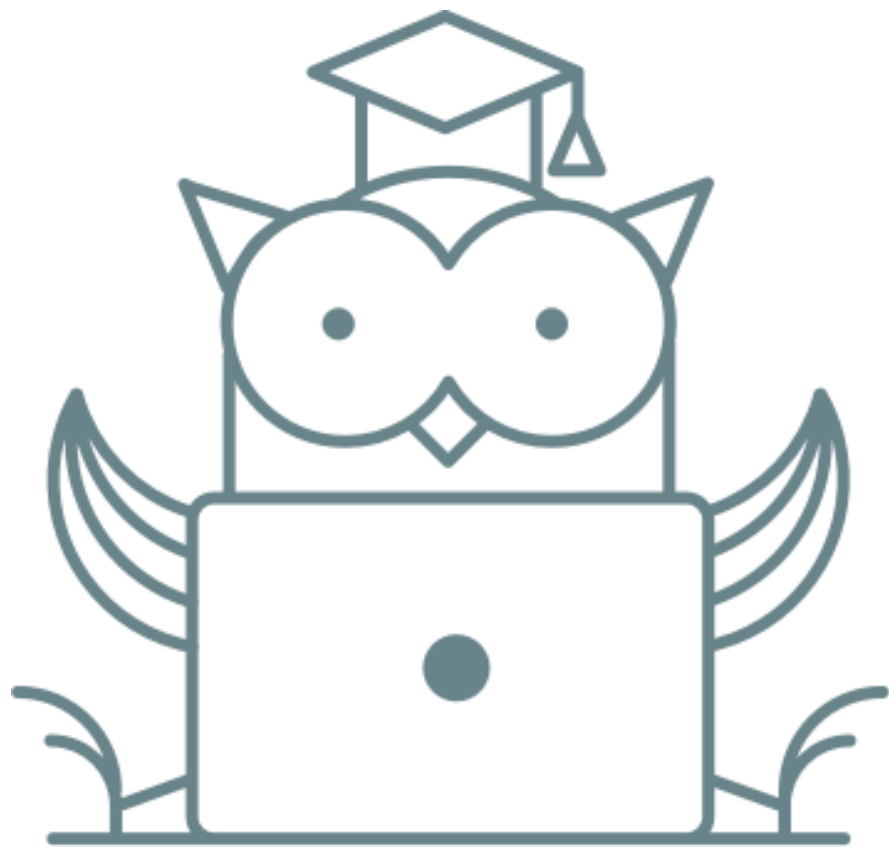
# Утилизация меток

Метки можно «утилизировать» многократно (потому что область их действия заканчивается вместе с тем блоком или циклом, с которым они связаны):

```
public static void main(String[] args){                // выдает 1234
    label: {
        System.out.print("1");
        if (false) break label; System.out.print("2");
    }
    label:
        if (1 > 0) {
            System.out.print("3");
            if (false) break label; System.out.print("4");
        }
}
```



**Вопросы?**



**Команда break**

# Трансферные команды

- `break`
- `continue`
- `return`
- `try-catch-finally`
- `throw`
- `assert`

# Структура break

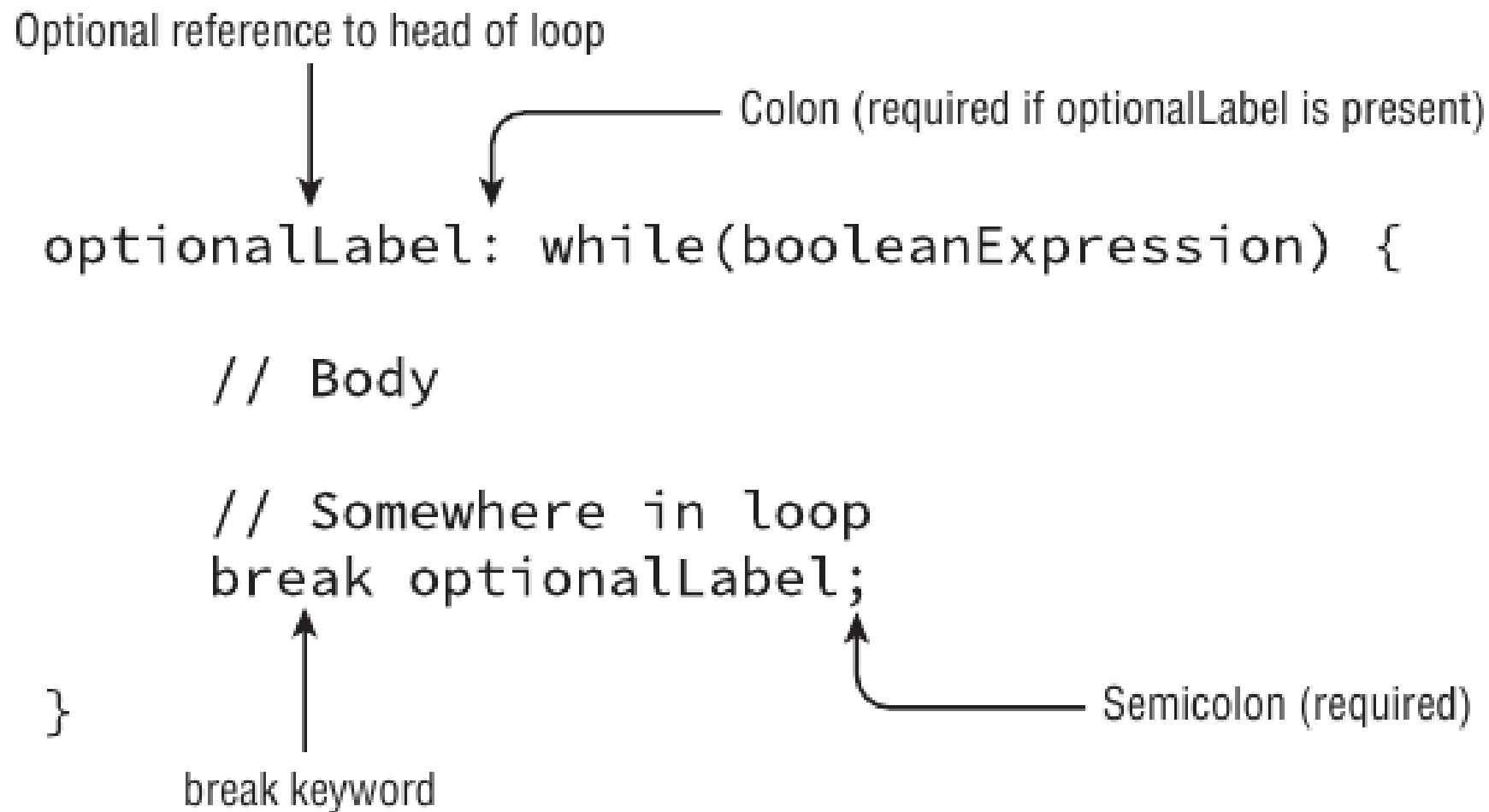
Optional reference to head of loop

Colon (required if optionalLabel is present)

```
optionalLabel: while(booleanExpression) {  
  
    // Body  
  
    // Somewhere in loop  
    break optionalLabel;  
}
```

break keyword

Semicolon (required)





# Принцип работы

- ❑ Оператор `break` заставляет выпрыгнуть из цикла «чисто», другими словами, прекращается не только текущая итерация, но не будут исполнены и все последующие. Контроль передается той операции, которая идет сразу после данного цикла.

# Как это работает

```
public class FindInMatrix {
    public static void main(String[] args) {
        int[][] list = {{1,13},{5,2},{2,2}};
        int searchValue = 2;
        int positionX = -1;
        int positionY = -1;

        PARENT_LOOP: for(int i=0; i<list.length; i++) {
            for(int j=0; j<list[i].length; j++) {
                if(list[i][j]==searchValue) {
                    positionX = i;
                    positionY = j;
                    break PARENT_LOOP;
                }
            }
        }

        if(positionX==-1 || positionY==-1) {
            System.out.println("Value "+searchValue+" not found");
        } else {
            System.out.println("Value "+searchValue+" found at: " +
                "("+positionX+","+positionY+")");
        }
    }
}
```

Value 2 found at: (1,1)

# Как это работает

```
public class FindInMatrix {
    public static void main(String[] args) {
        int[][] list = {{1,13},{5,2},{2,2}};
        int searchValue = 2;
        int positionX = -1;
        int positionY = -1;

        PARENT_LOOP: for(int i=0; i<list.length; i++) {
            for(int j=0; j<list[i].length; j++) {
                if(list[i][j]==searchValue) {
                    positionX = i;
                    positionY = j;
                    break;
                }
            }
        }

        if(positionX==-1 || positionY==-1) {
            System.out.println("Value "+searchValue+" not found");
        } else {
            System.out.println("Value "+searchValue+" found at: " +
                "("+positionX+","+positionY+")");
        }
    }
}
```

Value 2 found at: (2,0)

# Как это работает

```
public class FindInMatrix {
    public static void main(String[] args) {
        int[][] list = {{1,13},{5,2},{2,2}};
        int searchValue = 2;
        int positionX = -1;
        int positionY = -1;

        PARENT_LOOP: for(int i=0; i<list.length; i++) {
            for(int j=0; j<list[i].length; j++) {
                if(list[i][j]==searchValue) {
                    positionX = i;
                    positionY = j;
                }
            }
        }

        if(positionX== -1 || positionY== -1) {
            System.out.println("Value "+searchValue+" not found");
        } else {
            System.out.println("Value "+searchValue+" found at: " +
                "("+positionX+","+positionY+")");
        }
    }
}
```

Value 2 found at: (2,1)

# for n switch

```
class BreakOut {  
  
    public static void main(String[] args) {  
        System.out.println("i      sqrt(i)");  
        for (int i = 1; i <= 5; ++i) {  
            if (i == 4)  
                break;                                // (1) Terminate loop. Control to  
(2).  
            // Rest of loop body skipped when i gets the value 4.  
            System.out.printf("%d      %.2f%n", i, Math.sqrt(i));  
        } // end for  
        // (2) Continue here.  
        int n = 2;  
        switch (n) {  
            case 1:  
                System.out.println(n);  
                break;  
            case 2:  
                System.out.println("Inner for(;;) loop: ");  
                for (int j = 0; j <= n; j++) {  
                    if (j == 2)  
                        break;                                // (3) Terminate loop. Control to  
(4).  
                    System.out.println("j = " + j);  
                }  
            default:  
                System.out.println("default: n = " + n); // (4) Continue here.  
        }  
    }  
}
```

Output from the program:

```
i      sqrt(i)  
1      1.00  
2      1.41  
3      1.73  
Inner for(;;) loop:  
j = 0  
j = 1  
default: n = 2
```

# Помеченный break

```
out:                // Label.
{                  // (1) Labeled block.
    // ...
    if (j == 10) break out; // (2) Terminate block. Control to (3).
    System.out.println(j); // Rest of the block not executed if j == 10.
    // ...
}
// (3) Continue here.
```

# Помеченный break

```
class LabeledBreakOut {
    public static void main(String[] args) {
        int[][] squareMatrix = {{4, 3, 5}, {2, 1, 6}, {9, 7, 8}};
        int sum = 0;
        outer: for (int i = 0; i < squareMatrix.length; ++i){    // (1) label
            for (int j = 0; j < squareMatrix[i].length; ++j) {    // (2)
                if (j == i) break;    // (3) Terminate inner loop. Control to
(5) .
                System.out.println("Element[" + i + ", " + j + "]: " +
                                    squareMatrix[i][j]);
                sum += squareMatrix[i][j];
                if (sum > 10) break outer; // (4) Terminate both loops. Control to
(6) .
            } // end inner loop
            // (5) Continue with update expression in the outer loop header.
        } // end outer loop
        // (6) Continue here.
        System.out.println("sum: " + sum);
    }
}
```

Output from the program:

```
Element[1, 0]: 2
Element[2, 0]: 9
sum: 11
```

# Упражнение

Which of the following statements are valid when occurring on their own?

Select the three correct answers.

(a) `while () break;`

(b) `do { break; } while (true);`

(c) `if (true) { break; }`

(d) `switch (1) { default: break; }`

(e) `for (;true;) break;`





**Ответ: BDE**



**Вопросы?**



**Команда continue**

# Структура continue

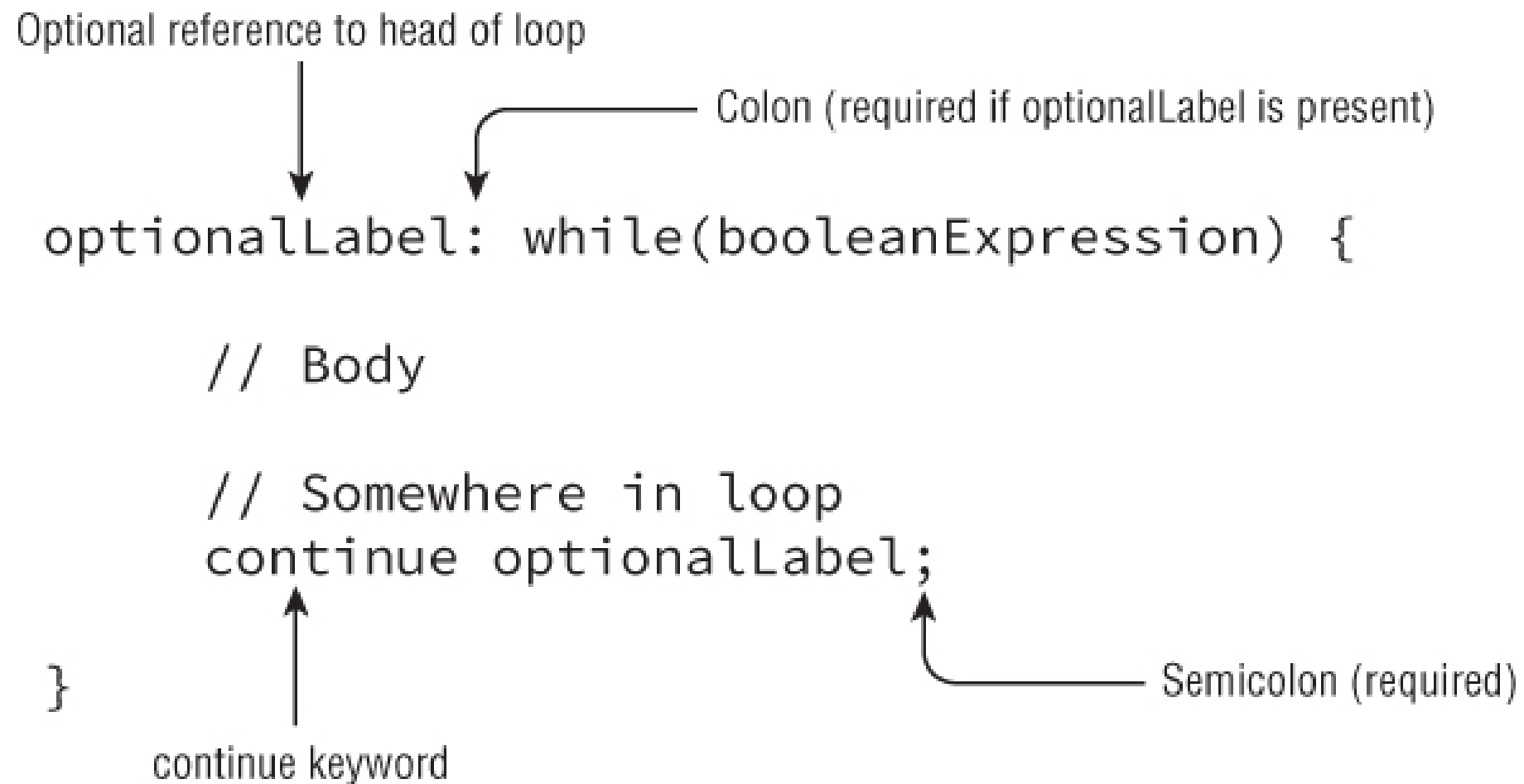
Optional reference to head of loop

Colon (required if optionalLabel is present)

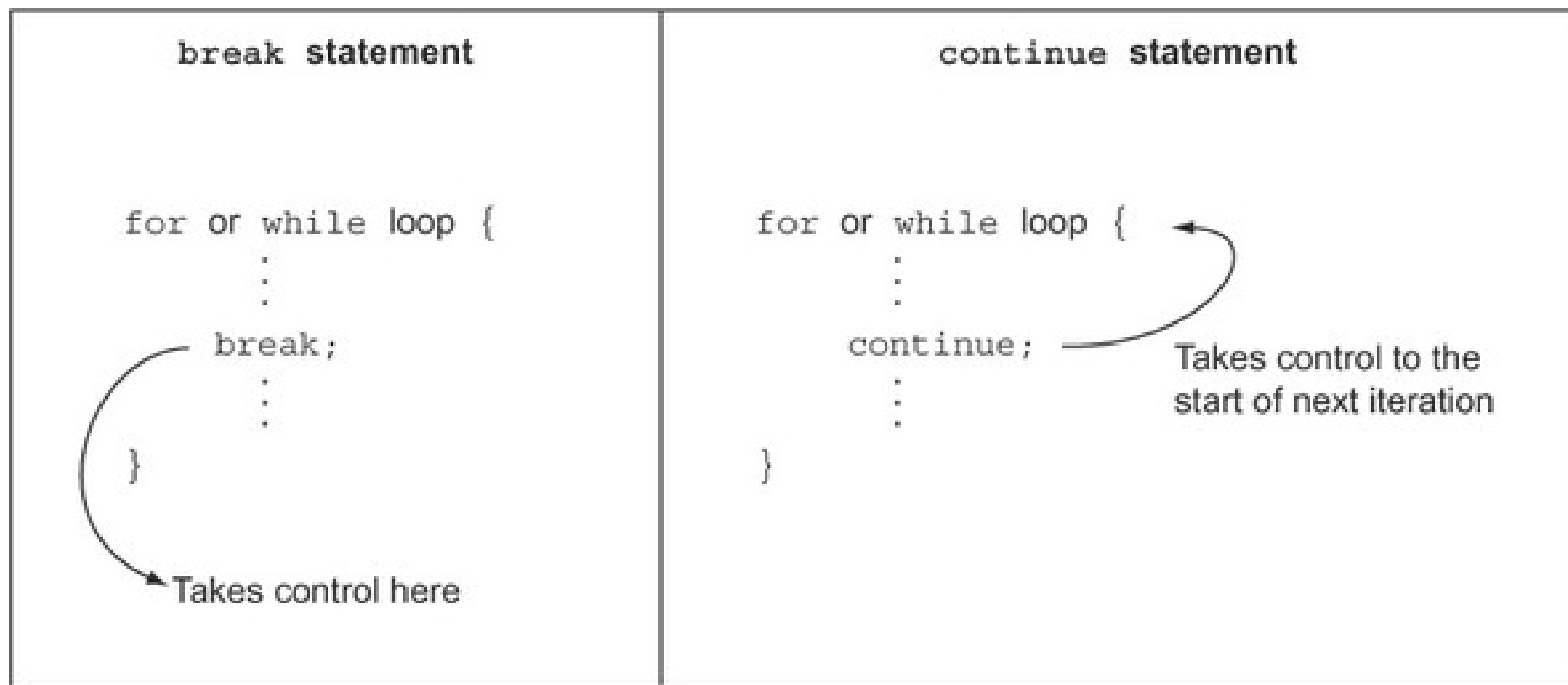
```
optionalLabel: while(booleanExpression) {  
  
    // Body  
  
    // Somewhere in loop  
    continue optionalLabel;  
}
```

continue keyword

Semicolon (required)



# Отличие от break



# Область применения

- ❑ `continue` действителен лишь внутри циклов; годится ЛЮБОЙ цикл, в т.ч. `for-each`, `for`, `do-while` и `while`.

# Как это работает

```
public class CleaningSchedule {  
    public static void main(String[] args) {  
        CLEANING: for(char stables = 'a'; stables<='d'; stables++) {  
            for(int leopard = 1; leopard<4; leopard++) {  
                if(stables=='b' || leopard==2) {  
                    continue CLEANING;  
                }  
                System.out.println("Cleaning: "+stables+", "+leopard);  
            }  
        }  
    }  
}
```

```
Cleaning: a,1  
Cleaning: c,1  
Cleaning: d,1
```

# Как это работает

```
public class CleaningSchedule {  
    public static void main(String[] args) {  
        CLEANING: for(char stables = 'a'; stables<='d'; stables++) {  
            for(int leopard = 1; leopard<4; leopard++) {  
                if(stables=='b' || leopard==2) {  
                    continue  
                }  
                System.out.println("Cleaning: "+stables+", "+leopard);  
            }  
        }  
    }  
}
```

```
Cleaning: a,1  
Cleaning: a,3  
Cleaning: c,1  
Cleaning: c,3  
Cleaning: d,1  
Cleaning: d,3
```



# Как это работает

```
public class CleaningSchedule {  
    public static void main(String[] args) {  
        CLEANING: for(char stables = 'a'; stables<='d'; stables++) {  
            for(int leopard = 1; leopard<4; leopard++) {  
  
                System.out.println("Cleaning: "+stables+", "+leopard);  
  
            } } } }  
}
```

```
Cleaning: a,1  
Cleaning: a,2  
Cleaning: a,3  
Cleaning: b,1  
Cleaning: b,2  
Cleaning: b,3  
Cleaning: c,1  
Cleaning: c,2  
Cleaning: c,3  
Cleaning: d,1  
Cleaning: d,2  
Cleaning: d,3
```

# continue без метки

```
class Skip {  
    public static void main(String[] args) {  
        System.out.println("i    sqrt(i)");  
        for (int i = 1; i <= 5; ++i) {  
            if (i == 4) continue;           // (1) Control to (2).  
            // Rest of loop body skipped when i has the value 4.  
            System.out.printf("%d    %.2f%n", i, Math.sqrt(i));  
            // (2) Continue with update expression in the loop header.  
        } // end for  
    }  
}
```

Output from the program:

i	sqrt(i)
1	1.00
2	1.41
3	1.73
5	2.24

# continue с меткой

```
class LabeledSkip {
    public static void main(String[] args) {
        int[][] squareMatrix = {{4, 3, 5}, {2, 1, 6}, {9, 7, 8}};
        int sum = 0;
        outer: for (int i = 0; i < squareMatrix.length; ++i){    // (1) label
            for (int j = 0; j < squareMatrix[i].length; ++j) {    // (2)
                if (j == i) continue;                            // (3) Control to
(5) .
                System.out.println("Element[" + i + ", " + j + "]: " +
                    squareMatrix[i][j]);
                sum += squareMatrix[i][j];
                if (sum > 10) continue outer;                    // (4) Control to
(6) .
                // (5) Continue with update expression in the inner loop header.
            } // end inner loop
            // (6) Continue with update expression in the outer loop header.
        } // end outer loop
        System.out.println("sum: " + sum);
    }
}
```

Output from the program:

```
Element[0, 1]: 3
Element[0, 2]: 5
Element[1, 0]: 2
Element[1, 2]: 6
Element[2, 0]: 9
sum: 25
```

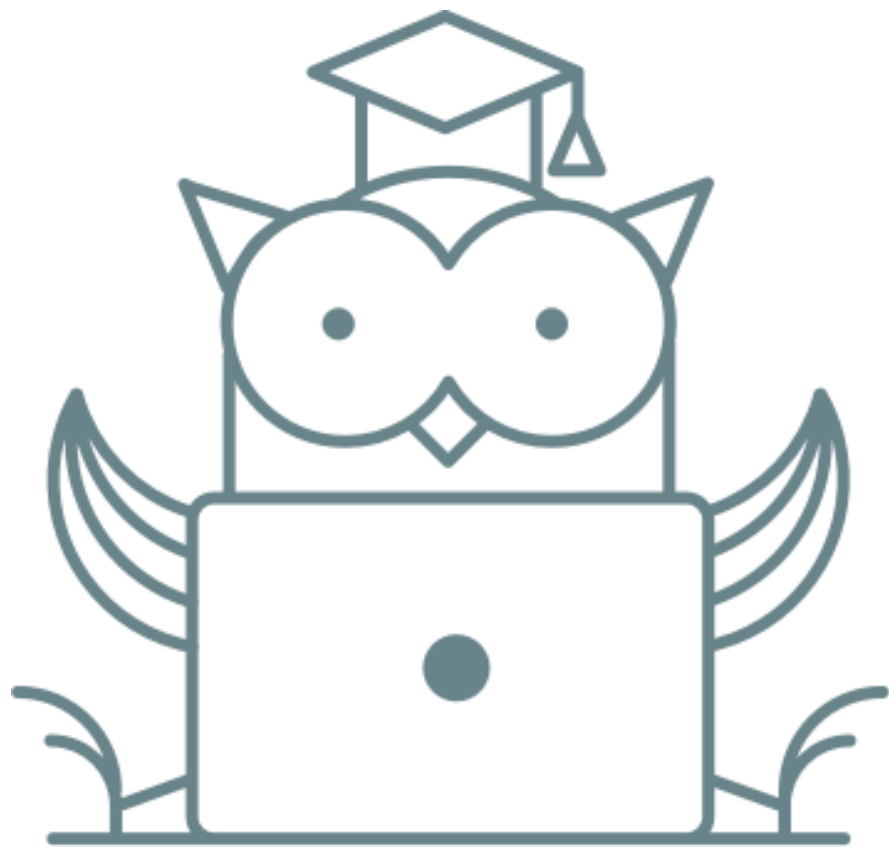
# Всё вместе

	Allows optional labels	Allows <i>break</i> statement	Allows <i>continue</i> statement
<code>while</code>	Yes	Yes	Yes
<code>do while</code>	Yes	Yes	Yes
<code>for</code>	Yes	Yes	Yes
<code>switch</code>	Yes	Yes	No

# Упражнение

```
private static void run() {  
    System.out.print("Сайт курса:");  
    http://otus.ru  
    return;  
}
```

- A. Напечатает "Сайт курса:"
- B. Напечатает "Сайт курса:http://otus.ru"
- C. Зависнет
- D. Compile error



**Ответ: A**



**Вопросы?**

---

**Домашнее задание**

**Тест**





**Пожалуйста, пройдите опрос**

**<https://otus.ru/polls/17822/>**



**Спасибо  
за внимание!**

**Только приятных  
повторений!**