OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

# 15 – Using Loop Constructs (Часть 1)

Дмитрий Коган

# Как меня слышно и видно?

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

# Цели :

- Изучим все виды циклов
- Сравним их между собой
- Поймём, когда какой использовать

# 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

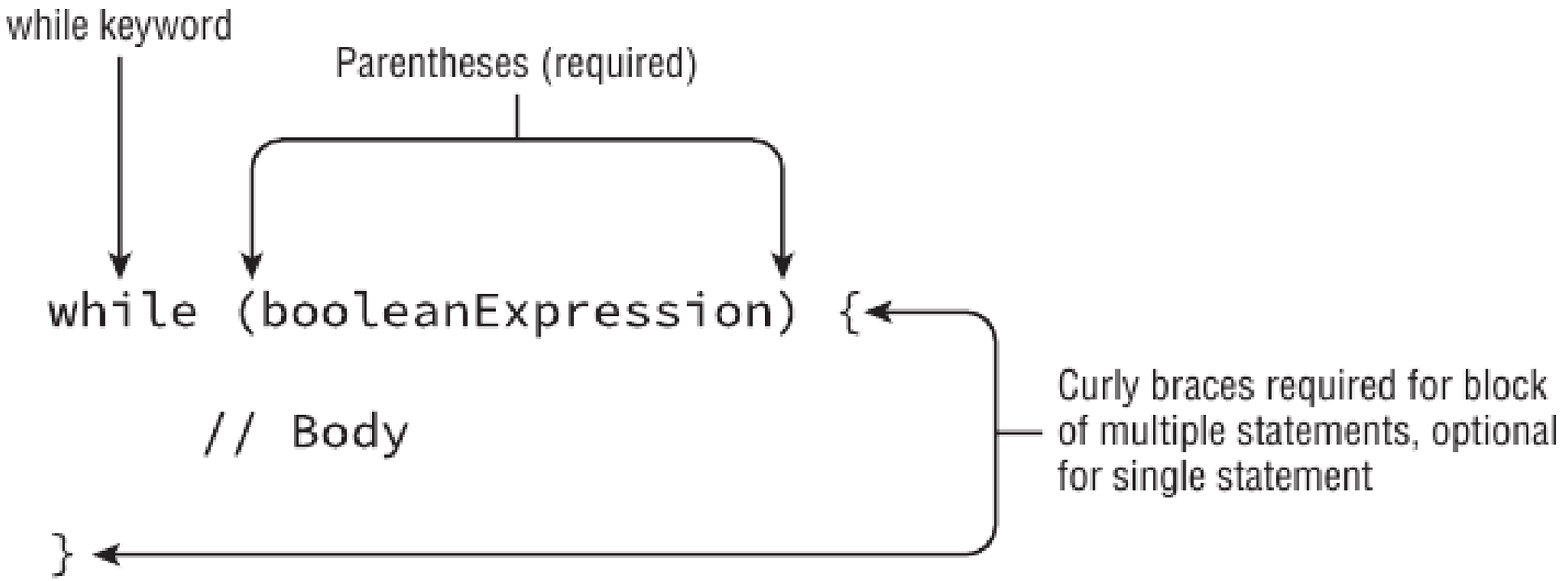
# Подтемы экзамена

**Using Loop Constructs**

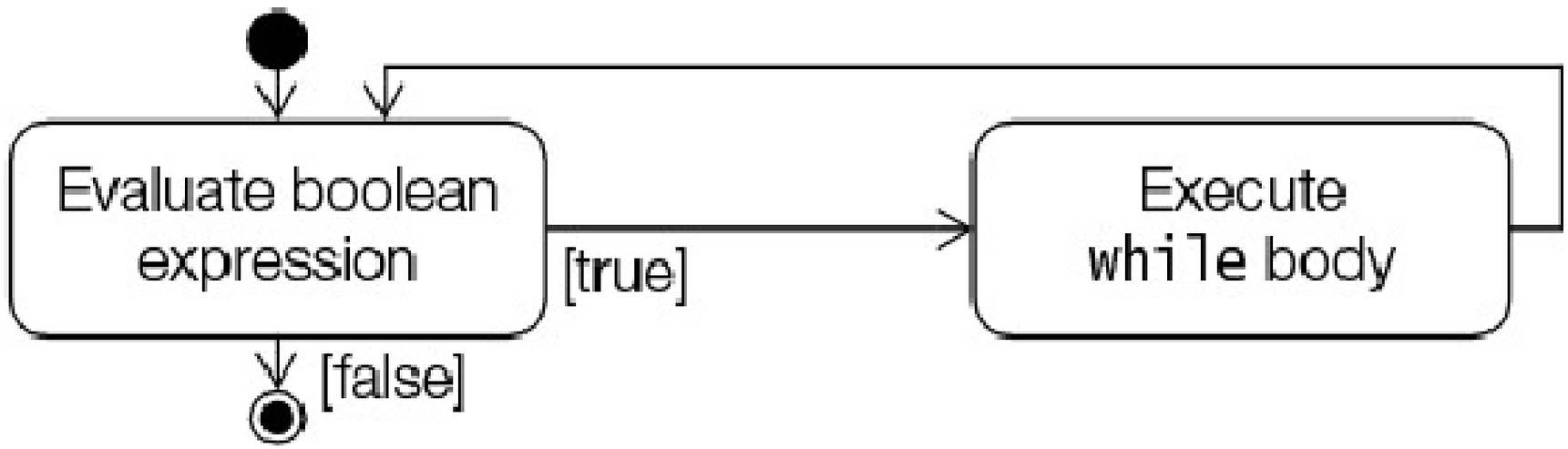
- **Create and use while loops**
- **Create and use for loops including the enhanced for loop**
- **Create and use do/while loops**
- **Compare loop constructs**
- **Use break and continue**

# Цикл while

# Структура while



# Диаграмма активности



# Пустое тело

```
while (noSignOfLife())
  keepLooking();
```

```
while (noSignOfLife());      // Empty statement as loop body!
  keepLooking();             // Statement not in the loop body.
```

# Присвоения

```
int a = 0;
while( (a = a + 1) > 0) {                    // VALID
    System.out.println("hello");
    break;
}
```

`while`-выражение не принимает деклараций переменных; зато в нем можно поставить вызов метода.

# Как работает while

```
int roomInBelly = 5;
public void eatCheese(int bitesOfCheese) {
    while (bitesOfCheese > 0 && roomInBelly > 0) {
        bitesOfCheese--;
        roomInBelly--;
    }
    System.out.println(bitesOfCheese+" pieces of cheese left");
}
```

# Количество итераций

```
int full = 5;
while(full < 5) {
    System.out.println("Not full!");
    full++;
}
```

# Упражнение

```
class Test {
    public static void main(String[] args) {
        boolean b1, b2;
        int a = 0;
        while (b1 = b2 = false) { }
        while (!!true) { break; }
        while (a == 0 ? false : true) { }
        while (new Test().equals("?!")) { }
    }
}
```

**How many LOCs fail compilation?**

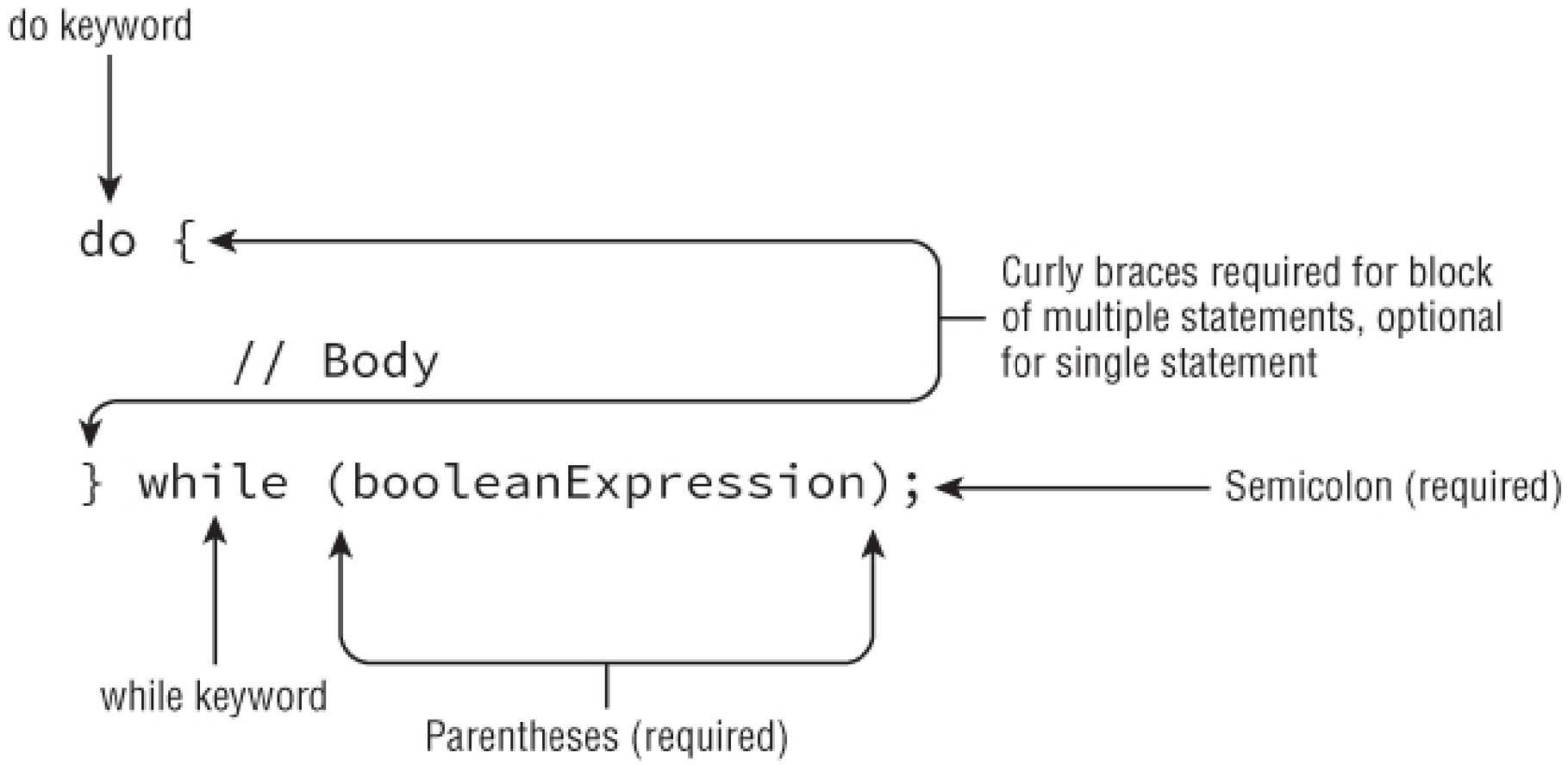
A. None
B. One
C. Two
D. Three
E. Four

**Ответ: А**

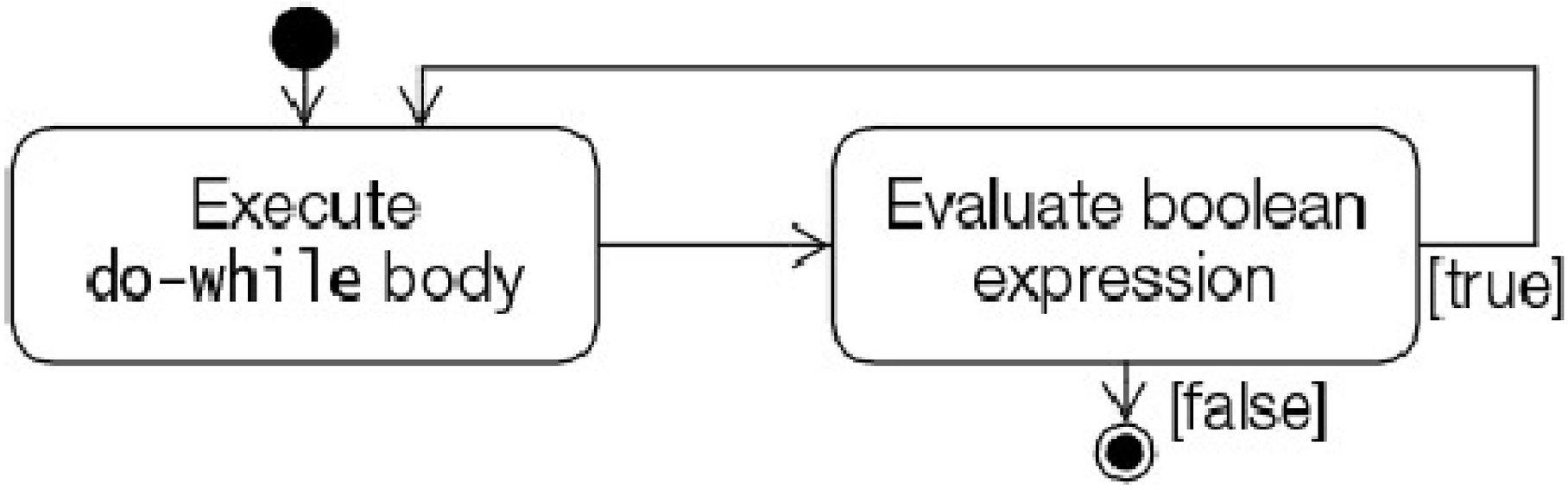
# Вопросы?

# Цикл do-while

# Структура do-while



# Диаграмма активности



# Как работает do-while

```
int lizard = 0;
do {
    lizard++;
} while(false);
System.out.println(lizard);  // 1
```

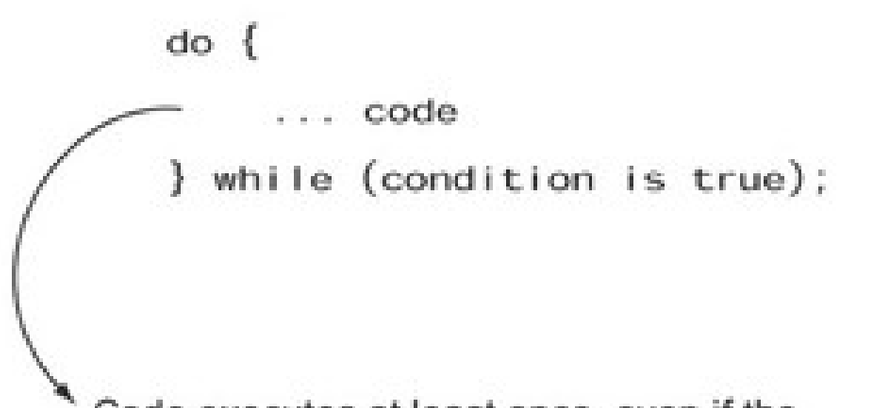
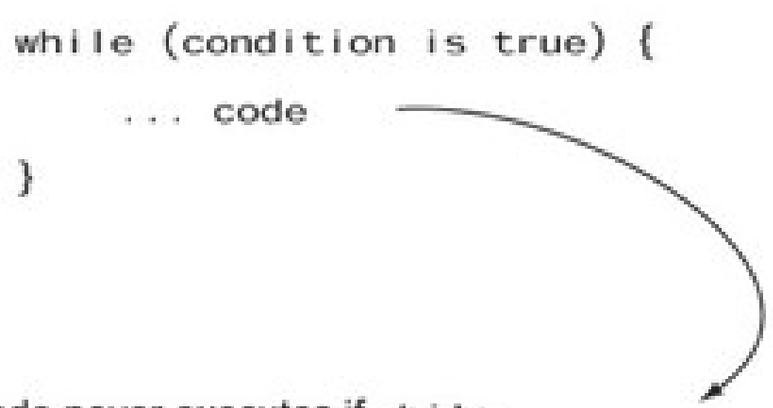
# Большая разница

```
while (cat.isAway()) {                    // (1)
  mice.play();
}
```

```
do {                                      // (2)
  mice.play();
} while (cat.isAway());
```

- `do-while` выполняется как минимум один раз.

# Сравнение

| do-while loop | while loop |
| --- | --- |
|  |  |
| Code executes at least once, even if the `while` condition initially evaluates to `false`. | Code never executes if `while` condition initially evaluates to `false`. |

# Конвертация

```
while(llama > 10) {
    System.out.println("Llama!");
    llama--;
}
```

```
if(llama > 10) {
    do {
        System.out.println("Llama!");
        llama--;
    } while(llama > 10);
}
```

# Область действия

```
do {
    boolean y = false;
    System.out.println(y);
} while (y); // Out of scope для y
```

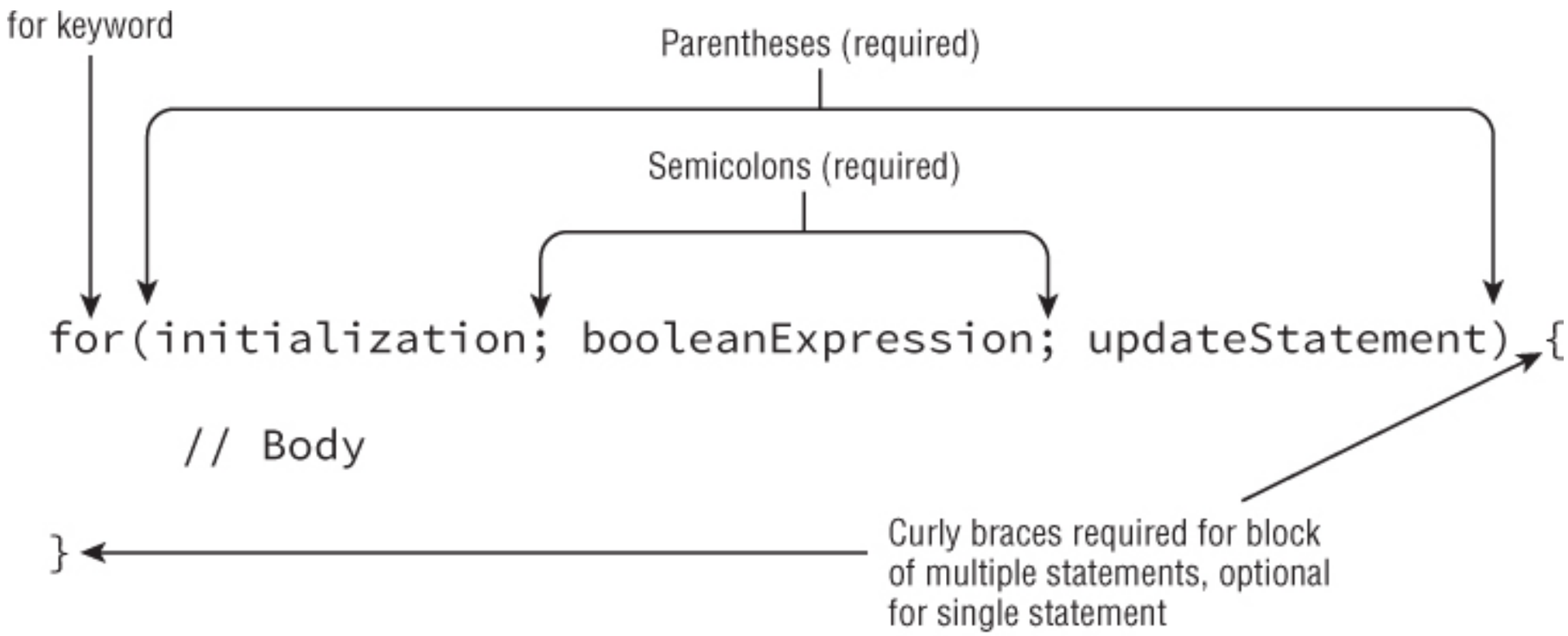
# Бесконечный цикл

```
int pen = 2;
int pigs = 5;
while(pen < 10)
    pigs++;
```

# Вопросы?

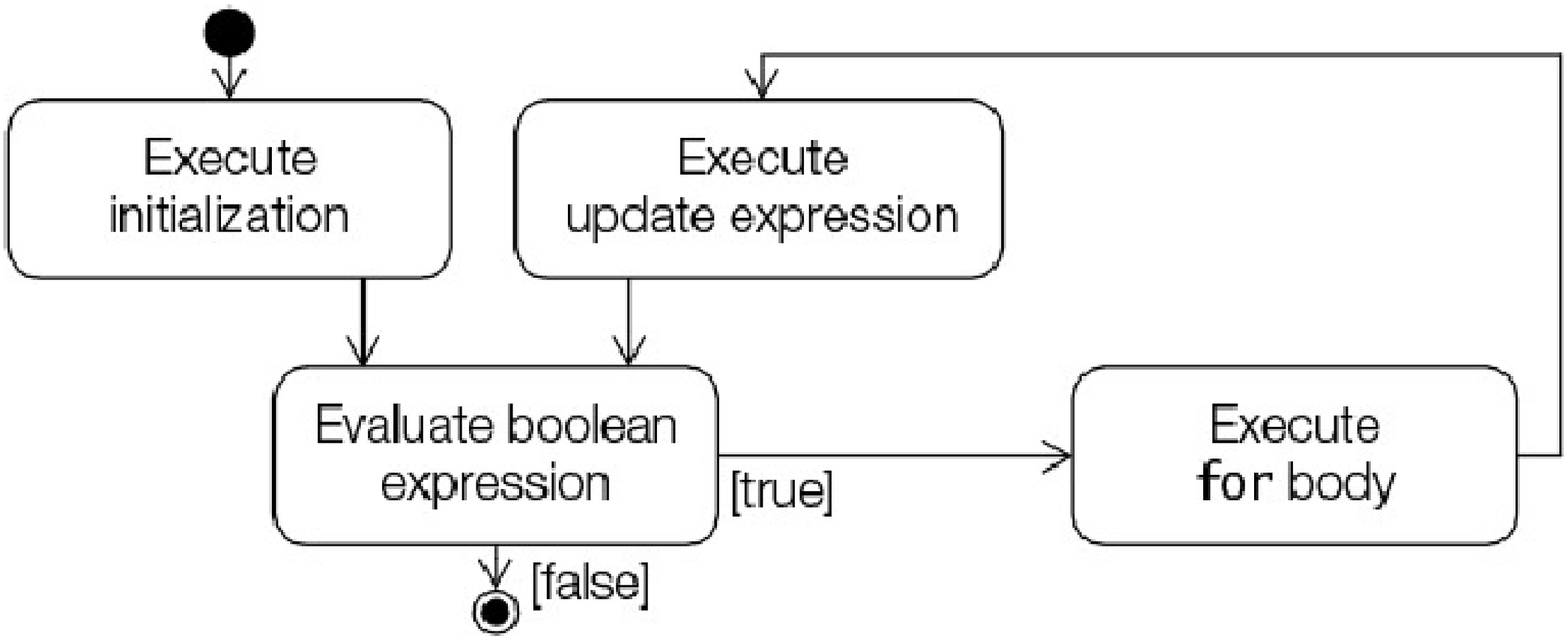
# Цикл for

# Структура for

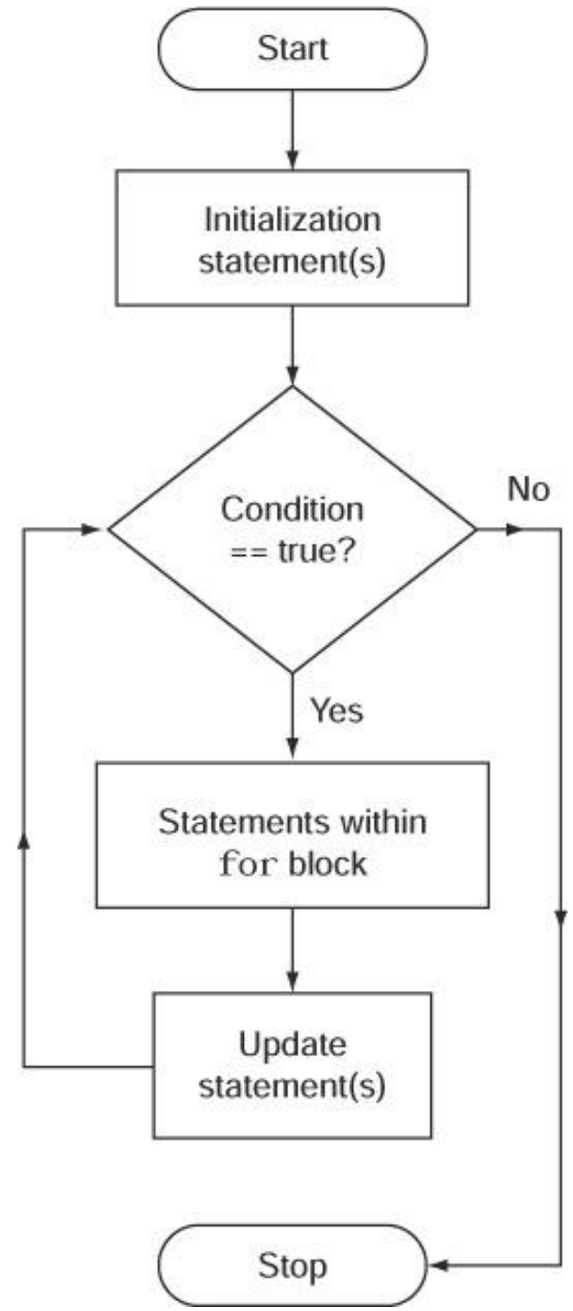



1. Initialization statement executes
2. If booleanExpression is true continue, else exit loop
3. Body executes
4. Execute updateStatements
5. Return to Step 2

# Диаграмма активности

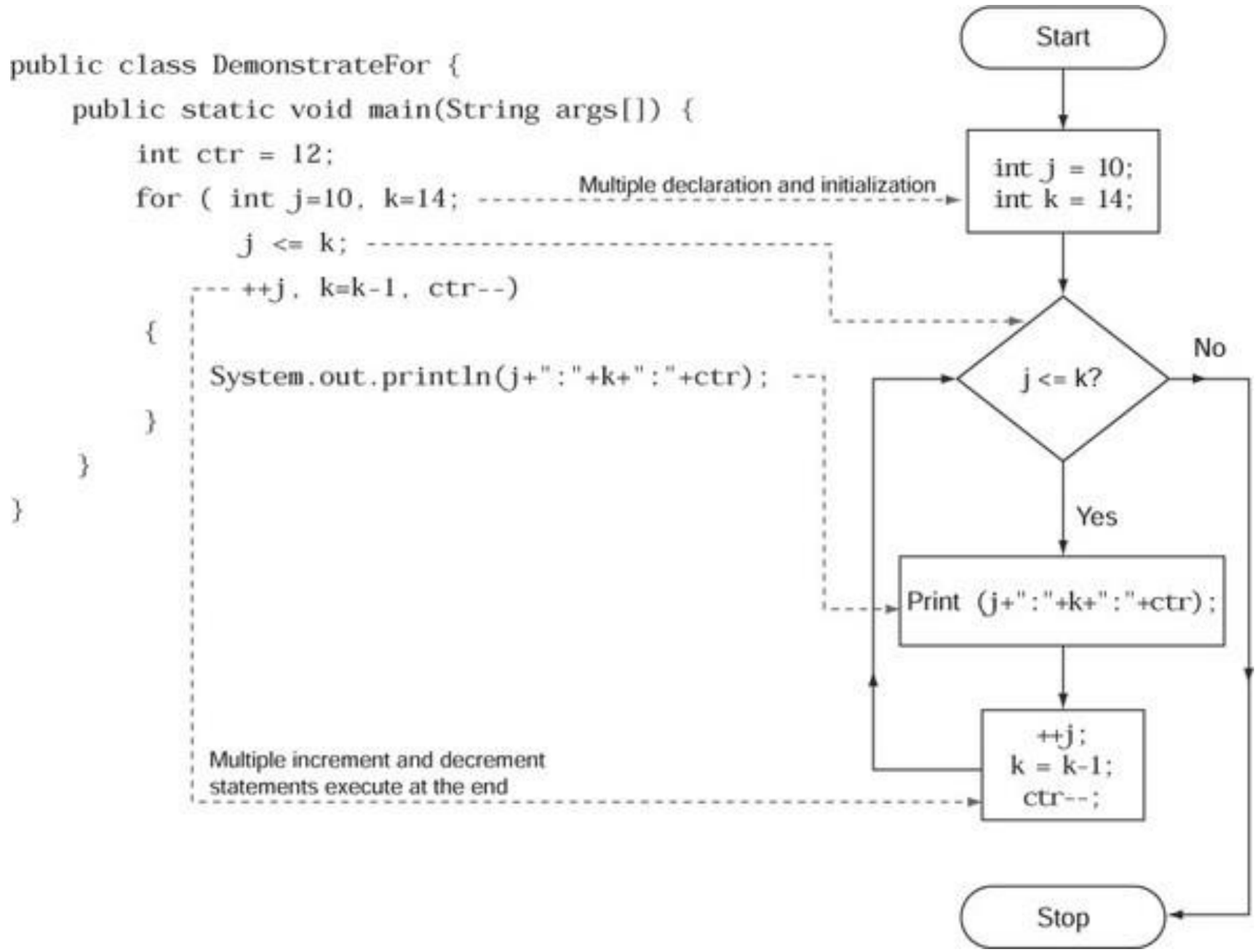


# Ещё одна диаграмма



# Как это работает

```
public class DemonstrateFor {
    public static void main(String args[]) {
        int ctr = 12;
        for ( int j=10, k=14;
              j <= k;
            --- ++j, k=k-1, ctr--)
        {
            System.out.println(j+":"+k+":"+ctr);
        }
    }
}
```



# Область действия

```
for(int i=0; i < 10; i++)
    System.out.print("Value is: "+i);
System.out.println(i);  // DOES NOT COMPILE
```

# Туда и обратно

```
int sum = 0;
int[] array = {12, 23, 5, 7, 19};
for (int index = 0; index < array.length; index++)
  sum += array[index];
```

```
int sum = 0;
int[] array = {12, 23, 5, 7, 19};
for (int index = array.length - 1; index >= 0; index--)
  sum += array[index];
```

# Бесконечный цикл

```
for( ; ; )
    System.out.println("Hello World");
```

- Компилятор считает `for(;;)` за `for(;true;)`.

# Множественные выражения

```
int x = 0;
for(long y = 0, z = 4; x < 5 && y < 10; x++, y++) {
   System.out.print(y + " "); }
System.out.print(x + " ");
```

# Декларация

```
for (int i = 0, j = 1, k = 2; … ; …) …;
```

```
for (int i = 0, String str = “@”; … ; …) …;  // (3) Compile-time error
```

# Повторная декларация

```
int x = 0;
for(int x = 4; x < 5; x++) {    // DOES NOT COMPILE
   System.out.print(x + " ");
}
```

```
int x = 0;
for(x = 0; x < 5; x++) {
   System.out.print(x + " ");
}
```

# Инициализация

```
int i, j, k;  // Variable declaration
for (i = 0, j = 1, k = 2; … ; …) …;          // (4) Only initialization
```

# Не смешивать

```
// (5) Not legal and ugly:
for (int i = 0, System.out.println(“This won’t do!”); flag; i++) { // Error!
  // loop body
}

// (6) Legal, but still ugly:
int i;                                          // Declaration factored out.
for (i = 0, System.out.println(“This is legal!”); flag; i++) {     // OK.
  // loop body
}
```

# Секция update

Секция *ForUpdate* стандартного `for`-цикла допускает лишь:

– присваивающее выражение;
– операторы инкремента / декремента (как префиксные, так и постфиксные);
– вызов метода;
– создание экземпляра класса;

```
public class ForIncrementStatements {
    public static void main(String args[]) {
        String line = "ab";
        for (int i=0; i < line.length(); ++i, printMethod())
            System.out.println(line.charAt(i));
    }
    private static void printMethod() {
        System.out.println("Happy");
    }
}
```

**The increment block can also call methods.**

**printMethod is called by the for loop's increment block.**

# Изменения налету

```
for(int i=0; i<10; i++)
    i = 0;

for(int j=1; j<10; j++)
    j--;

for(int k=0; k<10; )
    k++;
```

# После завершения

- После завершения `for(i = 0; i < n; i++)` значение **i** равно **n**, а вовсе не **n-1**.

# Трансформер

- Стандартный `for` – наиболее фундаментальный и гибкий цикл из всех; он способен заменить любую другую циклическую конструкцию.

Любую секцию `for`-цикла можно переставить в его тело

```
int i, s, count = 10;
for (i = 0, s = 0; i < count; i++) { s+=i; }
-----------------------------------------
int i, s, count = 10;
for(i = 0, s = 0; i < count; s+=i, i++);
-----------------------------------------
int s = 0, count = 10;
for(int i=0; i < count; s+=i++);
-----------------------------------------
int i=0, s=0, count=10;
for(; i< count; s+=i){ i++; }
-----------------------------------------
int i=0, s=0, count=10;
for(; i < count; ){ s+=i; i++; }
-----------------------------------------
int i=0, s=0, count=10;
for(; ;){ s+=i++; if (i>count) break; }
```

# Ещё бесконечный цикл

- `for(...;...;`<*no update*>`){`<*no update*>`}` создает синтаксически корректный, но бесконечный цикл.

# Упражнение

Which of the following `for` statements is valid?

Select the one correct answer.

(a)
```
int j = 10; for (int i = 0, j += 90; i < j; i++) { j-
  -; }
```

(b)
```
for (int i = 10; i = 0; i--) {}
```

(c)
```
for (int i = 0, j = 100; i < j; i++, --j) {;}
```

(d)
```
int i, j; for (j = 100; i < j; j--) { i += 2; }
```

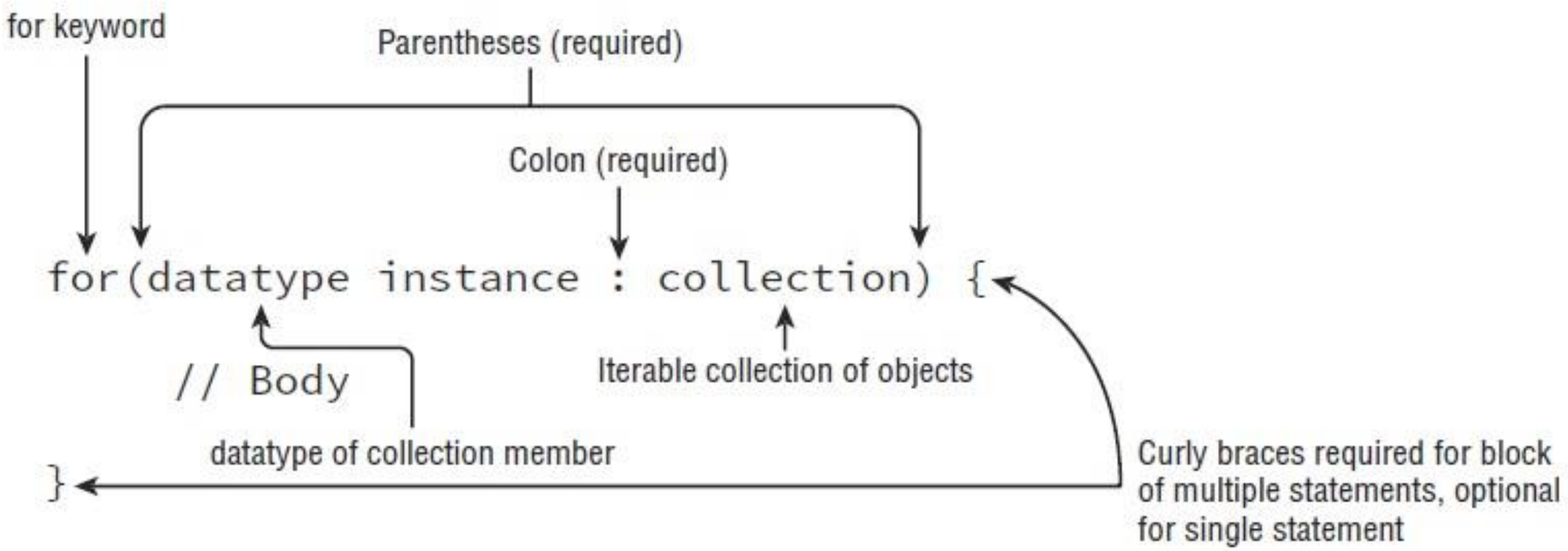
(e)
```
int i = 100; for ((i > 0); i--) {}
```

## Ответ: C

# Вопросы?

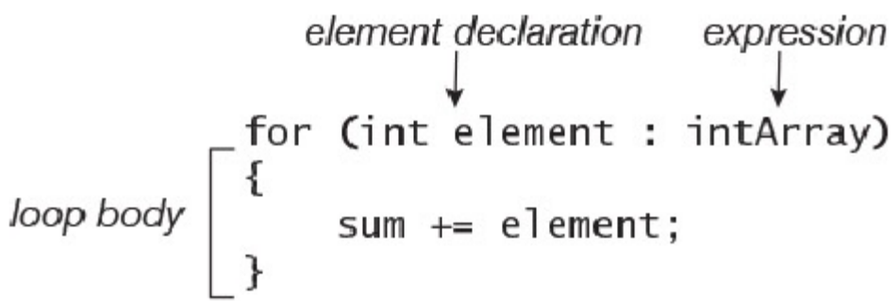
# Цикл for-each

# Структура for-each



# Конвертация

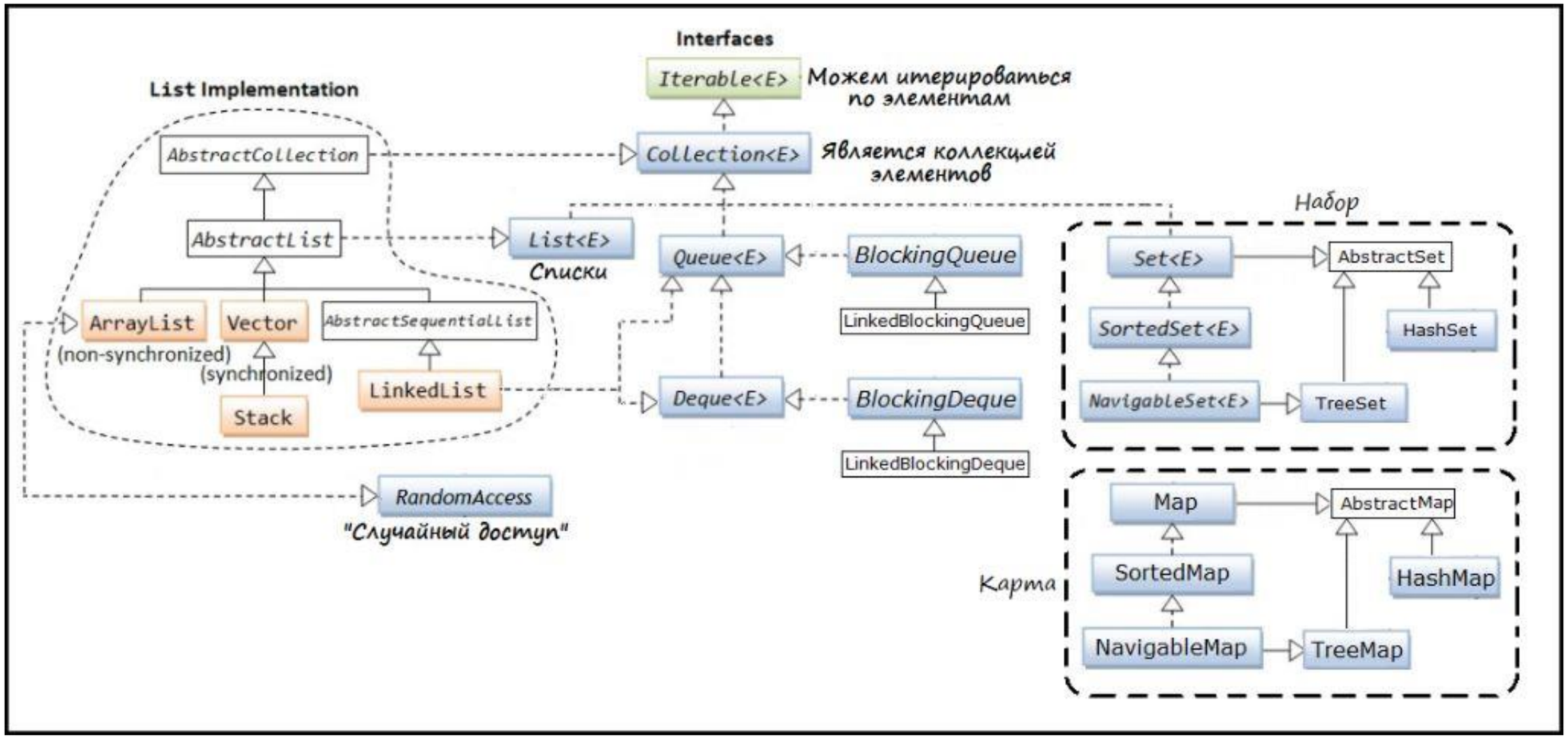
```
int sum = 0;
int[] intArray = {12, 23, 5, 7, 19};
for (int index = 0; index < intArray.length; index++) { // (1) using for(;;)
loop
  sum += intArray[index];
}
```



# Область применения

- Цикл `for-each` чаще всего применяется для поэлементного обхода структур данных или коллекций, например, типа **List**. С другой стороны, с его помощью нельзя:
  - инициализировать или модифицировать элементы массива;
  - удалять элементы из коллекций;
  - обходить сразу несколько коллекций или массивов внутри одного цикла.

# Collection Framework



# Ссылки на объекты

```
StringBuilder myArr[] = {
                new StringBuilder("Java"),
                new StringBuilder("Loop")
                 };
for (StringBuilder val : myArr)
    System.out.println(val);
for (StringBuilder val : myArr)
    val.append("Oracle");
for (StringBuilder val : myArr)
    System.out.println(val);
```

Iterates through array myArr and prints Java and Loop

Appends Oracle to value referenced by loop variable val

Iterates through array myArr and prints JavaOracle and LoopOracle

```
StringBuilder myArr[] = {
                new StringBuilder("Java"),
                new StringBuilder("Loop")
                 };
or (StringBuilder val : myArr)
    System.out.println (val);
for (StringBuilder val : myArr)
    val = new StringBuilder("Oracle");
for (StringBuilder val : myArr)
    System.out.println (val);
```

Iterates through array myArr and prints Java and Loop

Assigns new StringBuilder object to reference variable val with value Oracle

Iterates through array myArray and still prints Java and Loop

# Не тут-то было

```
ArrayList<StringBuilder> myList= new ArrayList<>();
myList.add(new StringBuilder("One"));
myList.add(new StringBuilder("Two"));
for (StringBuilder val : myList)
    System.out.println (val);
for (StringBuilder val : myList)
    val = null;
for (StringBuilder val : myList)
    System.out.println(val);
```

**Doesn't remove an object from list; sets value of loop variable to null**

```
One
Two
One
Two
```

# Инициализация

- Нельзя использовать уже существующую / ранее объявленную переменную в декларационной части «перебирающего» `for`. Что любопытно, эта переменная (даже примитивного типа!) может быть `final`. Кстати, это единственный модификатор, который можно ставить внутри декларации `for-each`:

```
int[] arr = {0, 1};
for (final int e : arr) {  }                    // VALID
```

# Как это работает

```
public void printNames(String[] names) {
    for(int counter=0; counter<names.length; counter++)
        System.out.println(names[counter]);
}
```

```
public void printNames(String[] names) {
    for(String name : names)
        System.out.println(name);
}
```

# Упражнение

```
int[] intArr = {1, 2, 4, 8, 16};
```

**Which two code fragments, when used independently, print all elements in the array?**

A.
```
for (int i : intArr) {
  System.out.print(intArr[i] + " "); }
```

B.
```
for (int i : intArr) {
  System.out.print(i + " "); }
```

C.
```
for (int i=0 : intArr) {
  System.out.print(intArr[i] + " ");
  i++; }
```

D.
```
for (int i=0; i < intArr.length; i++) {
  System.out.print(i + " "); }
```

E.
```
for (int i=0; i < intArr.length; i++) {
  System.out.print(intArr[i] + " "); }
```

F.
```
for (int i; i < intArr.length; i++) {
  System.out.print(intArr[i] + " "); }
```

## Ответ: BE

# Используем for-each

```
final String[] names = new String[3];
names[0] = "Lisa";
names[1] = "Kevin";
names[2] = "Roger";

for(String name : names) {
   System.out.print(name + ", ");
}
```

```
Lisa, Kevin, Roger,
```

```
List<String> values = new ArrayList<String>();
values.add("Lisa");
values.add("Kevin");
values.add("Roger");

for(var value : values) {
   System.out.print(value + ", ");
}
```

```
Lisa, Kevin, Roger,
```

```
String names = "Lisa";
for(String name : names) {   // DOES NOT COMPILE
   System.out.print(name + " ");
}
```

```
String[] names = new String[3];
for(int name : names) {  // DOES NOT COMPILE
   System.out.print(name + " ");
}
```

# Используем for

```
List<String> names = new ArrayList<String>();
names.add("Lisa");
names.add("Kevin");
names.add("Roger");

for(int i=0; i<names.size(); i++) {
   String name = names.get(i);
   if(i > 0) {
      System.out.print(", ");
   }
   System.out.print(name);
}
```

```
Lisa, Kevin, Roger
```

```
int[] values = new int[3];
values[0] = 1;
values[1] = Integer.valueOf(3);
values[2] = 6;

for(int i=1; i<values.length; i++) {
   System.out.print((values[i]-values[i-1]) + ", ");
}
```

```
2, 3,
```

# Под капотом

```
for(String name : names) {
   System.out.print(name + ", ");
}
for(int i=0; i < names.length; i++) {
   String name = names[i];
   System.out.print(name + ", ");
}
```

names is an array of String

```
for(int value : values) {
   System.out.print(value + ", ");
}
for(Iterator<Integer> i = values.iterator(); i.hasNext(); ) {
   int value = i.next();
   System.out.print(value + ", ");
}
```

values is an instance of List<Integer>

# Как можно

```
// Some 1-dim arrays:
int[]     intArray =    {10, 20, 30};
Integer[] intObjArray = {10, 20, 30};
String[]  strArray =    {“one”, “two”};

// Some 2-dim arrays:
Object[][] objArrayOfArrays = {intObjArray, strArray};
Number[][] numArrayOfArrays = {{1.5, 2.5}, intObjArray, {100L, 200L}};
int[][]    intArrayOfArrays = {{20}, intArray, {40}};

// Iterate over an array of Strings.
// Expression type is String[], and element type is String.
// String is assignable to Object (widening conversion).
for (Object obj : strArray) {}

// Iterate over an array of ints.
// Expression type is int[], and element type is int.
// int is assignable to Integer (boxing conversion)
for (Integer iRef : intArrayOfArrays[0]){}

// Iterate over an array of Integers.
// Expression type is Integer[], and element type is Integer.
// Integer is assignable to int (unboxing conversion)
for (int i : intObjArray){}
```

# Как тоже можно

```
// Iterate over a 2-dim array of ints.
// Outer loop: expression type is int[][], and element type is int[].
// Inner loop: expression type is int[], and element type is int.
for (int[] row : intArrayOfArrays)
  for (int val : row) {}

// Iterate over a 2-dim array of Numbers.
// Outer loop: expression type is Number[][], and element type is Number[].
// Outer loop: Number[] is assignable to Object[] (widening conversion).
// Inner loop: expression type is Object[], and element type is Object.
for (Object[] row : numArrayOfArrays)
  for (Object obj : row) {}

// Outer loop: expression type is Integer[][], and element type is Integer[].
// Outer loop: Integer[] is assignable to Number[].
// Inner loop: expression type is int[], and element type is int.
// Inner loop: int is assignable to double.
for (Number[] row : new Integer[][] {intObjArray, intObjArray, intObjArray})
  for (double num : new int[] {}) {}
```

# Как нельзя

```
// Expression type is Number[][], and element type is Number[].
// Number[] is not assignable to Number.
for (Number num : numArrayOfArrays) {}       // Compile-time error.

// Expression type is Number[], and element type is Number.
// Number is not assignable to int.
for (int row : numArrayOfArrays[0]) {}       // Compile-time error.

// Outer loop: expression type is int[][], and element type is int[].
// int[] is not assignable to Integer[].
for (Integer[] row : intArrayOfArrays)       // Compile-time error.
  for (int val : row) {}

// Expression type is Object[][], and element type is Object[].
// Object[] is not assignable to Integer[].
for (Integer[] row : objArrayOfArrays) {}    // Compile-time error.

// Outer loop: expression type is String[], and element type is String.
// Inner loop: expression type is String, which is not legal here. Not an
array.
for (String str : strArray)
  for (char val : str) {}                    // Compile-time error.
```

# Упражнение

```
class Test {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3};
        for ( *** ) { }
    }
}
```

**Which option(s) can replace *** so that the class will compile and run successfully? (Choose all that apply)**

A. int e : arr
B. int i = 0; i < 0; i++
C. ; ;
D. int i; i < 4; i++
E. boolean b = true; b; b = !b

# Вопросы?

# Домашнее задание

## Тест

# Пожалуйста, пройдите опрос

https://otus.ru/polls/17821/

# Спасибо за внимание!

## Только приятных повторений!