Java Course
Lecture 2 - Control flow statements

IT Learning &
Outsourcing Center

www.pragmatic.bg

Lecture 2 - Summary

- Ternary operator
- Switch
- Loops
- while
- for
- do-while
- Keywords - break and continue
- Arrays and array manipulation

? :
Ternary operator

- The ternary operator is a logical construct. Named that way because it uses tree operands.
- General Form

```
<boolean expression> ? <if true result> : <if false result>
```

- Example code :

```
int alarmTime = isWeekEnd ? 8 : 10;
```

Switch

- The switch statement allows for decision to be made on the bases of the value of a certain variable of a numeric type or String, then based on that value a decision path is chosen.
- Syntax :

```
switch (key) {
        case value: break;
        default: break;
}
```

Problem to solve

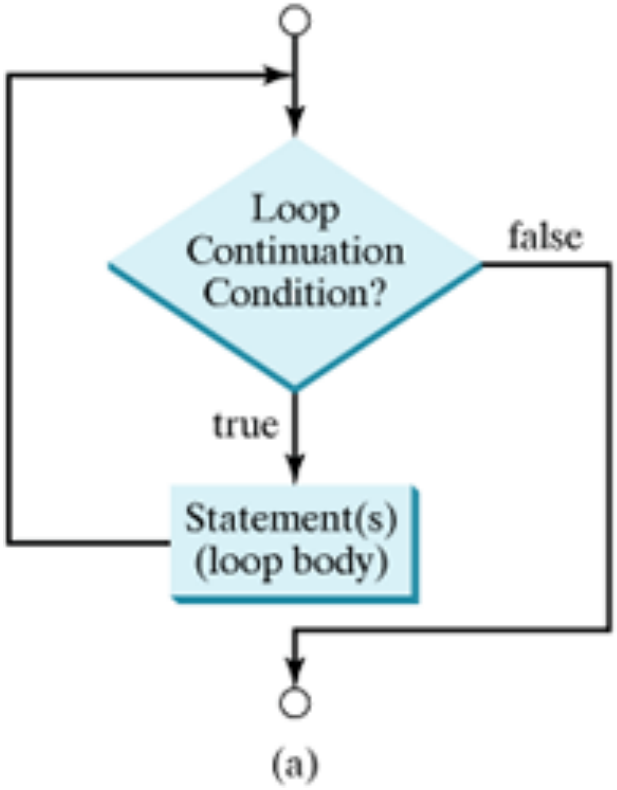
- Print all the numbers
  - From 1 to 5
  - From 1 to 1000
  - From 1 to n
  - From n to m

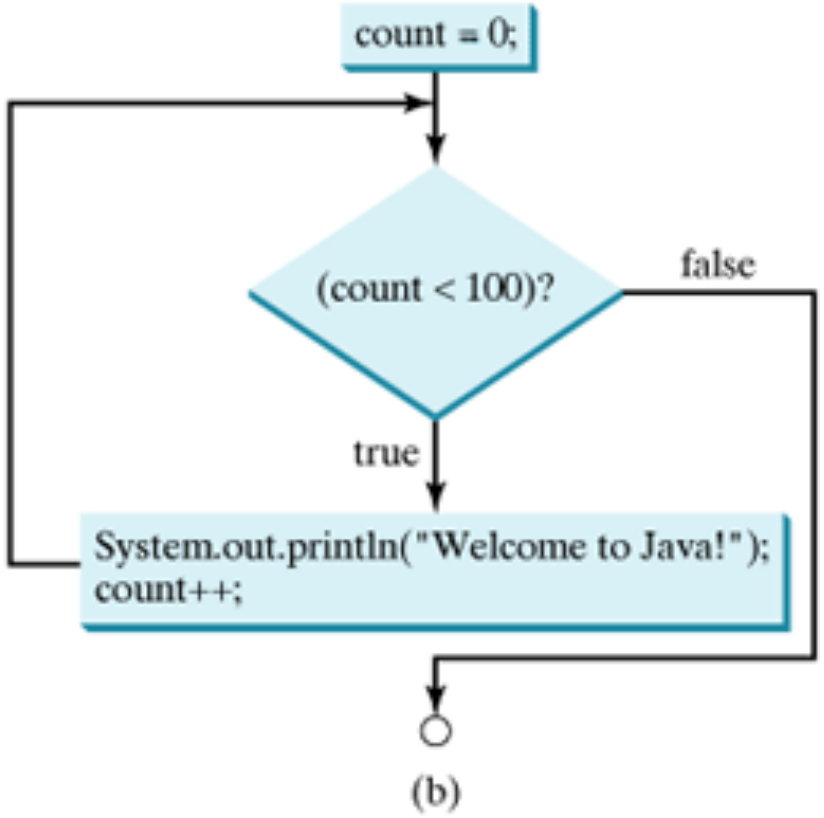
What is loop?

- A loop is a structure that allows sequence of statement to be executed more times in a row
- Loops have a boolean condition and a block of code for execution. While the condition is true, the block is being executed.
- A loop that never ends is called an infinite loop

While loop

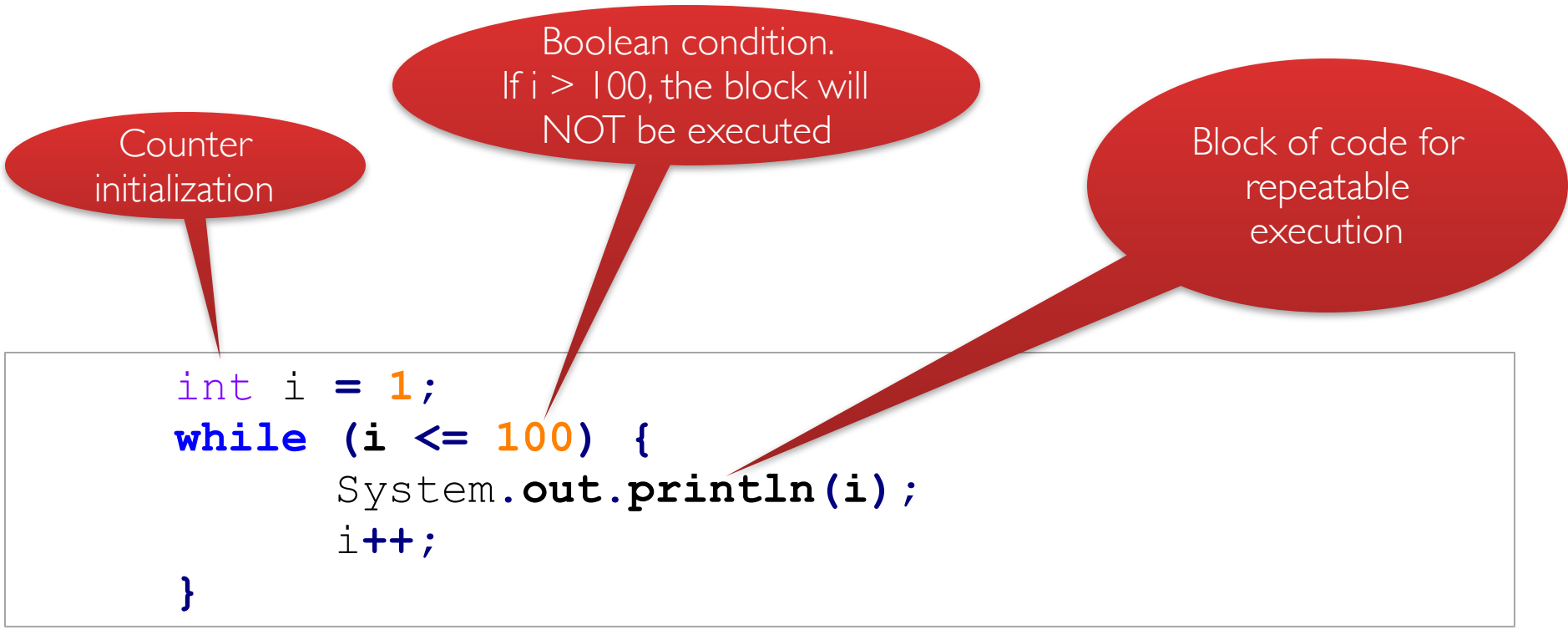
- While the condition is true, the block is being executed.





While loop

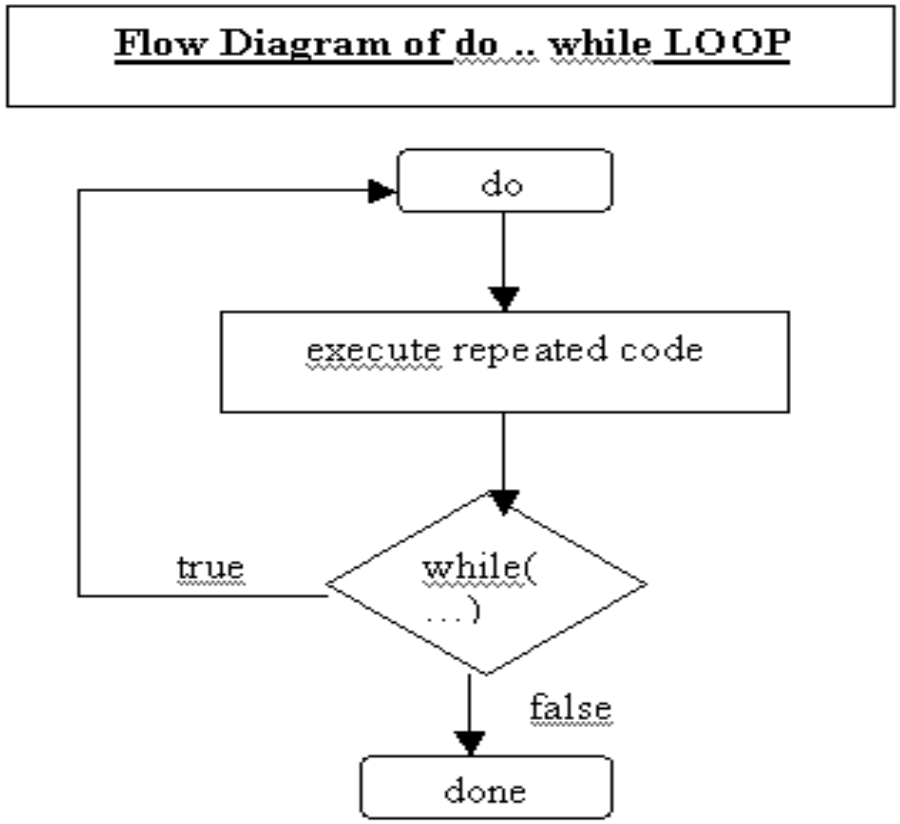
- While loop example:




```
int i = 1;
while (i <= 100) {
        System.out.println(i);
        i++;
}
```

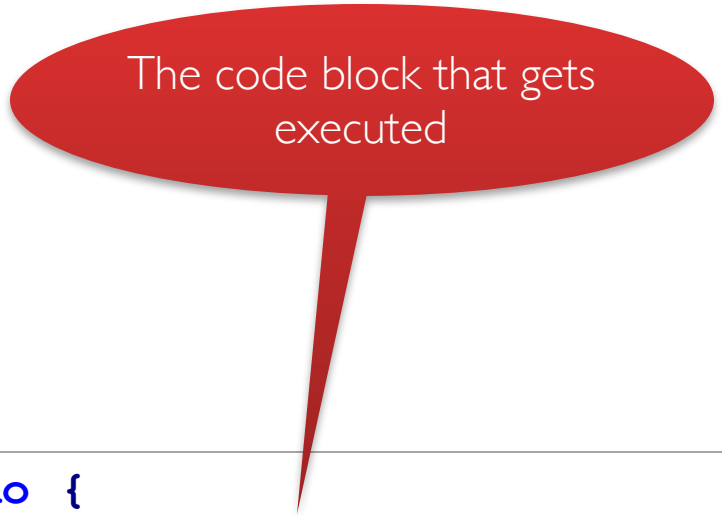
do-while

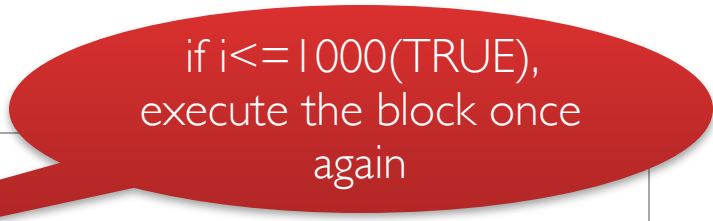
- Gets executed at least once
- Condition is after the execution



Example of do-while

- An example of a do-while loop:

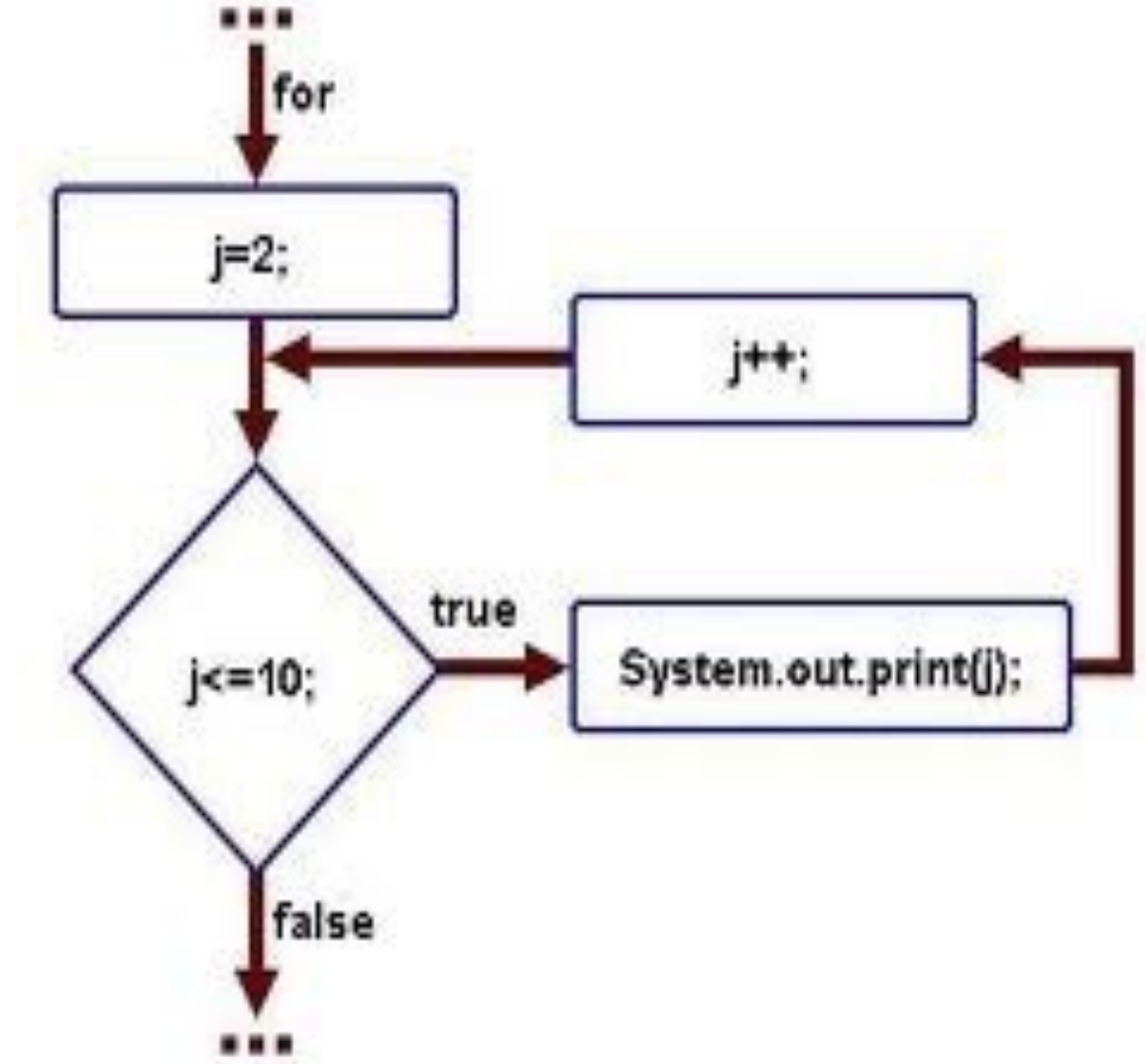





```
do {
        System.out.println(i);
        i++;
} while (i <= 1000)
```

For loop

- FOR loop:
  - Initialization
  - Condition
  - Increment
  - Body



Example of for loop

An example of for loop:

```
for (int i = 0; i < args.length; i++) {
        System.out.println(i);
}
```

- Initialization - `int i = 0`
- Condition - `i < args.length`
- Increment - `i++`
- Body - `{`

```
        System.out.println(i);
}
```

Problem

- Try to quit a for-loop during the execution of the repeatable block
- One possible solution is to set the counter to a value which will make the boolean condition quit the loop.... but there is a much better way

Break

- Break is a keyword
- A statement by itself
- It doesn't require anything else
- It stops the execution of the loop

```
for (int i = 0; i < 50; i++) {
        if (i == 7) {
                break;
        }
}
```

The loop will quit when i = 7

Problem

- Try to omit specific block of code in the body – for example sum all numbers between 1 and 100 but omit all numbers between 51 and 74
- Encapsulating the code in if-else statements may be used. Although for more complicated structures should be used for more complicated cases

Continue

- Continue is a keyword
- A statement by itself
- It doesn't require anything else
- It stops the current iteration of the loop, but doesn't stop the loop

```
for (int i = 0; i < 101; i++) {
        if (i > 51 && i < 71) {
                continue;
        }
        sum = sum + i;
}
```

if it is between 51 and 71, it will skip everything that is after continue

Array

- An array is simply put a grouping of "things" of the same type
- Arrays are objects themselves

Array Declaration

- Declaration tells the compiler the array's name and what type its elements will be

Example:

```
int[] ints;
Dimension[] dims;
float[][] twoDimensions;
```

- The square brackets can come before or after the array variable name:

```
int ints[];
```

Array Construction

- The declaration does not specify the size of an array
  - Size is specified at runtime, when the array is allocated via the new keyword
  - Example

```
int[] ints; // Declaration
ints = new int[25]; // Construction
```

- Declaration and construction may be performed in a single line:
  - `int[] ints = new int[25];`

Array Initialization

- When an array is constructed, its elements are automatically initialized to their default values
  - These defaults are the same as for object member variables
  - Numerical elements are initialized to 0
  - Non-numeric elements are initialized to 0-like values

Elements Initialization

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

Array Elements Initialization

- Initial values for the elements can be specified at the time of declaration and initialization

```
float[] diameters =
{1.1f, 2.2f, 3.3f, 4.4f, 5.5f};
```

- The array size is inferred from the number of elements within the curly braces

Access to Elements

- Accessing array elements:

```
int[] arr = new int[10];
arr[3] = 5; // Writing element
int value = arr[3]; // Reading element
```

- Elements access is range checked

```
int[] arr = new int[10];
int value = arr[10]; // ArrayIndexOutOfBoundsException
```

- Arrays has field **length** that contains their number of elements

Arrays – Example

- Пример

New for"each" loop

- In Java 5 a new loop a.k.a. for each loop was added.
- It works only for array or "Iterable" objects.
- Syntax :

```
int[] array = {1, 2, 3, 4, 5, 6};
for(int i : array){
    System.out.print(i+ ", ");
}
```

This will print 1, 2, 3 and so on

Note that "i" will not be a counter but an element of the array

Iterable is something we will speak about later

Multi-dimensional Arrays

- Multidimensional arrays in Java are actually arrays of arrays
- Defining matrix:

```
int[][] matrix = new int[3][4];
```

- Accessing matrix elements

```
matrix[1][3] = 42;
```

- Getting the number of rows/columns:

```
int rows = matrix.length;
int colsInFirstRow = matrix[0].length;
```

Examples

```
int[][] array = new int[2][3];
 array[0][0] = 5;
```

How to print it ?

```
for (int i = 0; i < array.length; i++) {
    for (int j = 0; j < array[i].length; j++) {
        System.out.print(array[i][j] + " ");
    }
    System.out.print("\n");
}
```

```
5 0 0
0 0 0
```

Example 2

```
int[][] example = new int[5][];
example[0] = new int[3];
example[1] = new int [5];
example[2] = new int[10];
example[3] = new int[7];
example[4] = new int[2];

for (int i = 0; i < example.length; i++) {
    for (int j=0; j < example[i].length; j++) {
        System.out.print(example[i][j] + " ");
    }
    System.out.print("\n");
}
```

```
0 0 0
0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0
0 0 0 0 0 0 0
0 0
```

foreach example


```
    for(int[] iArray : array) {
       for(int iElement: iArray){
            System.out.print(iElement + " ");
       }
       System.out.print("\n");
}
```

5 0 0
0 0 0

Q and A ?

Problems

- What's the difference between if/else statement and the ternary operator
- Is it possible to express a **for** loop with a while() loop ?
- What are the three params of a for loop ? What do they mean ?
- Can you access the iteration variable of loop outside of the loop ?
- If you have to iterate over an array, what's the best way to do it ?
- Is an array an object or a primitive ? What if it's an array of primitives ?
- What's the difference between **break** and **continue ?**
- When you create an array reference is there any memory allocated ?
- When do you specify the length of the array ?
- Can you change the maximum length of an array once that array has been constructed ?
- Write a small program to get the n-th element of an array.
- Create a program that prints out all numbers divisible by tree within a user specified interval. Ensure the user can enter only positive numbers as the interval limits !