Object-Oriented Programming I

Repetition (Loops)

Slides by Magdin Stoica
Updates by Georg Feil

Learning Outcomes

1. Explain the need for repeating a sequence of statements in computer programs
2. Categorize loop statements into counted (definite) loops and conditional loops
3. Define a counted loop using a “**`for`**” statement
4. Explain how the for-loop statement controls the flow of the program
5. Define a conditional loop using the “**`while`** ” statement
6. Explain how the while-loop statement controls the flow of the program
7. Define a conditional loop using the “**`do-while`**” statement
8. Explain how the do-while statement controls the flow of the program
9. Compare and contrast the 3 types of loop statements: for, while, do-while

Reading Assignments

- Introduction to Java Programming (required)
  - Chapter 4: Loops
- Head First Java (recommended)
  - Chapter 1, pages 10 – 12

The Need For Loops

- Suppose you are given the following problems:
  1. “Add up all the numbers from 1 to 100 and print out the result”
  2. “Input a number and print your name that many times”
  3. “Generate random numbers from 0.0 to 1.0 until you get one that’s larger than 0.99, then print it out & quit”
  4. “Print out all the numbers from 100 to 3000 that are divisible by 23”
- We can’t reasonably use huge amounts of copy & paste to solve these problems
- This is especially true if the number of times we need to do something depends on user input, or previous calculation

Loops are the Easy Way!

Loop Types

- Definite (counted) Loops
  - Repeat the statements in the loop block an exact, definite, number of times (e.g. 10, 20, 0)

- Useful when you know (or your program has calculated) how many times something needs to be repeated
  - For example, counting or enumerating items

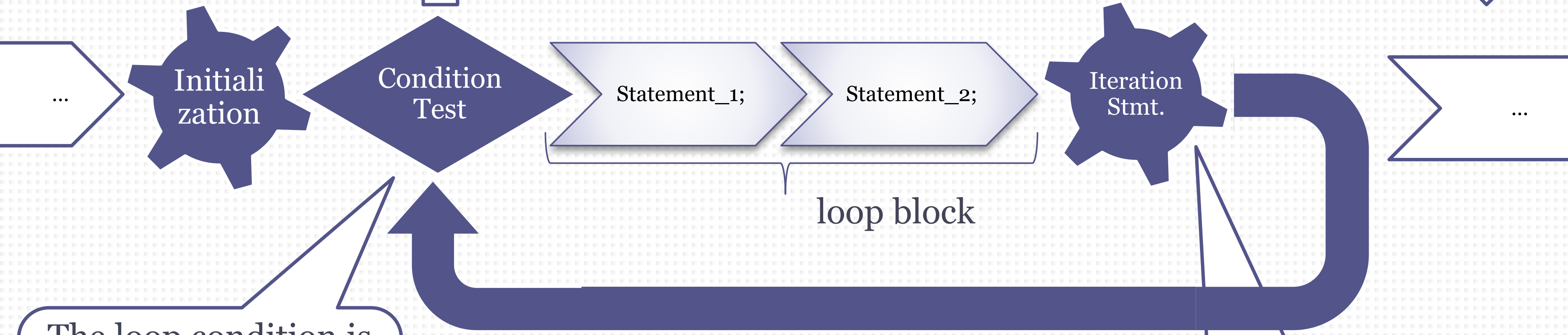
Loop Types

- Indefinite (conditional) Loops
  - Repeat the statements inside the loop block for as long as a condition is true
  - The condition is evaluated every time the sequence is about to be repeated to verify if another repetition is required
  - The condition can be evaluate before or after the sequence of statements is executed
  - If the loop condition never changes from true to false the sequence will be executed infinitely unless it is "broken"
    – infinite loop!
- Useful when you don't know how many times something needs to be repeated

The for Loop

- Counted (definite) loop that repeats the loop block a definite number of times
  - Defines and uses a counter that remembers how many times the loop has executed and ensures the code is only repeated a given number of times
- Consists of three components separated by two semicolons
  - Initialization component: declares (optionally) and initializes a counter variable
  - Condition test: defines a limit condition on the counter which stops the loop when it becomes false
    - e.g. if the counter reaches a certain value
    - Evaluated at the beginning of each iteration
  - Iteration component: changes the counter (e.g. increments, decrements).
    - Executed after all the statements in the loop block
    - Should affect the condition test, moving the loop closer to its conclusion

The for Loop



If the condition is false, the counter reached its limit so skip the loop block and continue with the next statement

The initialization is executed first.

The loop condition is evaluated at the beginning of each iteration.

Go back to the condition and re-evaluate

Execute the “next step” statement

The for Statement

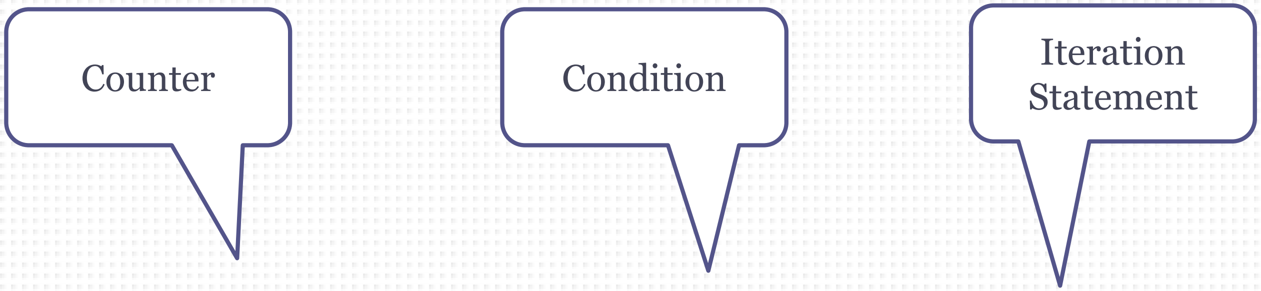
The three components are separated by two semicolons

```
for (<initialization> ; <condition>; <next step>)
{
        <statement 1>;
        <statement 2>;
        <statement 3>;
        …
}
```

Loop Block

These statements will repeat for as long as the condition is true

Example: for statement



Counter

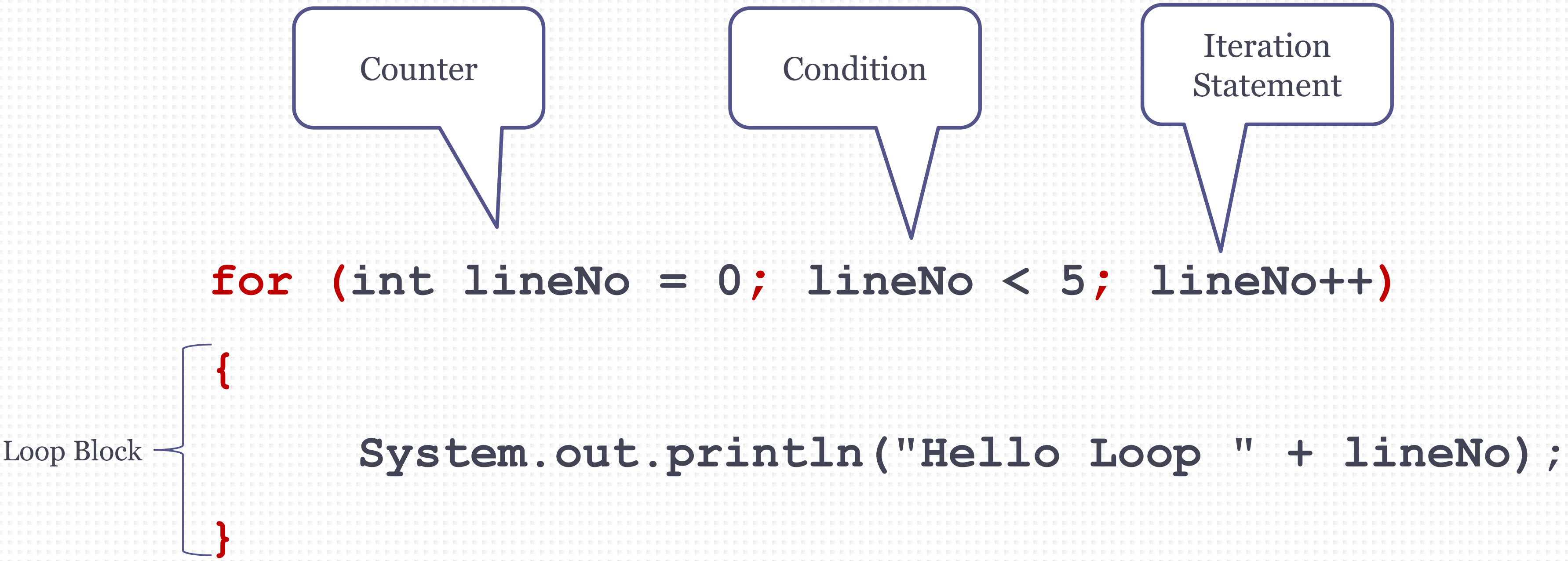
Condition

Iteration Statement

```
for (int lineNo = 0; lineNo < 5; lineNo=lineNo+1)
{
        System.out.println("Hello Loop " + lineNo);
}
```

Loop Block

Preview: for with increment operator



Counter

Condition

Iteration Statement

```
for (int lineNo = 0; lineNo < 5; lineNo++)
{
        System.out.println("Hello Loop " + lineNo);
}
```

Loop Block

This is the same as the loop on the previous slide,
++ means “increment variable by one”

Naming your counter variable

- Good names
  - lineNum: line number that goes from zero to 10 for every page
  - iPlayer: i is short for “index” so iPlayer is short for “index of player”
  - empNum: employee number
  - questionCounter
- Not so good names
  - i
  - j
  - k
  - l
  - These are good prefixes but do not identify what the counter is counting.

Use a good, representative name for the counter variable

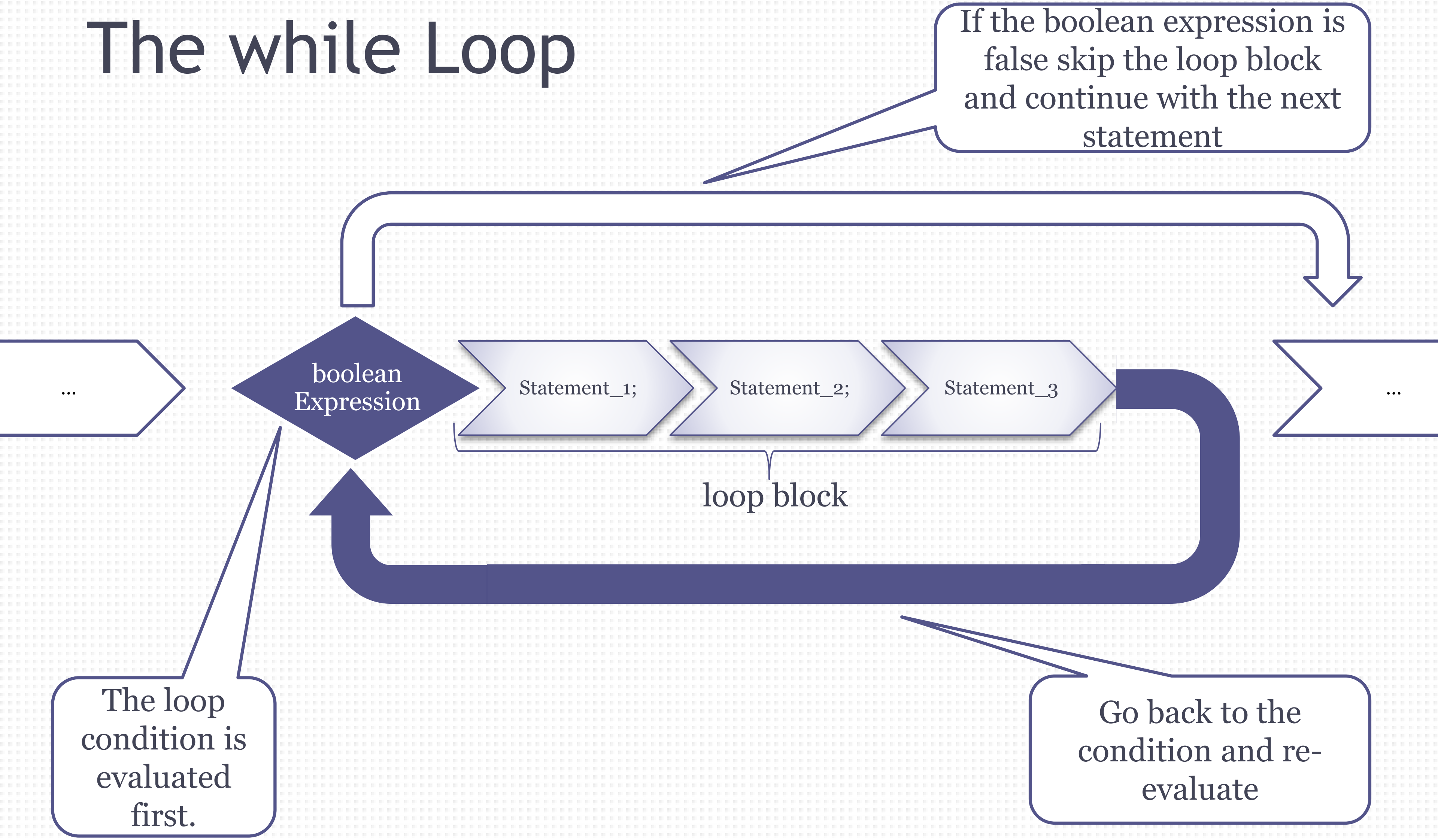
Exercise 1

- Solve problem #1 and #2 on slide 3 using for loops
- In problem #1 see what happens if you add up the numbers from 1 – 1000, 1 – 100000, 1 – 1000000, 1 – 1000000000 etc.

The while Loop

- Conditional (or counted) loop that repeats the loop block while the boolean expression it defines is true
- The condition is evaluated BEFORE the loop block is executed
  - It is possible for the statements inside the loop block to never execute
- The loop stops when the boolean expression becomes false
  - The sequence of statements that are executed in the loop must change one or more of the operands of the boolean expression
- Infinite loops
  - If the value of the boolean expression never changes and a break is not used the loop will run forever which would be a logic error

The while Loop


If the boolean expression is false skip the loop block and continue with the next statement
…
boolean Expression
Statement_1;
Statement_2;
Statement_3
…
loop block
The loop condition is evaluated first.
Go back to the condition and re-evaluate

The while Statement

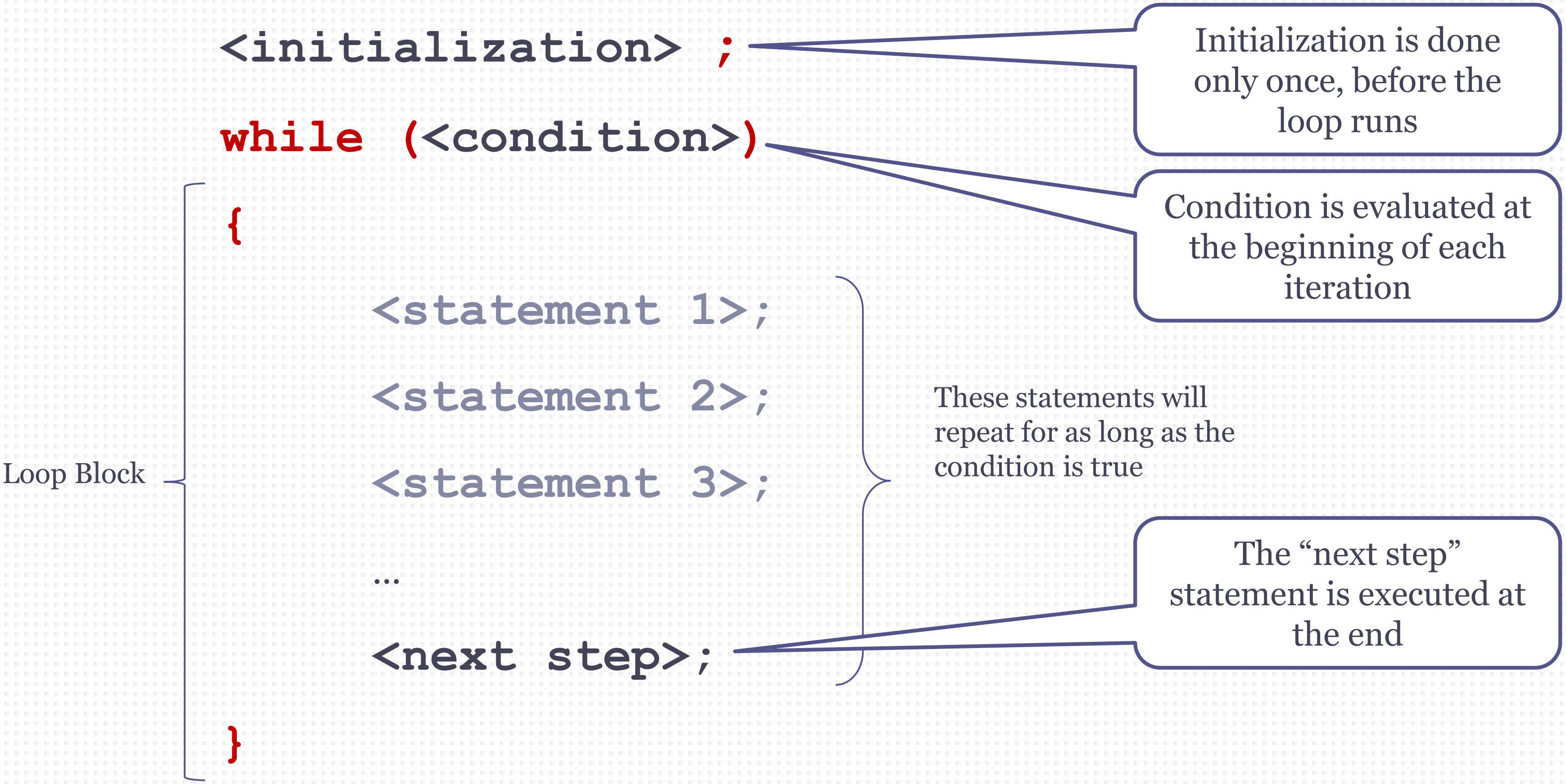
The loop condition. Loop will executes as long as the expression is true

```
while (<boolean expression>)
{
        <statement 1>;
        <statement 2>;
        <statement 3>;
        …
}
```

Loop Block

These statements will repeat for as long as the boolean expression is true

Definite Loops Using ‘while’



```
<initialization> ;
while (<condition>)
{
        <statement 1>;
        <statement 2>;
        <statement 3>;
        …
        <next step>;
}
```

Initialization is done only once, before the loop runs

Condition is evaluated at the beginning of each iteration

Loop Block

These statements will repeat for as long as the condition is true

The “next step” statement is executed at the end

Example: definite while statement

```
int lineNo = 0;
while (lineNo < 5) {
    System.out.println("Hello Loop " + lineNo);
    lineNo=lineNo+1;
}
```

Example: For loop, same as prev slide

```
for (int lineNo = 0; lineNo < 5; lineNo=lineNo+1) {
      System.out.println("Hello Loop " + lineNo);
}
```

Exercise 2

- Solve problem #3 and #4 on slide 3 using while loops
  - Add a counter and print how many times each loop executes (an extra counter is not necessary for the loop to work, it's just for information)
  - For problem #3 see what happens if you change the "0.99" by adding nines

Commenting a Loop

- Always comment a loop statement, regardless of type
- Your comment should explain why you need the loop
  - Example: “Go through all the employees to calculate the total number of hours worked”
- Commenting while and do-while statements:
  - Use the words “repeat” and “as long as”
  - Example: “repeat the game play as long as the user answers yes”
- Commenting “for” loops:
  - Use the words “for each … repeat…”
  - Use the words “repeat … n times”
  - Example: “Repeat the game for 10 rounds”

The “for” Loop in Detail: loop direction

- Forward or backward – your choice
  - Forward: increment the counter until it reaches an upper limit
  - Backward: decrement the counter until it reaches a lower limit

Looping Forward

Start with zero

counter < “limit”

++

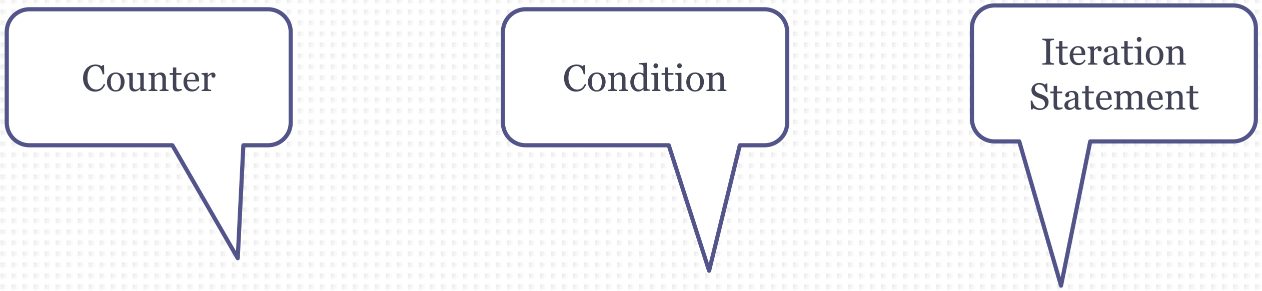
Looping Backward

Start with limit or limit - 1

counter >= 0 (or > 0)

--

Example: for statement backward



Counter

Condition

Iteration Statement

```
for (int lineNo = 4; lineNo >= 0; lineNo=lineNo-1)
{
        System.out.println("Hello Loop " + lineNo);
}
```

Loop Block

Exercise 3

- Write a for loop to print the numbers from 100 to 1 (backwards) in steps of 3,
  - 100, 97, 94, … , 1

The for Loop in Detail

- Any or all components of the for loop may be empty (missing)
  - No loop counter is defined: another variable defined outside of the loop may be used

```
for ( ; factor < 100; factor = factor * 2 )  {…}
```

  - No condition is defined: the break keyword is used to break the loop

```
for ( int total = 0;  ;  )
     …
     if ( <test> ) {
          break;
     }
     …
}
```

  - No iteration statement is defined: one or more statements inside the loop block move the loop to the next iteration

Indefinite Loops Using ‘for’

- Note that it’s possible to use a ‘for’ loop for indefinite loops similar to ‘while’
  - Just put in a condition test but no initialization or iteration statements
- This is not recommended, but you may see it done sometimes
  - Not good programming style, use ‘while’

'continue' and 'break'

Controlling loops: continue and break

- Do *partial* iterations using the continue keyword
  - It is possible to skip particular iterations
  - It is possible to partially execute an iteration
  - continue is used in conjunction with ‘if’ to decide that certain iterations could be entirely or partially skipped
- End a loop early with the break keyword
  - break ends the loop regardless of the result of the boolean expression
  - *Where have we seen this statement before? What did it do?*
- These keywords can be used will all types of loops

Example: continue

```
for (int num = 0; num < 100; num = num + 1)
{
      if (num % 2 == 0)
      {
            //skip the print statement for even numbers
            continue;
      }
      System.out.println(num + " is an odd number");
}
```

Loop Block

Example: break

```
int factor = 1;
while (true)
{
        factor = factor * 2;
        if (factor > 1000)
        {
                break;       // Quit the loop
        }
        System.out.println(factor);
}
```

Loop Block

Looping: Common Mistakes

- Getting one of the 3 components of the for loop wrong
  - Not initializing the counter properly.
    - Usually starts at zero and increments by one
  - Not testing the limit of the counter properly.
    - If the counter starts at zero use “less than the limit”
    - If the counter starts at 1 use “less than or equal the limit”
  - Not changing the counter properly in the iteration statement
    - Iterations could be skipped
    - Iterations could be repeated
- Not debugging to verify your assumptions
  - Always debug the first iteration to check the initialization
  - An iteration in the middle to see how it works
  - The last iteration to check the proper loop termination

Infinite Loops

while

```
while (5 > 3)
{
    …
}
```

for

```
for (count=1; count<9; )
{
    …
}
```

‘Infinite’ Loops… done properly

while

```
while (true)
{
    …
    if (…)
    {
        …
        break;
    }
    …
}
```

for (BUT DON’T DO THIS!)

```
for (;;)
{
    …
    if (…)
    {
        …
        break;
    }
    …
}
```

Exercise 4

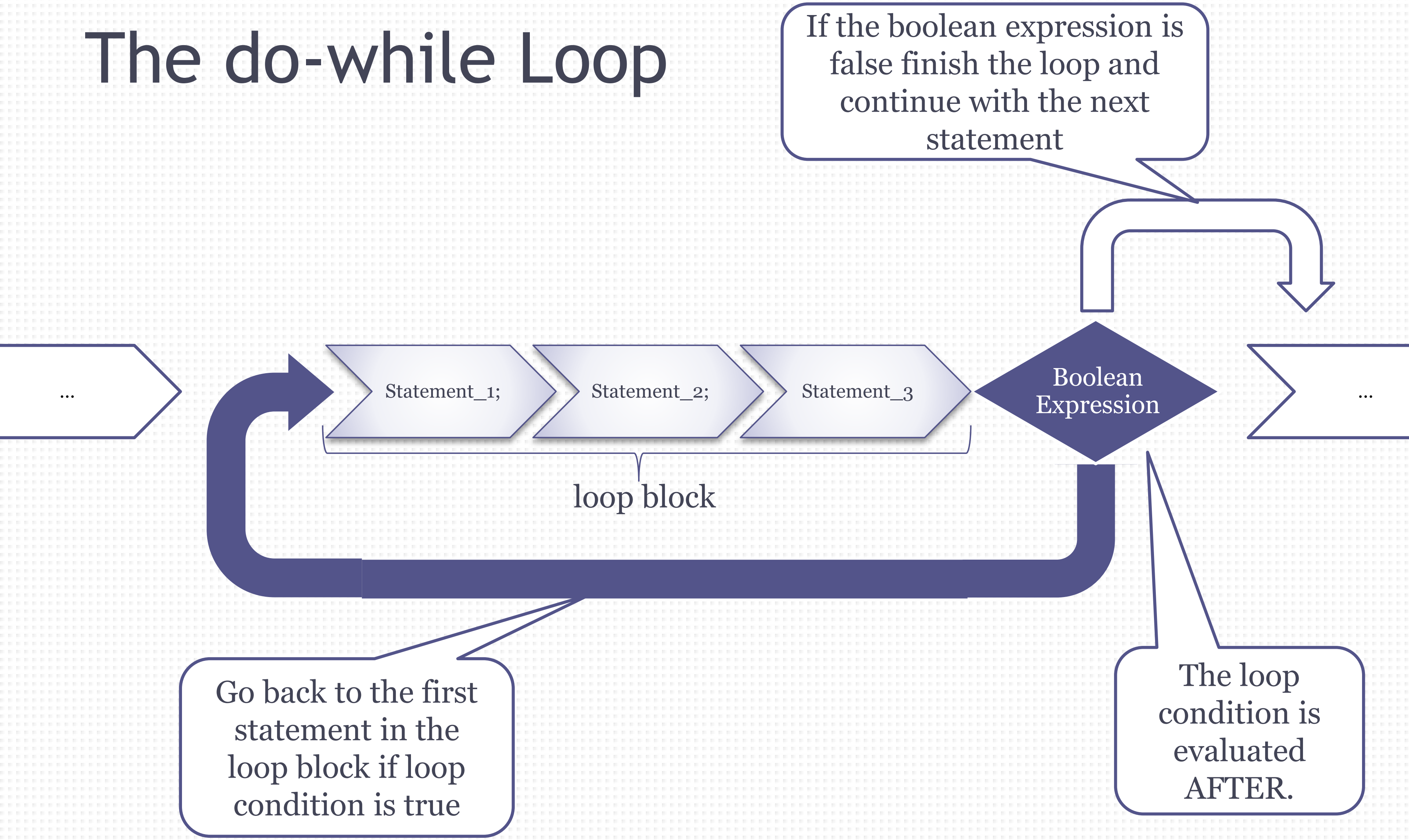
- Solve our assignment #1 part B elegantly & efficiently using loops
  - Which type of loop should we use?

- Use the Dr. Java debugger to examine how the loop works

The do-while Loop

- Conditional loop that repeats the loop block while the boolean expression it defines is true
- The condition is evaluated AFTER the loop block executes
  - The statements inside the loop block will always execute at least once
- The loop stops when the boolean expression becomes false
  - The sequence of statements that are executed in the loop must change one or more of the operands of the boolean expression
  - A "**`break`**" statement ends the loop regardless of the result of the boolean expression
- Infinite loops
  - If the value of the boolean expression never changes and a break is not used the loop will run forever which would be a logic error

The do-while Loop



If the boolean expression is false finish the loop and continue with the next statement

The loop condition is evaluated AFTER.

Go back to the first statement in the loop block if loop condition is true

The do-while Loop (pseudocode)

```
do
{
        <statement 1>;
        <statement 2>;
        <statement 3>;
        …
} while (<boolean expression>);
```

Loop Block

These statements will execute at least once and repeat for as long as the boolean expression is true

The loop condition. Loop will executes as long as the expression is true

The only control structure that must end with a semicolon

do-while Loop Example: Input Until 0

```
Scanner inp = new Scanner(System.in);
int num;
do {
    System.out.print("Enter a number: ");
    num = inp.nextInt();
    System.out.println("You entered " + num);
} while (num != 0);
```

Loop condition is tested at the *end* of each iteration, always loops at least once

The only control structure that must end with a semicolon

Exercise 5: Reducing code duplication

- Examine your solution to Exercise 2 (problem #3 on slide 3, random number > 0.99 problem)
  - Did you need to call nextDouble() in two places?

- Convert the loop to a do-while loop and eliminate the duplicate code
  - Call nextDouble in only one place

Exercise 6: Combining loops and methods

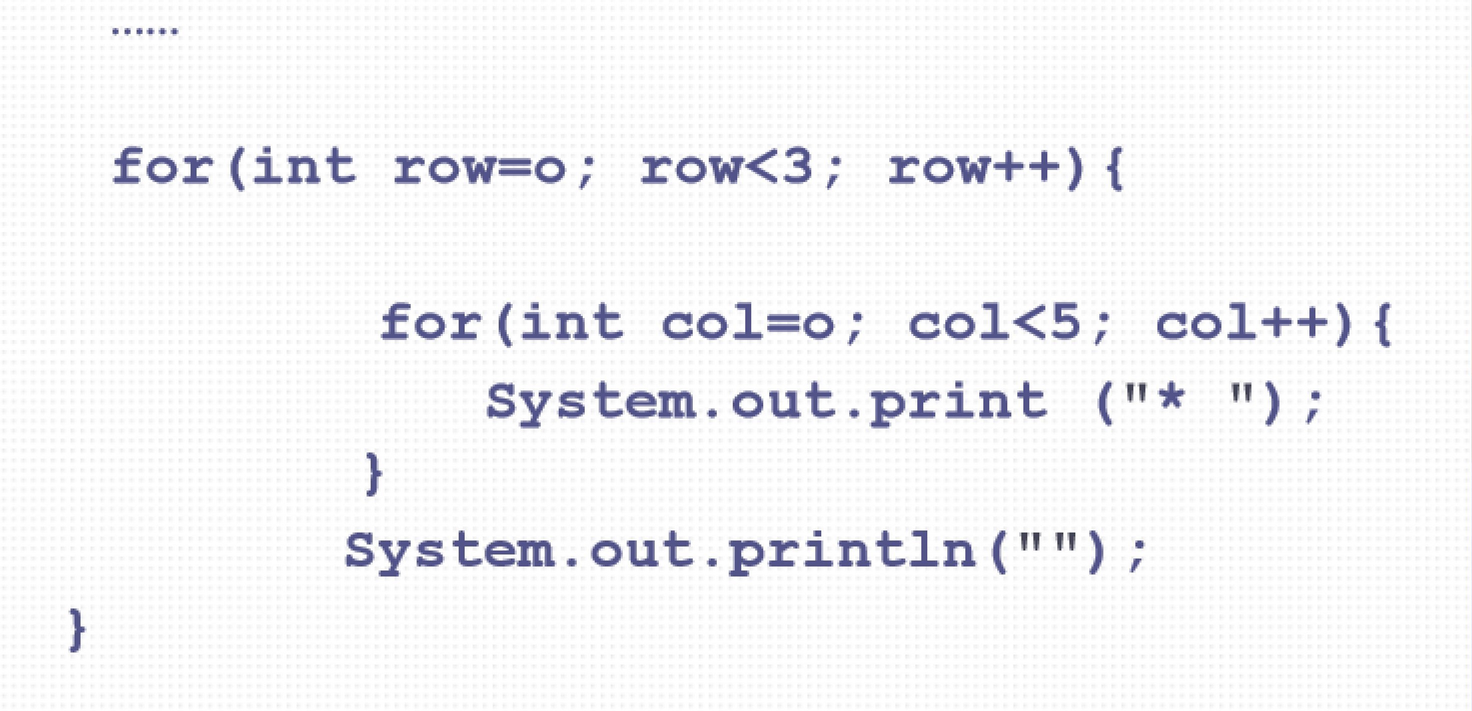
- Extend your solution from Exercise 5 (problem #3 on slide 3, random number > 0.99 problem)
- Move the code which finds a random number that’s > 0.99 into a separate method
  - The method should return the number it found
- Use method calls in a loop to make it produce 5 different results (repeat 5 times)
  - Don’t use ‘static’ except for the main method declaration (create an object of “self”)

The Need For Loops… more advanced

- Consider writing a program to print a rectangle with different dimensions (width x height)
  - 10 x 5
  - 20 x 10
- How can write one program that can draw both rectangles?
- What if the dimensions are given by the user in an interactive manner?
  - How can we draw the rectangle when we don’t know before hand what the dimension are?

```
OOOOOOOOOOOOOOOOOOOO
OOOOOOOOOOOOOOOOOOOO
OOOOOOOOOOOOOOOOOOOO
OOOOOOOOOOOOOOOOOOOO
OOOOOOOOOOOOOOOOOOOO
OOOOOOOOOOOOOOOOOOOO
OOOOOOOOOOOOOOOOOOOO
OOOOOOOOOOOOOOOOOOOO
OOOOOOOOOOOOOOOOOOOO
OOOOOOOOOOOOOOOOOOOO
```

Nested Loops

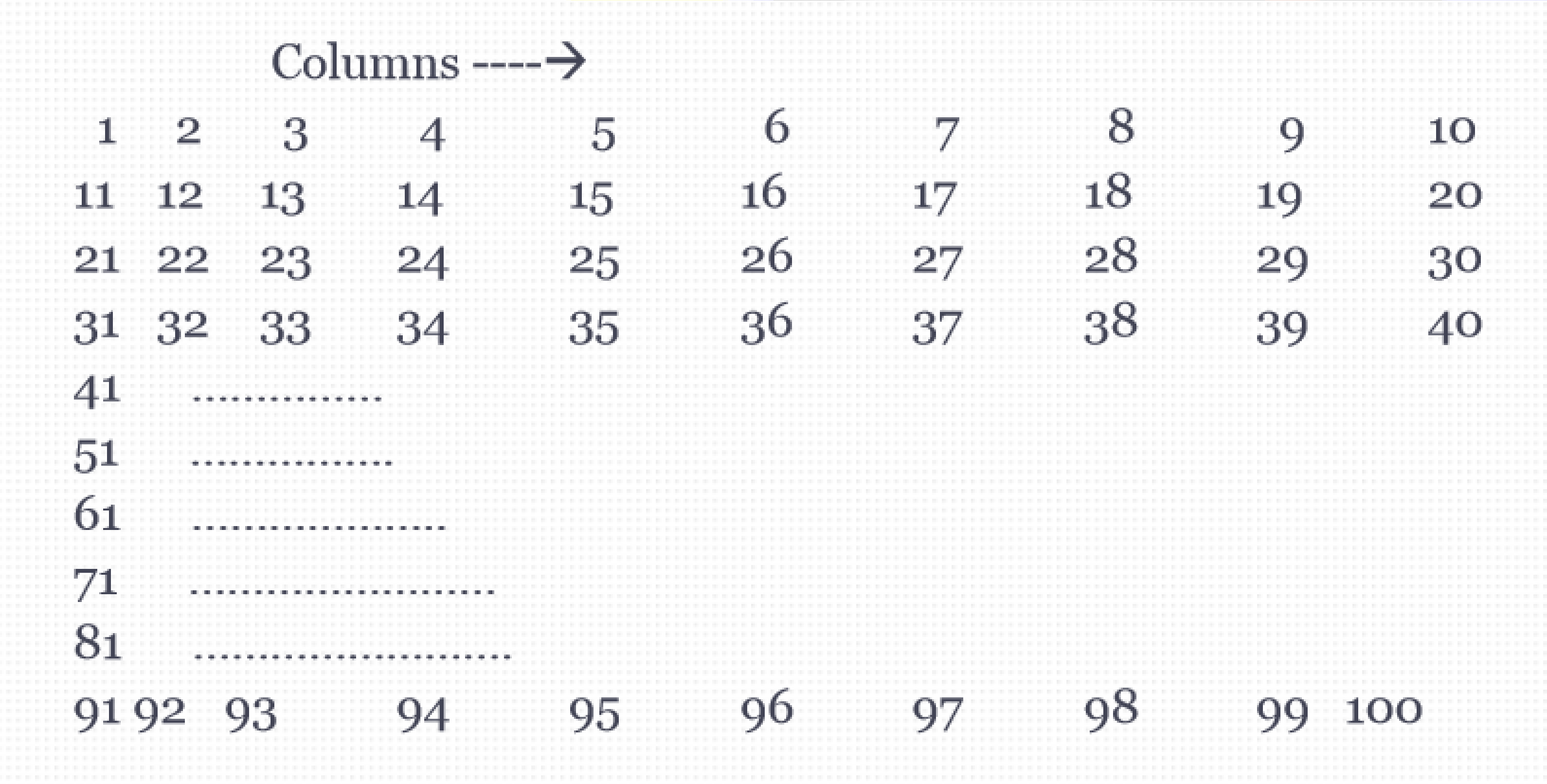


```
  ......

  for(int row=o; row<3; row++){

          for(int col=o; col<5; col++){
              System.out.print ("* ");
          }
         System.out.println("");
}
```

Develop a nested for loop to generate the following table



Columns ----→

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	....								
51	....								
61	....								
71	....								
81	....								
91	92	93	94	95	96	97	98	99	100