

CSN-103: Fundamentals of Object Oriented Programming



Iteration Statements

- **Loops:** Repeatedly executes the same set of instructions until a termination condition is met
- Three types
 - while
 - do-while
 - for



while Loop Statement

- *while* repeats a statement(s) while its controlling expression is true
- General form

```
while(condition)
{
// body of loop
}
```

- The body of the loop will be executed as long as the conditional expression is true
- When condition becomes false, control passes to the next line of code immediately following the loop

Programmer Husband

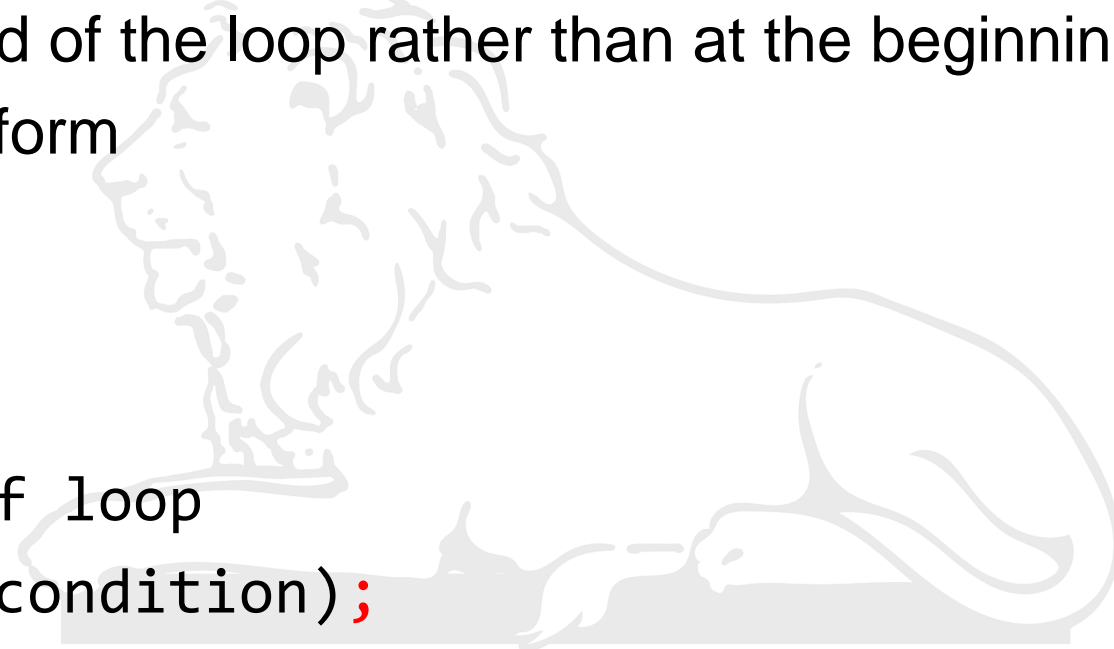


A programmer was walking out of door for work, his wife said “while you're out, buy some milk” and he never came home.

do-while Loop Statement

- If the *condition* of a while loop is initially false, then the body of the loop will never execute
- Sometimes, you would like to test the termination expression at the end of the loop rather than at the beginning
- General form

```
do
{
// body of loop
} while (condition);
```

A faint, stylized illustration of a lion lying down, facing left, positioned behind the code block.

While vs. do while

```
while (not edge) {  
    run();  
}
```

```
do {  
    run();  
} while (not edge);
```



for Loop Statement

- The general form of the **for** statement:

```
for(initialization; condition; iteration)  
{  
    // body  
}
```

- for loop operates as follows
 - When the loop first starts, the initialization portion of the loop is executed
 - Sets the value of the *loop control variable*
 - Next, *condition* is evaluated (Boolean expression)
 - If this condition is true, then the body of the loop is executed. If it is false, the loop terminates
 - Next, the iteration portion of the loop is executed
 - Increments or decrements the loop control variable

for Loop Variations

- 1) Using the Comma ,
 - Can include more than one statement in the **initialization** and **iteration** portions of the for loop (**not for condition**)
- 2) Either the **initialization** or the **iteration** expression or both may be absent (what about the **condition??**)
- 3) The *For-Each* Version of the for Loop (Works with *collection*)

```
int nums[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };  
int sum = 0;  
for(int i=0; i < 10; i++)  
    sum += nums[i];
```

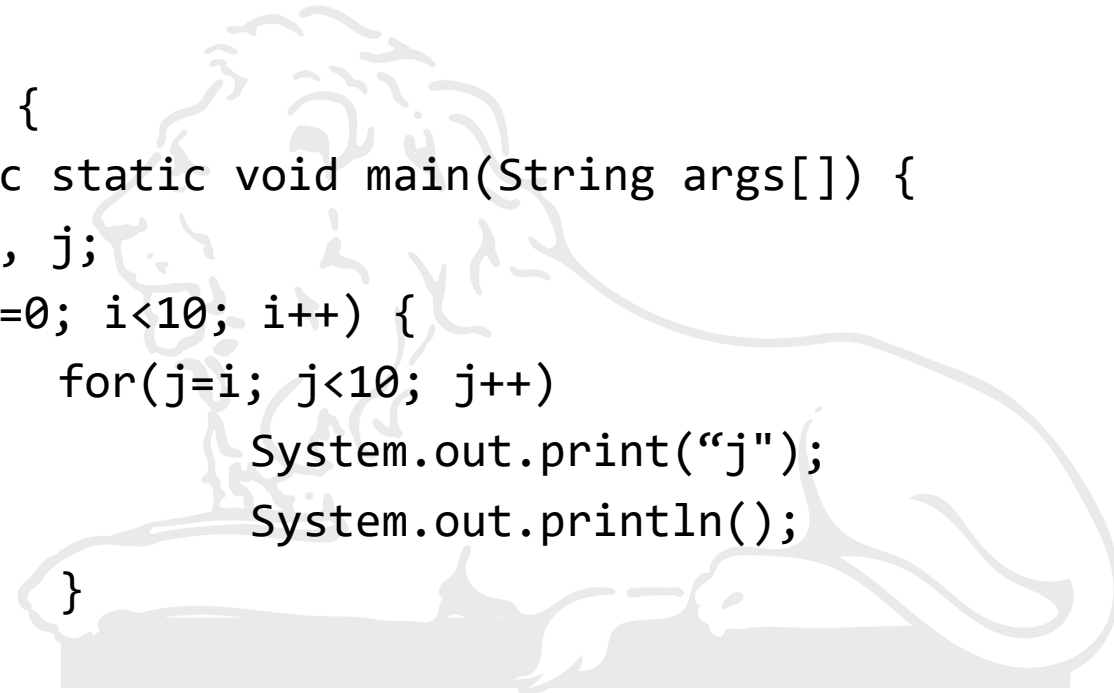
Equivalent to

```
int nums[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };  
int sum = 0;  
for(int x: nums)  
    sum += x;
```


Nested Loops

- One loop may be inside another loop
- Example:

```
class Nested {  
    public static void main(String args[]) {  
        int i, j;  
        for(i=0; i<10; i++) {  
            for(j=i; j<10; j++)  
                System.out.print("j");  
            System.out.println();  
        }  
    }  
}
```

A large, faint watermark of a lion statue, likely the Ashoka Lion Capital, is visible in the background of the slide, behind the code text.

Jump Statements

- Java supports three jump statements
 - **break** primarily used in two scenarios:
 - **switch** statement: Terminates a sequence of execution
 - **Loop (while and for)**: Causes loop to terminate early
 - **continue**
 - Forces an early iteration of a loop
 - Stop processing the remainder of the code in its body for a particular iteration
 - Control to be transferred directly to the **conditional** expression
 - **return**
 - Used to explicitly return from a method/function
 - Control transfers back to the caller of the method/function

