Programming II
COMP212

Object Oriented Programming With Java

basic programming concetpts review

Variables

Variables

- variable declarations

Identifiers

CODE

```
int x;
int y;
```

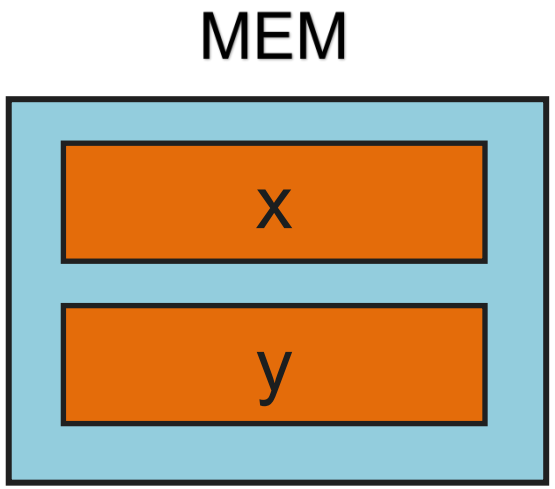
Data Types

datatype variableName;

Assignment Statements

- The syntax for assignment statements

variable = expression;




```
int x;
x = 40;
int y = 3;
int z = y + 30;
z = y - x;
```

Various Types in Java

- Primitive type
  - for storing simple values
- Reference type
  - for storing complex objects

Primitive Types

Type	Bytes	Range	storage forme
byte	1	[-127, 128]	8-bit signed
short	2	[-32k, 32K]	16-bit signed
int	4	[-2147483648, 2147483647]	32-bit signed
long	8	[-9223372036854775808, 9223372036854775807]	64-bit signed
float	4	Negative range: -3.4028235E +38 to -1.4E -45 Positive range: 1.4E -45 to 3.4028235E +38	32-bit IEEE 754
double	8	Negative range: -1.7976931348623157E + 308 to -4.9E -324 Positive range: 4.9E -324 to 1.7976931348623157E +308	64-bit IEEE 754
char	2	A, B, C ...	16-bit Unicode
boolean	1	true/false	

Primitive Types

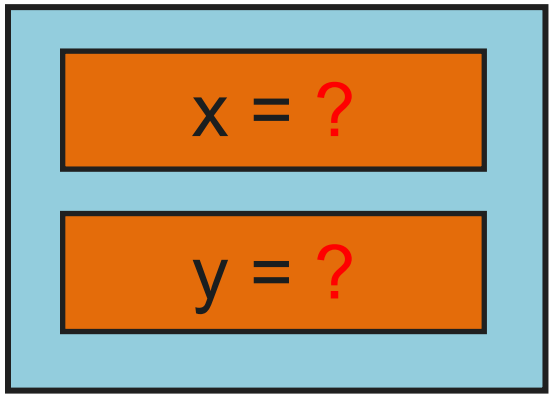
```
int someInt = 45;
long someLongInt = 123_456_789l;
float someFloat = 45.036f;
double someDouble = 3.467E120;
char someChar = 'G';
boolean someState = true ;
```

Mathematical Operations

CODE

```
int x;
x = 4 + 3 * 2;
int y = x - 6;
x = x * y;
```

MEM

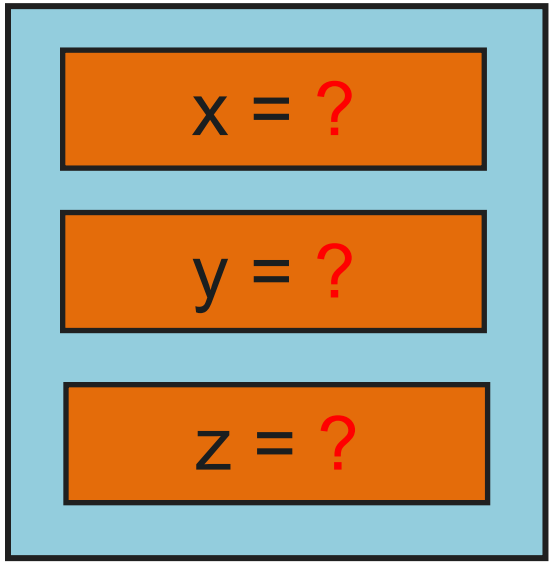


Mathematical Operations

CODE

```
int x;
x = 2;
int y = x * 3;
int z = y / 2;
x = (y + z) % 2;
```

MEM



Flow Control

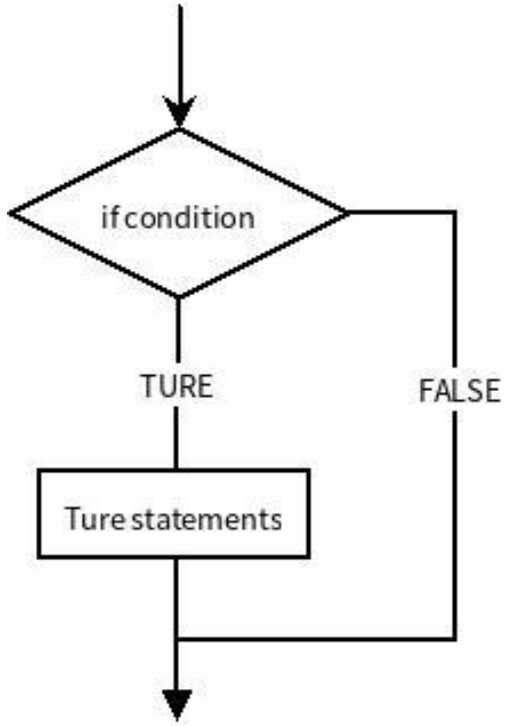
- Java executes one statement after the other in the order they are written
- Many Java statements are flow control statements:
  - Selections: if, if else, swithch
  - Loop: for, while, do while
  - Escapes: break, continue, return

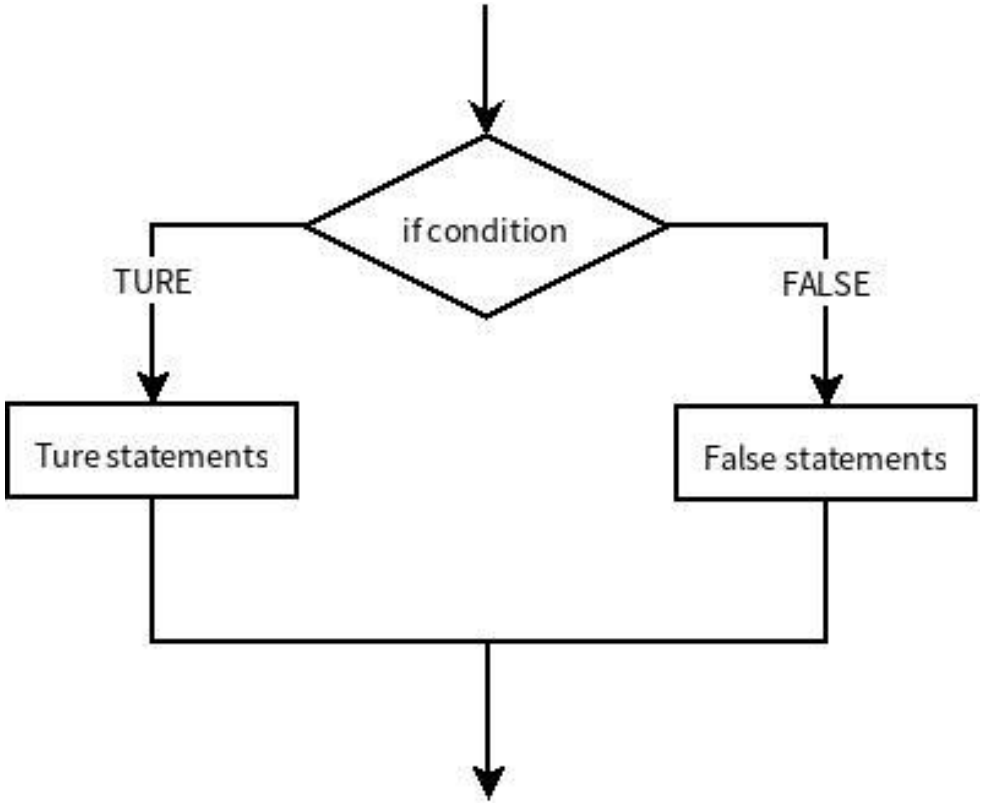
Selections

if... ; if...else...

Conditional statements (if)

- An **if** statement is a construct that enables a program to specify alternative paths of execution.





boolean Data Type

- boolean Relational Operators

Java Operator	*Mathematics Symbol*	*Name*	*Example (radius is 5)*	*Result*
`<`	$<$	Less than	`radius < 0`	`false`
`<=`	$\leq$	Less than or equal to	`radius <= 0`	`false`
`>`	$>$	Greater than	`radius > 0`	`true`
`>=`	$\geq$	Greater than or equal to	`radius >= 0`	`true`
`==`	$=$	Equal to	`radius == 0`	`false`
`!=`	$\neq$	Not equal to	`radius != 0`	`true`

Case Study: Computing BMI

- Body mass index (BMI) is a measure of health based on height and weight. It can be calculated by taking your weight in kilograms and dividing it by the square of your height in meters. The interpretation of BMI for people 20 years or older is as follows:

BMI	Interpretation
$\text{BMI} < 18.5$	Underweight
$18.5 \leq \text{BMI} < 25.0$	Normal
$25.0 \leq \text{BMI} < 30.0$	Overweight
$30.0 \leq \text{BMI}$	Obese

DEMO

Logical Operators

- Sometimes, whether a statement is executed is determined by a combination of several conditions. You can use logical operators to combine these conditions to form a compound Boolean expression. Logical operators, also known as Boolean operators, operate on Boolean values to create a new Boolean value.

Operator	*Name*	*Description*
`!`	not	Logical negation
`&&`	and	Logical conjunction
`||`	or	Logical disjunction
`^`	exclusive or	Logical exclusion

Logical Operators

- True Table of The Logical Operators

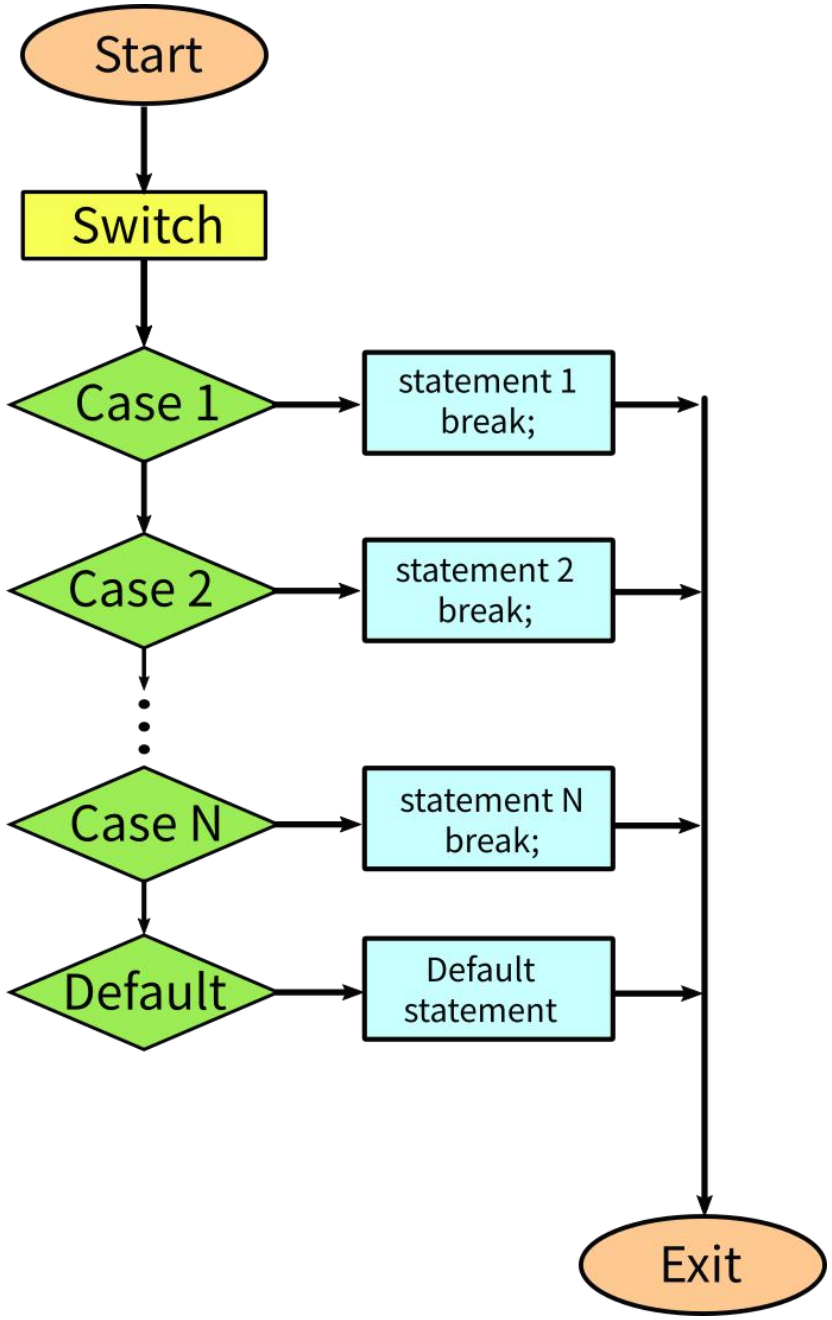
A	B	A && B	A || B	A ^ B	! A
T	T	T	T	F	F
T	F	F	T	T	F
F	T	F	T	T	T
F	F	F	F	F	T

T = ture ; F = false

DEMO

swithch statement

- A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.



DEMO

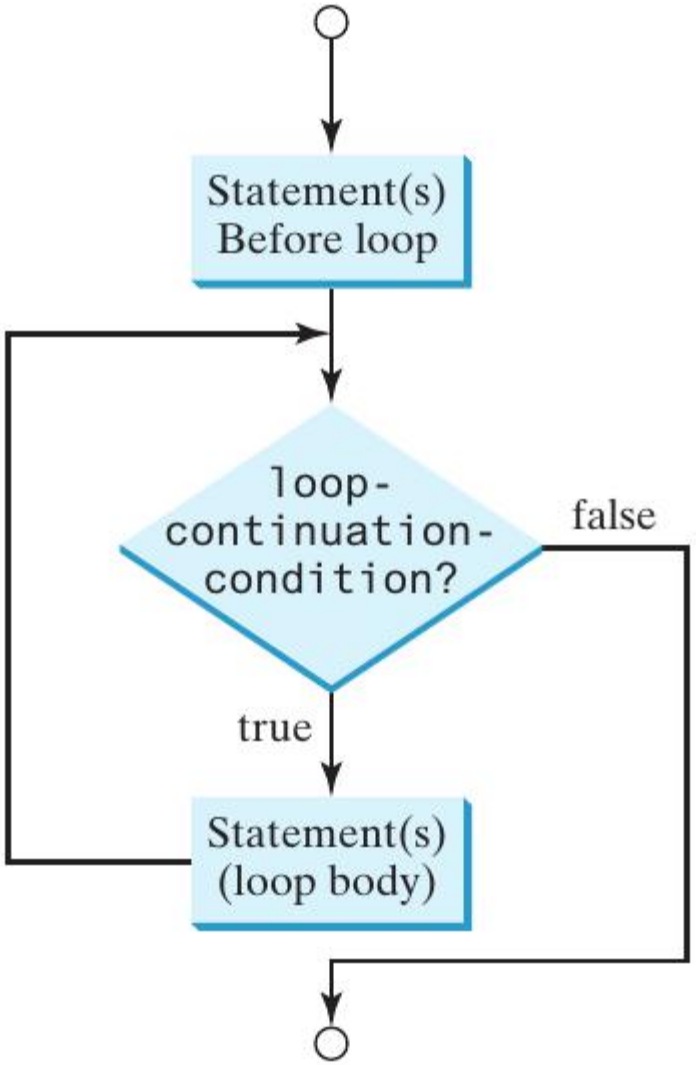
Loop

while ; for

while Loop

- Java provides a powerful construct called a loop that controls how many times an operation or a sequence of operations is performed in succession.

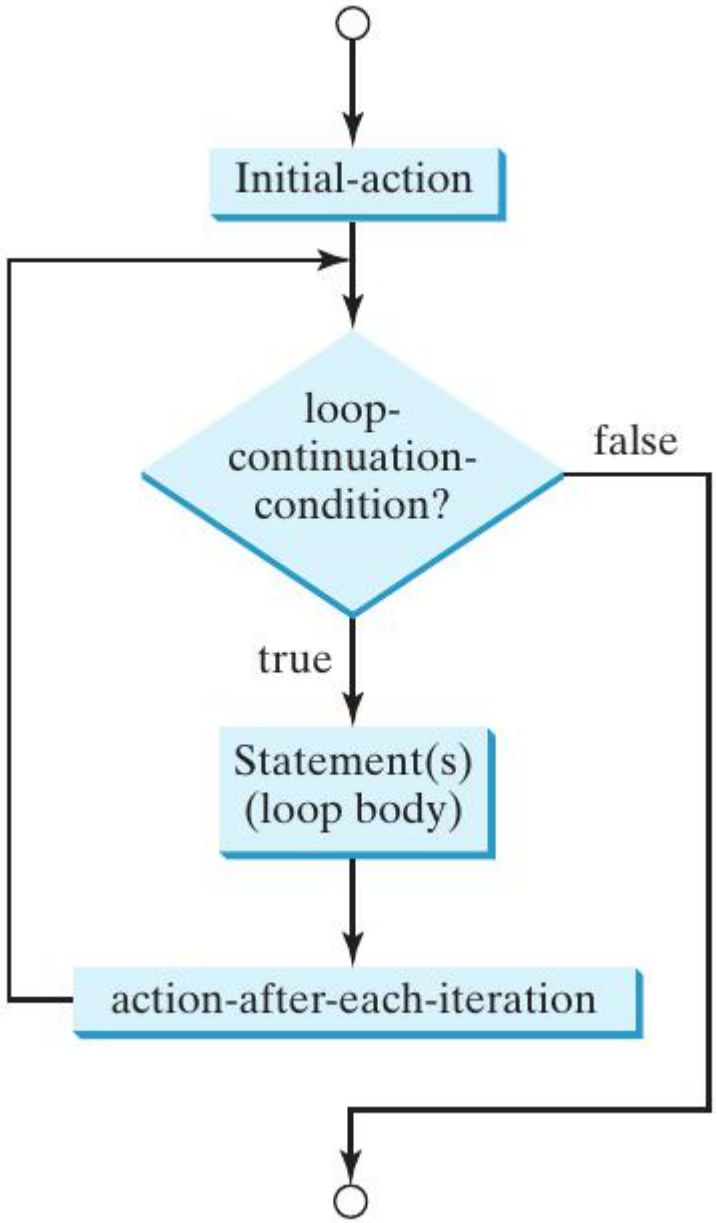
```
while (loop-continuation-condition) {
  // Loop body
  Statement(s);
}
```



DEMO

for loop

```
for (initial-action; loop-continuation-condition;
     action-after-each-iteration) {
  // Loop body;
  Statement(s);
}
```



DEMO

Break & Continue Statement

- when a **break** statment is encountered inside a loop, the loop is immediatly terminated and the program control resumes at the next statement following the loop.

- The **continue** statement is used in loop control structure when you need to the next iteration of the loop immediately. It can be used with for loop and while loop.

Java Comments

The Java comment are the statements that are not executed by the compliler and interpreter. The comments can be used to provide information or explanation about the variable, mehtod, class or any statement.

- Types of Java Comments
  1. Single Line Comment: //
  2. Multi Line Comment: /* */
  3. Documentation Comment: /** */

Function (Method)

Functions (Methods)

- Functions extract a computation out, giving it a name and parameters. You then can use the function to perform that computation without rewriting it. (reusable code and simplify coding).
- Technically speaking Java doesn’t have functions. It has ***methods*** since all code in Java is inside of objects.

Defining a Method

Define a method

modifier

return value type

method name

formal parameters

method header

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
     result = num1;
  else
     result = num2;

  return result;
}
```

method body

parameter list

method signature

return value

Invoke a method

```
int z = max(x, y);
```

actual parameters (arguments)

Why we need function ?

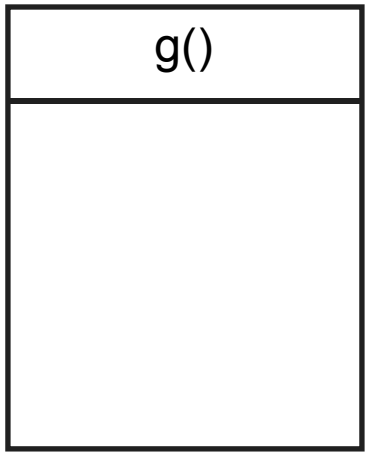
- factorial: the factorial of a positive integer n, denoted by $n!$, is product of all positive integers less than or equal to n:

$$n! \;=\; n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1$$

$$0! = 1$$

Calling a Method

```
int f(int x, int y) {
    if (x < y) {
        System.out.println("x < y");
        return y + x;
    }
    else {
        System.out.println("x >= y");
        if (x > 8) {
            return y + 7;
        }
    }
    return x - 2;
}
```




```
int g() {
    int a = f(3, 4);
    int b = f(a, 5);
    return b;
}
```

Functions (Methods)

function myFunction

```
int myFunction(int x, int y) {
    int z = 2 * x -y;
    return z * x;
}
```

function g

```
int g() {
    int a;
    a = myFunction(3, 7);
    int b = f(a*a);
    return b;
}
```

function f

```
int f(int n) {
    return 3 + myFunction(n, n+1);
}
```