Variables, Data Types and Operators

Basic Programming Teaching Team 2022

Objectives

After studying this material, students should be able to:

1. Understand and explain about data types
2. Describe and explain about variables
3. Understand and describe about Operators (Arithmetic Assignment, Combined Assignment, Increment, Decrement, Relational, Logic, Conditional, Bitwise, Casting)

Variable

- Variables are used in programming languages to store temporary values that can be reused later.
- Variables have a data type and name.
- The data type indicates the type of value in that variable.

What do you imagine about this picture?

Variable Type

- **Local variables** are variables that can only be recognized in sub-programs
- **Global variables** are variables that can be recognized throughout the program

Variable Writing Rules

- Variable names cannot use Java keywords, such as **if**, **main**, **for**, **else**, **class**, and so on
- Variable names may include letters, numbers (0-9), underscores (_), and dollar symbols ($), but symbols should be avoided
- Variable names should start with lowercase letters
- If the variable name is more than one word, the writing is combined and the word after it starts with a capital letter

Variable Writing Rules

- Writing format:

  <data type> <name> [= initial value]

  The initial value inside the [] sign is optional
- Example:

```
Int length;
int length = 125;
```

Data Type

- A data type is the type of data we want to store in a variable.
- Data types can be categorized into two groups, namely:
  1. Primitive data types
  2. Reference data types.

What do you imagine about this picture?

How about this picture?

Primitive Data Types

Data Type	Description	Size	Minimum	Maximum
boolean	true / false	1-bit		
char	Unicode character	16-bit		
byte	Integers	8-bit	-127	128
short	Integers	16-bit	-32768	32767
int	Integers	32-bit	-2147483648	2147483647
long	Integers	64-bit	-9223372036854775808	9223372036854775807
float	Floating point	32-bit	1.40129846432481707e-45	3.40282346638528860e+38
double	Floating point	64-bit	4.94065645841246544e-324	1.79769313486231570e+308

Variable Declaration

Declaration

```
int value;
double number;
float a, b, c;
```

Assigning a value

```
int value = 75;
double number = 2.5;
```

Print Variable

```
System.out.println(value);
System.out.println(a);
```

Or

```
System.out.println("Your score is " + value);
System.out.println("The number is " + a);
System.out.println("The vehicle mileage is " + x + " km");
```

Casting Data Types

- Casting is the process of assigning a primitive data type to another primitive data type
- **Widening casting** (auto): changes the data type from a smaller size to a larger data type

  byte → short → char → int → long → float → double

Illustration of widening casting

Casting Data Types

- **Narrowing casting** (manual): changes the data type from a larger size to a smaller data type

  double → float → long → int → char → short → byte

Illustration of narrowing casting

Example of Casting Data Types

- Widening casting (auto)

```
byte age = 9;
double myDouble = age;
System.out.println(age); //Output 9
System.out.println(myDouble); //Output 9.0
```

- Narrowing casting (manual)

```
double gpa = 3.78;
int myInt = (int) gpa;
System.out.println(gpa); //Output 3.78
System.out.println(myInt); //Output 3
```

ASCII

- ASCII stands for American Standard Code for Information Interchange.
- As the name implies, ASCII is used for information exchange and data communication.
- ASCII is a numeric code that represents a character.

Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char
0	0x00	000	0000000	NUL	32	0x20	040	0100000	space	64	0x40	100	1000000	@	96	0x60	140	1100000	`
1	0x01	001	0000001	SOH	33	0x21	041	0100001	!	65	0x41	101	1000001	A	97	0x61	141	1100001	a
2	0x02	002	0000010	STX	34	0x22	042	0100010	"	66	0x42	102	1000010	B	98	0x62	142	1100010	b
3	0x03	003	0000011	ETX	35	0x23	043	0100011	#	67	0x43	103	1000011	C	99	0x63	143	1100011	c
4	0x04	004	0000100	EOT	36	0x24	044	0100100	$	68	0x44	104	1000100	D	100	0x64	144	1100100	d
5	0x05	005	0000101	ENQ	37	0x25	045	0100101	%	69	0x45	105	1000101	E	101	0x65	145	1100101	e
6	0x06	006	0000110	ACK	38	0x26	046	0100110	&	70	0x46	106	1000110	F	102	0x66	146	1100110	f
7	0x07	007	0000111	BEL	39	0x27	047	0100111	'	71	0x47	107	1000111	G	103	0x67	147	1100111	g
8	0x08	010	0001000	BS	40	0x28	050	0101000	(	72	0x48	110	1001000	H	104	0x68	150	1101000	h
9	0x09	011	0001001	TAB	41	0x29	051	0101001	)	73	0x49	111	1001001	I	105	0x69	151	1101001	i
10	0x0A	012	0001010	LF	42	0x2A	052	0101010	*	74	0x4A	112	1001010	J	106	0x6A	152	1101010	j
11	0x0B	013	0001011	VT	43	0x2B	053	0101011	+	75	0x4B	113	1001011	K	107	0x6B	153	1101011	k
12	0x0C	014	0001100	FF	44	0x2C	054	0101100	,	76	0x4C	114	1001100	L	108	0x6C	154	1101100	l
13	0x0D	015	0001101	CR	45	0x2D	055	0101101	-	77	0x4D	115	1001101	M	109	0x6D	155	1101101	m
14	0x0E	016	0001110	SO	46	0x2E	056	0101110	.	78	0x4E	116	1001110	N	110	0x6E	156	1101110	n
15	0x0F	017	0001111	SI	47	0x2F	057	0101111	/	79	0x4F	117	1001111	O	111	0x6F	157	1101111	o
16	0x10	020	0010000	DLE	48	0x30	060	0110000	0	80	0x50	120	1010000	P	112	0x70	160	1110000	p
17	0x11	021	0010001	DC1	49	0x31	061	0110001	1	81	0x51	121	1010001	Q	113	0x71	161	1110001	q
18	0x12	022	0010010	DC2	50	0x32	062	0110010	2	82	0x52	122	1010010	R	114	0x72	162	1110010	r
19	0x13	023	0010011	DC3	51	0x33	063	0110011	3	83	0x53	123	1010011	S	115	0x73	163	1110011	s
20	0x14	024	0010100	DC4	52	0x34	064	0110100	4	84	0x54	124	1010100	T	116	0x74	164	1110100	t
21	0x15	025	0010101	NAK	53	0x35	065	0110101	5	85	0x55	125	1010101	U	117	0x75	165	1110101	u
22	0x16	026	0010110	SYN	54	0x36	066	0110110	6	86	0x56	126	1010110	V	118	0x76	166	1110110	v
23	0x17	027	0010111	ETB	55	0x37	067	0110111	7	87	0x57	127	1010111	W	119	0x77	167	1110111	w
24	0x18	030	0011000	CAN	56	0x38	070	0111000	8	88	0x58	130	1011000	X	120	0x78	170	1111000	x
25	0x19	031	0011001	EM	57	0x39	071	0111001	9	89	0x59	131	1011001	Y	121	0x79	171	1111001	y
26	0x1A	032	0011010	SUB	58	0x3A	072	0111010	:	90	0x5A	132	1011010	Z	122	0x7A	172	1111010	z
27	0x1B	033	0011011	ESC	59	0x3B	073	0111011	;	91	0x5B	133	1011011	[	123	0x7B	173	1111011	{
28	0x1C	034	0011100	FS	60	0x3C	074	0111100	<	92	0x5C	134	1011100	\	124	0x7C	174	1111100	|
29	0x1D	035	0011101	GS	61	0x3D	075	0111101	=	93	0x5D	135	1011101	]	125	0x7D	175	1111101	}
30	0x1E	036	0011110	RS	62	0x3E	076	0111110	>	94	0x5E	136	1011110	^	126	0x7E	176	1111110	~
31	0x1F	037	0011111	US	63	0x3F	077	0111111	?	95	0x5F	137	1011111	_	127	0x7F	177	1111111	DEL

Reference Data Type

- Non-primitive data types are created based on the needs of the programmer.
- The non-primitive default value is **null**
- Declaration of this data type is almost the same as declaration of primitive data types.
- Non-primitive data types are preceded by uppercase letters

Reference Data Type

- The distinctive feature of reference data types is their ability to hold **multiple values**.
- In primitive data types, only 1 value can be accommodated.

Primitive Type:

- int x = 9; (there is only 1 value - number 9)
- char myLetter = "h"; (there is only 1 value - letter h)

Reference Type:

- String script = "I Learn Java"; (there are 12 values, including spaces)
- int [] list = {1, 4, 9, 16, 25, 36, 49}; (there are 7 integer values)

Operator

- Operators are symbols commonly used in writing a statement in any programming language. The operator will perform an operation on the operand according to its function.
- Examples of operations include addition, subtraction, division and so on.

```
3 + 8 * 4
3  8  4  is operand
 +  * is Operator
```

Operator Types

1. Arithmetic Operators
2. Increment and Decrement Operators
3. Assignment Operators
4. Relational Operators
5. Logical Operators
6. Bitwise Operators

1. Arithmetic Operators

Arithmetic operator is an operator that functions for arithmetic operations.

Operator	Meaning	Example	Result
+	Addition	10 + 2	12
-	Subtraction	10 – 2	8
*	Multiplication	10 * 2	20
/	Division	10 / 2	5
%	Modulus (remainder)	10 % 2	0

```java
public class arithmetic {
    public static void main(String[] args) {
        int a = 15;
        int b = 10;
        System.out.println("Arithmetic Operator");
        System.out.println("The first number: " + a);
        System.out.println("The second number: " + b);
        System.out.println("a + b = " + (a + b));
        System.out.println("a - b = " + (a - b));
        System.out.println("a / b = " + (a / b));
        System.out.println("a * b = " + (a * b));
        System.out.println("a % b = " + (a % b));
    }
}
```

```
run:
Arithmetic Operator
The first number: 15
The second number: 10
a + b = 25
a - b = 5
a / b = 1
a * b = 150
a % b = 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Increment and Decrement Operators

The Increment and Decrement operators are used to increase or decrease an integer value by one unit, and can only be used on variables.

Name	Operator	Meaning
Pre increment operator	++x	Add 1 to x, then use new value of x
Post increment operator	x++	Use value of x, then add 1 to x
Pre decrement operator	--x	Take 1 from x, then use new value of x
Post decrement operator	x--	Use value of x, then take 1 from x

```java
public class OperatorIncrementdanDecrement {
    public static void main(String[] args) {
        int i = 1;
    //increment
        System.out.println("i : " + i);
        System.out.println("++i : " + ++i);
        System.out.println("i++ : " + i++);
    //decrement
        System.out.println("--i : " + --i);
        System.out.println("i--: " + i--);
        System.out.println("i : " + i);
```

variabeltipedataoperator.OperatorIncrementdanDecrement > main >

out - variabeltipedataoperator (run)

```
run:
i : 1
++i : 2
i++ : 2
--i : 2
i--: 2
i : 1
```

3. Assignment Operators

Java assignment operators are used to assign a value to a variable. The assignment operator is simply '=',

Operator	Meaning
=	**Equal**: It assigns value to the variable
+=	**Add Equal**: It updated the value of the variable after addition
-=	**Subtract Equal**: It updated the value of the variable after subtraction
*=	**Multiply Equal**: It updated the value of the variable after multiplication
/=	**Division Equal**: It updated the value of the variable after division
%=	**Remainder Equal**: It updated the value of the variable after remainder

3. Assignment Operators

- a = a + 5; can be shortened to a + = 5;
- b = b – 5; can be shortened to b - = 1;
- c = c * 5; can be shortened to c * = 3;
- d = d / 5; can be shortened to d / = 5;
- e = e% 5; can be shortened to e% = 5;

```java
public class operatorassigment2 {
    public static void main(String[] args) {
    int a = 10;
    // Demo operator assignment
    a += 5;
    System.out.println("value a [10] += 5 = " + a);

    int b = 10;
    b -= 5;
    System.out.println("value b [10] -= 5 = " + b);

    int c = 10;
    c *= 5;
    System.out.println("value c [10] *= 5 = " + c);

    int d = 10;
    d /= 5;
    System.out.println("value d [10] /= 5 = " + d);

    int e = 10;
    e %= 5;
    System.out.println("value e [10] %= 5 = " + e);
  }
```

Output - variabeltipedataoperator (run)

```
run:
value a [10] += 5 = 15
value b [10] -= 5 = 5
value c [10] *= 5 = 50
value d [10] /= 5 = 2
value e [10] %= 5 = 0
BUILD SUCCESSFUL (total time: 0 seconds)
```

4. Relational Operators

Relational operators in Java are used to generate boolean values which are often used to control the flow of a program.

Operator	Meaning	Example	Result
<	Less than	5 < 2	False
>	Greater than	7 > 2	True
<=	Less than or equal to	9 <= 9	True
>=	Greater than or equal to	4 >= 1	True
==	Equal to	6 == 3	False
!=	Not equal to	6 !=8	True

```java
public class relational {
    public static void main(String[] args) {
        int x, y, z;
        x = 100;
        y = 99;
        z = 99;
        System.out.println("The value of x is = " + x);
        System.out.println("The value of y is = " + y);
        System.out.println("The value of z is = " + z);
        System.out.println("The result of y == z is " + (y == z));
        System.out.println("The result of y < z is " + (y < z));
        System.out.println("The result of x != y is " + (x != y));
        System.out.println("The result of x >= z is " + (x >= z));
    }
}
```

```
run:
The value of x is = 100
The value of y is = 99
The value of z is = 99
The result of y == z is true
The result of y < z is false
The result of x != y is true
The result of x >= z is true
BUILD SUCCESSFUL (total time: 0 seconds)
```

5. Logical Operators

This operator is used for logical expressions that return boolean values. The operators used are AND (&&), OR (||) and NOT (!).

Operator	Meaning	Example	Result
&&	AND	x = 6 y = 3 (x < 10) && (y > 1)	True
||	OR	x = 6 y = 3 (x == 5) || (y == 9)	False
!	NOT	x = 6 y = 3 !(x == y)	True

```java
public class operatorlogika {
   public static void main(String[] args) {
         boolean _true = true;
         boolean _false = false;

         System.out.println("Relation with OR (||)");
         System.out.println("_true || _true : " +(_true||_true));
         System.out.println("_true || _false : " +(_true||_false));
         System.out.println("_false || _true : " +(_false||_true));
         System.out.println("_false || _false : " +(_false||_false));

         System.out.println("Relation with AND (&&)");
         System.out.println("_true && _true : " +(_true&&_true));
         System.out.println("_true && _false : " +(_true&&_false));
         System.out.println("_false && _true : " +(_false&&_true));
         System.out.println("_false && _false : " +(_false&&_false));
         System.out.println("Relation with NOT (!)");
         System.out.println("inverse of (NOT) _true is: " +!_true);
         System.out.println("inverse of (NOT) _false is: " +!_false);
   }
```

```
run:
Relation with OR (||)
_true || _true : true
_true || _false : true
_false || _true : true
_false || _false : false
Relation with AND (&&)
_true && _true : true
_true && _false : false
_false && _true : false
_false && _false : false
Relation with NOT (!)
inverse of (NOT) _true is: false
inverse of (NOT) _false is: true
BUILD SUCCESSFUL (total time: 1 second)
```

6. Bitwise Operators

- This operator is used to perform bit manipulation of a number
- Bitwise operator types:
  a. Bitwise OR (|)
  b. Bitwise AND (&)
  c. Bitwise XOR (^)
  d. Bitwise Complement (~)

6. Bitwise Operators

- Bitwise OR (|)

  The result of the bit is 1 when one of the bits is 1, otherwise it is 0.
- Example:

```
int a = 5;//0101
int b = 7;//0111
System.out.println(a|b);//output 7
//0101
//0111
//____
//0111 -> 7
```

6. Bitwise Operators

- Bitwise AND (&)

  The result of the bit is 1 when all the bits are 1, otherwise it is 0.
- Example:

```
int a = 5;//0101
int b = 7;//0111
System.out.println(a&b);//output 5
//0101
//0111
//____
//0101 -> 5
```

6. Bitwise Operators

- Bitwise XOR (^)

  Bit value is 1 when there are bits of 1 and 0, otherwise it is 0.
- Example:

```
int a = 5;//0101
int b = 7;//0111
System.out.println(a^b);//output 2
//0101
//0111
//____
//0010 -> 2
```

6. Bitwise Operators

- Bitwise Complement (~)

  The inverse bit value, when the bit value is 1 it produces 0 while the value 0 produces 1.

- Example:

```
int a = 5;//0101
System.out.println(~a);//output -6
//0101
//____
//1010 -> 10
```

~n = -(n+1)
~(-n) = (n-1)

Shift Operator

- The shift operator is used to perform bit shifts, either right or left.
- Type of shift operator:
  a. Shift right (>>)
  b. Shift left (<<)

Shift Operator

- Shift right operator (>>)

  Shift n bits to the right. n is the number of shifts
- Example:

```
int a = 11;//1011
System.out.println(a>>2);//output 2
//1011
//____
//0010 -> 2
```

Shift Operator

- Shift left operator (<<)

  Shift bits to the left by n, n is the number of shifts
- Example:

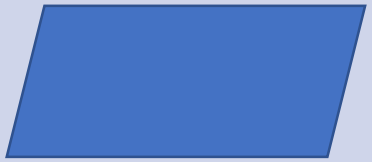
```
int a = 11;//1011
int b = a<<2;
System.out.println(b);//output 12
//1011
//____
//1100 -> 12
```

Flowchart

Flowchart

- A flowchart is a chart with certain symbols used to explain the sequence of processes and relationships between other processes in a program.

Flowchart Symbols

Symbol	Name	Description
	Terminator	Symbol for the **start** and **end** in a process
	Preparation	Symbol for preparing storage for use
	Input-output	Symbols representing input and output processes

Flowchart Symbols

Symbol	Name	Description
	Process	Symbol denoting processing by computer
	Flow line	Symbol to indicate the direction of the process flow

Example

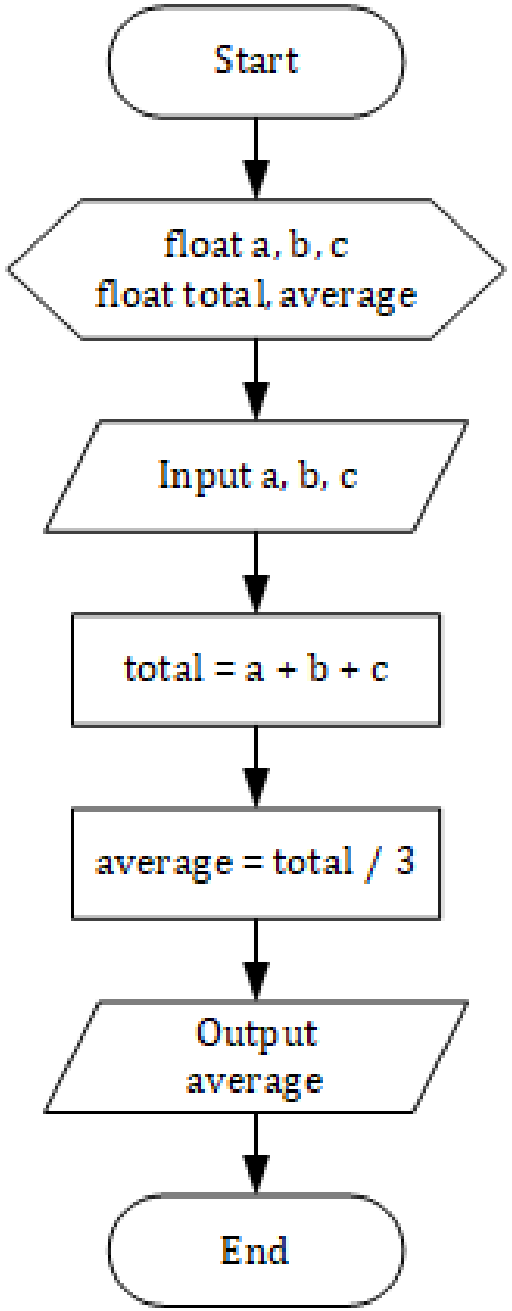
Problem:

Create an algorithm to find the average of three numbers!

Example

Answer:

- **Input**: three numbers (a, b, c)
- **Process:**
  1. Enter a, b, c
  2. Total = a + b + c
  3. Average = Total / 3
  4. Get average
- **Output**: average



Assignment

1. Two years ago, Harry was 6 times the age of Laras. Eighteen years later Harry will be twice as old as Laras. Create an algorithm and flowchart to determine the age of Harry and Laras!
2. Calculate the final score of students from the average score of assignment, quiz, midterm exam, and final exam! *(Assumption: there is only 1 assignment score and 1 quiz score)*