Variables, Data Types, Operators, and Input-Output

Programming Fundamentals Teaching Team 2023

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)

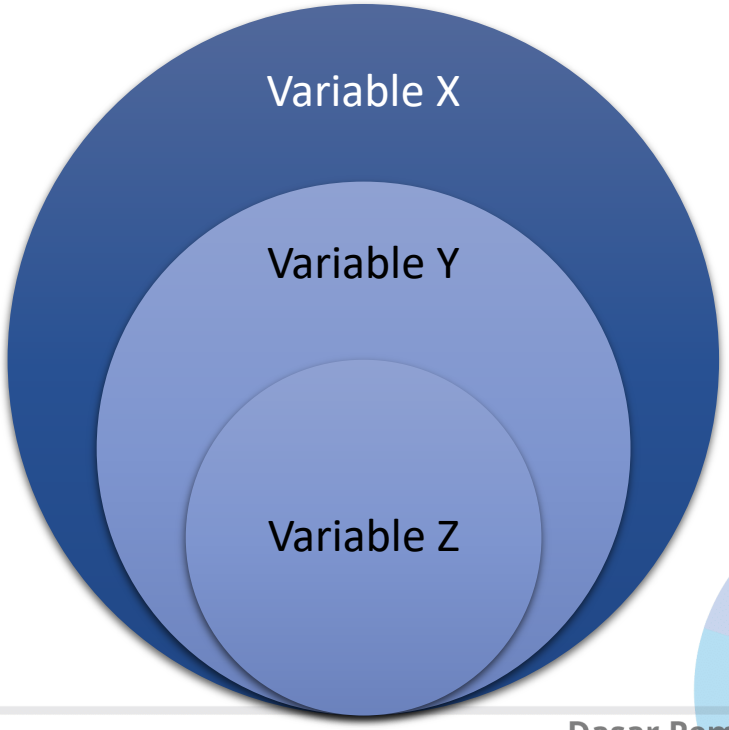
Variable

- 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 = 34;
```

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);
```

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.

USASCII code chart

b_7 b_6 b_5 →					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
Bits b_4	b_3	b_2	b_1	Column → / Row ↓	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	–	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	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	Usage	Equivalent To
+=	Op1 += Op2	Op1 = Op1 + Op2
-=	Op1 -= Op2	Op1 = Op1 - Op2
*=	Op1 *= Op2	Op1 = Op1 * Op2
/=	Op1 /= Op2	Op1 = Op1 / Op2
%=	Op1 %= Op2	Op1 = Op1 % Op2
&=	Op1 &= Op2	Op1 = Op1 & Op2
|=	Op1 |= Op2	Op1 = Op1 | Op2
^=	Op1 ^= Op2	Op1 = Op1 ^ Op2

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)

Use of Input in Java

- To read input from the keyboard, use the Scanner library, which is imported into the Java program.
- The way to do this is to write the command **import java.util.Scanner** in the top line of the program code to be created.
- Next, write the following scanner declaration command in the **main()** function:

```
Scanner sc = new Scanner(System.in);
```

- Next, depending on the type of input to be entered, it will be an integer (int), a comma number (float/double), or a character (String).
  1. If the input is an integer, then the command is **nextInt();**
  2. If the input is a floating point, then the command is **nextFloat();**
  3. If the input is text, then the command is **nextLine();**

Displays output in Java

To display output to the screen, there are several ways:

- **System.out.print(“Hello world”);** This command will display the words Hello world on the screen, or whatever we write in quotation marks.
- **System.out.println("Hello world");** This command will display the words Hello world on the screen, or whatever we write in quotation marks, as well as giving a line change command at the end of the word/sentence.
- **System.out.println(length);** This command will display the contents of the variable *length* to the screen. Note that to display the contents of a variable, there is no need to use quotation marks (“).
- **System.out.println("Rectangle length: " + length);** This command will display the sentence “Rectangle length: " then connect the contents of the *length* variable to the screen. Note that to connect sentences with variable content, use the plus sign (+).

Example of Case Study 1

Mr. Adi has a rectangular garden. Pak Adi wants to make a wooden fence to surround the garden.

Before creating a program to help Mr. Adi calculate the perimeter of his garden, help Mr. Adi to identify variables and data types along with the algorithm!

Example of Case Study 1 (Answer)

A. Determine the Algorithm

Input: length, width

Output: perimeter

Process:

1. input length, width **Line 10, 11**
2. perimeter = 2 x (length + width) **Line 12**
3. Output perimeter **Line 13**

B. Identify variables and data types based on algorithms

Line 6, 7, 8

Variables	Data types
length	int
width	int
perimeter	int

```
1  import java.util.Scanner;
2
3  public class Example1 {
4      public static void main(String[] args) {
5          Scanner input = new Scanner(System.in);
6          int length;
7          int width;
8          int perimeter;
9
10         length = input.nextInt();
11         width = input.nextInt();
12         perimeter = 2 * (length + width);
13         System.out.println(perimeter);
14     }
15 }
```

Example of Case Study 2

Mrs. Dina is one of ABC bank customers who saved Rp. 5 million. The bank provides interest of 2% every year. Mrs. Dina saved for 5 years. How much interest and savings can you take now?

Example of Case Study 2 (Answer)

A. Determine the Algorithm

Input: initial savings amount, savings period

Output: interest, final savings amount

Other data: interest percentage = 0.02

Process:

1. Input the initial savings amount, savings period
2. Calculate interest = savings period * interest percentage * initial savings amount
3. Calculate the final savings amount = interest + initial savings amount
4. Interest output and final savings amount

B. Identify variables and data types based on algorithms

Variables	Data types
init_sav_amount	int
sav_period	int
final_sav_amount	double
interest	double
interest_percent	double

Example of Case Study 2 (Answer)

```
1   import java.util.Scanner;
2
3   public class Example2 {
        Run | Debug
4       public static void main(String[] args) {
5           Scanner input = new Scanner(System.in);
6           int init_sav_amount, sav_period;
7           double interest_percent = 0.02, interest, final_sav_amount;
8
9           System.out.println(x:"Enter your initial savings amount: ");
10          init_sav_amount = input.nextInt();
11          System.out.println(x:"Enter your savings period: ");
12          sav_period = input.nextInt();
13
14          interest = sav_period * interest_percent * init_sav_amount;
15          final_sav_amount = interest + init_sav_amount;
16
17          System.out.println("Interest amount: " + interest);
18          System.out.println("Your final savings amount: " + final_sav_amount);
19      }
20  }
```

Line 6, 7: variable and data type declarations

Line 10, 12: input

Line 14, 15: process

Line 17, 18: output

Assignment

1. Describe the scope of your final project according to the topic of each group taken.
2. Based on the final project topic scope description, identify:
   a. Inputs and outputs
   b. Process