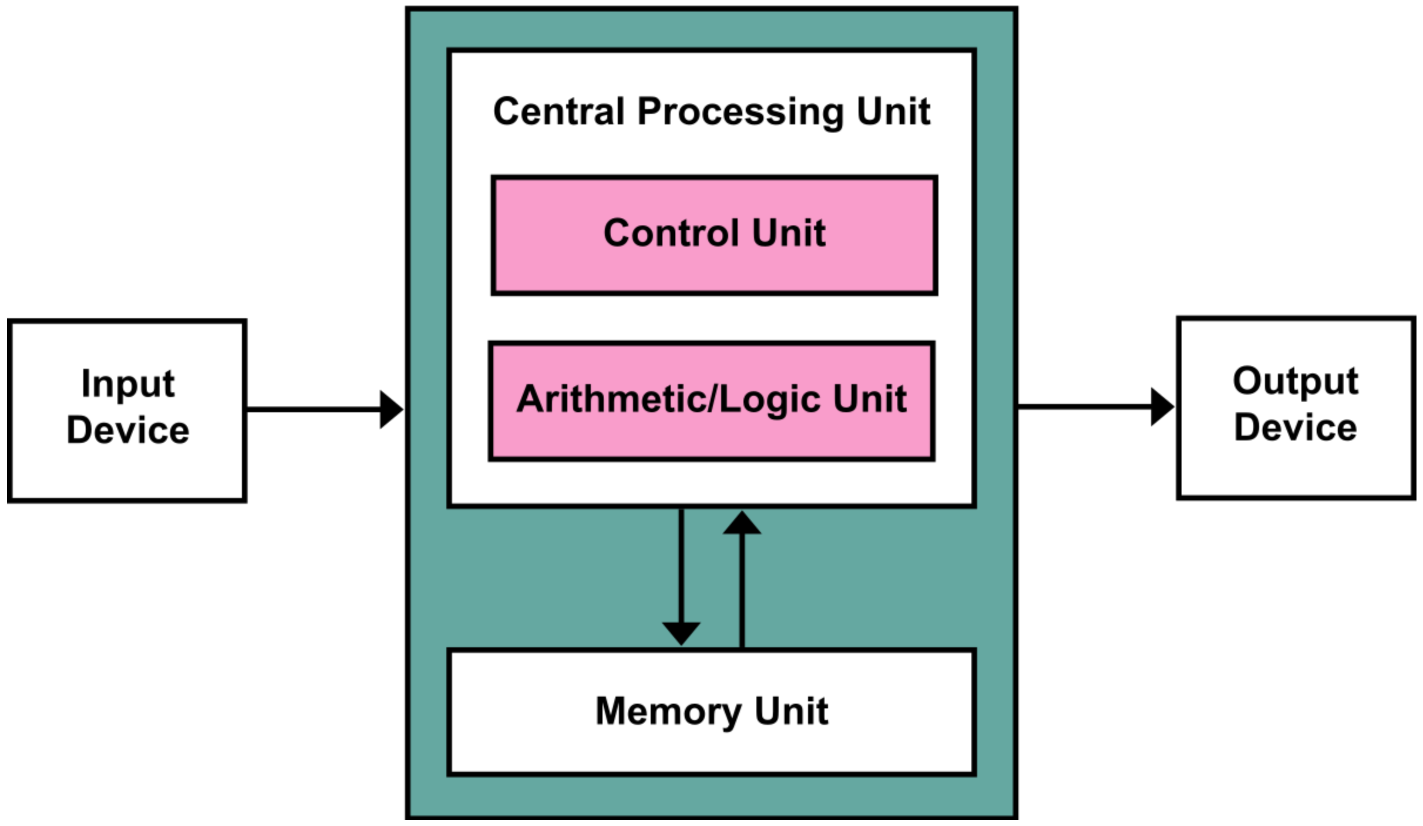
Variables and Data Types

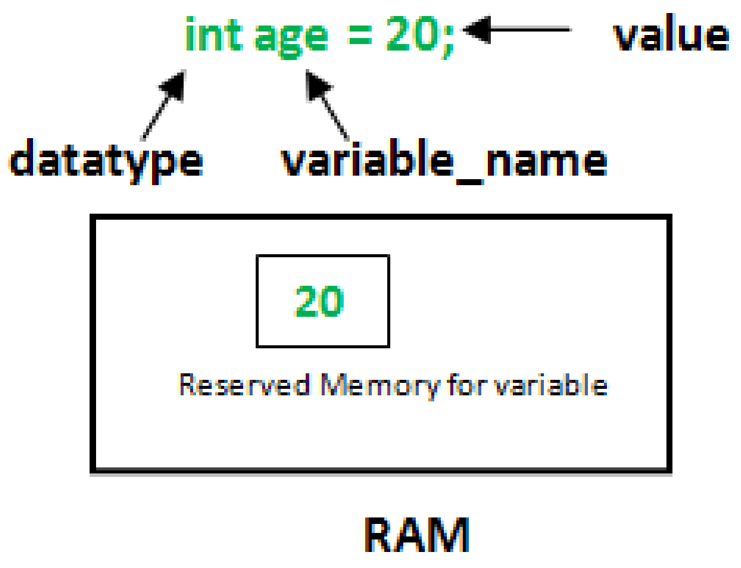
Java Programming for Absolute Beginners Phase I

A Quick Note on Memory



What is a Variable?

- Think of a variable as a placeholder for data in memory.
- All variables have a name, a value, and a **data type**.
- Remember that in memory there only exists one’s and zero’s, so the meaning of the value is determined by the **data type** of the variable.
- There are two types of data types in Java: **primitive** and **non-primitive**.



Java Primitive Types

Type	Size	Range	Default
boolean	1 bit	true or false	false
byte	8 bits	[-128, 127]	0
short	16 bits	[-32,768, 32,767]	0
char	16 bits	['\u0000', '\uffff'] or [0, 65535]	'\u0000'
int	32 bits	[-2,147,483,648 to 2,147,483,647]	0
long	64 bits	[-2^{63}, $2^{63}-1$]	0
float	32 bits	32-bit IEEE 754 floating-point	0.0
double	64 bits	64-bit IEEE 754 floating-point	0.0

Primitive Data Types

Declaring a Variable in Java

- Java is a **strongly typed** language, meaning a variable must be declared with a data type, and that data type cannot be changed.
- Declaring a variable will allocate space in RAM for that variable. The amount of bits allocated depends on the data type.
- The syntax to declare a variable in Java is as follows:
  - **<data_type> <variable_name>;**
    - Example: int i;
    - Example: double x;
    - Example: int age;
    - Example: String hello;

Variable Name Rules

- We will be calling referring to variable names as **identifiers**, because these rules apply to more than just variables (more on that later).
- Variables are given meaningful names to make programs expressive, but there are some limitations on how we can name our variables.
- An identifier must:
  - Start with either a letter (a-z | A-Z), a dollar sign ($), or an underscore (_)
  - May contain numbers after the first character
  - May be any amount of characters after first character
  - Must be unique in its scope (more on that later)
  - May not contain spaces or special characters such as #, +, -, *, /, !, ., ,, etc.
  - Are case sensitive
  - May not match a reserved keyword (words like: public, int, static, final, void, class, etc)
    - https://en.wikipedia.org/wiki/List_of_Java_keywords
- https://www.geeksforgeeks.org/java-naming-conventions/

Giving variables values

- Giving a variable a value is known as **assignment**.
- Assignment is achieved using an **assignment operator**, which in Java is the equals sign (=).
- The first time a variable is assigned is known as **initializing**.
  - Example:
    - int age;
    - age = 20;
  - Example 2: int age = 20;
- Note how you can initialize a variable at the same time as the declaration.
- To reassign a variable, simply use the assignment operator again.
- The assignment operation is executed right to left, meaning the value on the right hand side (RHS) is evaluated and then the result is stored in the variable on the left hand side (LHS)
- On the LHS of an assignment can only be a singular variable name, while on the RHS can be either a **literal**, or an **expression**.

The RHS of an Assignment: Literals

- The RHS of an assignment can contain either a literal, or an expression.
- A literal is basically a direct value written into your program. There different types of literals in Java which represent different data.
  - Numeric Literals:
    - Integer Literals: An integer literal is a positive whole number that can be written in the following bases:
      - Base 10: Base 10 literals are numbers that we are used to, like 1, and 2, and 50.
      - Base 8: Base 8 literals are prefixed by a leading 0, so the number 10 in a base 8 literal would be 012
      - Base 16: Base 16 (or hexadecimal) literals are written with 0x as the prefix, so we can have values like 0xFF.
      - Base 2: Base 2 (binary) literals are prefixed with 0b, so the number 12 would be 0b1100
      - (Underscores may be used in integer literals)
    - Floating Point Literals:
      - Floats are how we use real numbers in Java they can either be written in standard notation or in scientific notation with an E. For example, we can have either 120.5f or 1.205E2f. Note all floating point literals must end in the letter f, without the f, the literal would be treated as a double.

More Literals

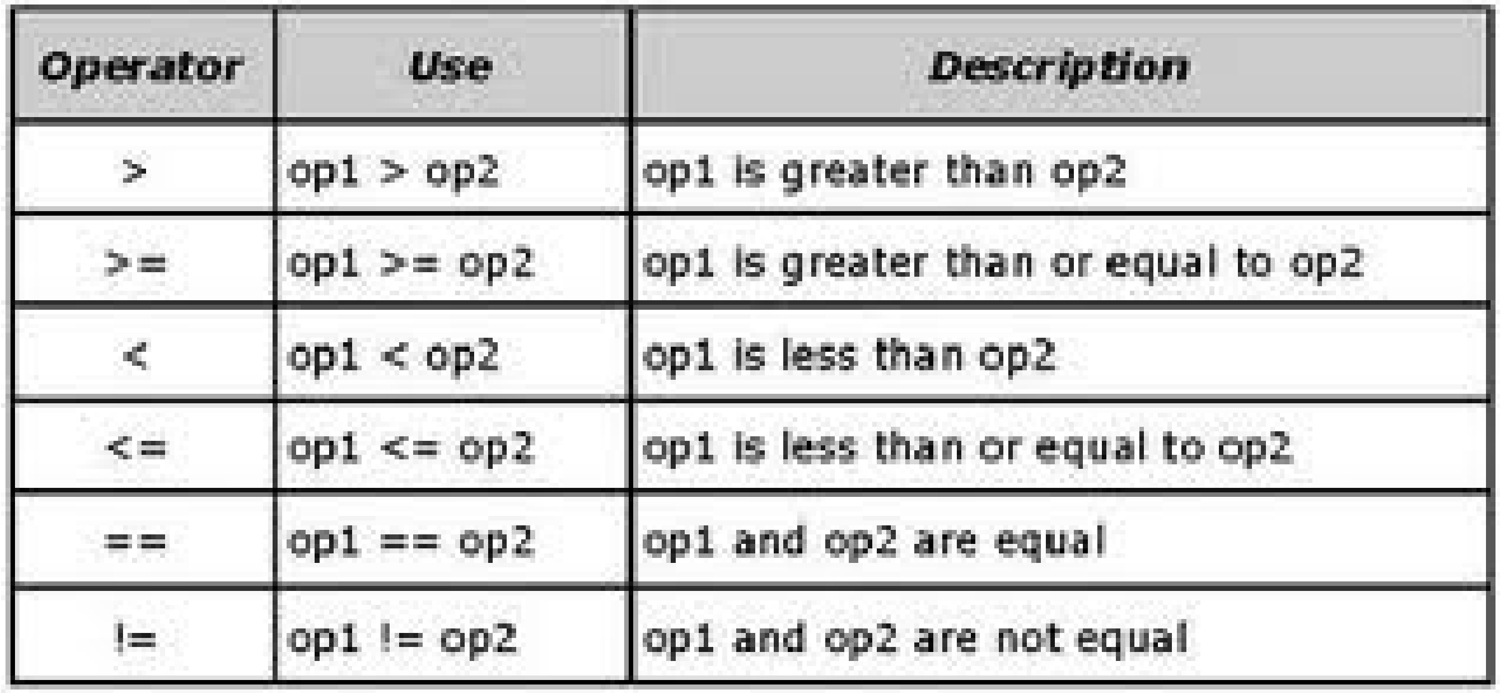
- Boolean Literals:
  - Either true or false
- Character Literals:
  - Characters are denoted by a single character surrounded by single quotes.
    - Example: ‘A’
- String Literals
  - Strings are the only non-primitive that can be used with literals.
  - A string literal is a series of characters enclosed in double quotation marks.
    - Example “hello!”
- Note: There are such things as “escape characters” we can use in our character and string literals, read up on them here: https://docs.oracle.com/javase/tutorial/java/data/characters.html

The RHS of an Assignment: Expressions

- The other option for an RHS of an assignment is an **expression**.
- An expression is a combination of literals, variables, and **operators** that produces a value.
  - Example: x + y * 3
- Today we will only be looking at arithmetic expressions.
- An operator is a symbol used to represent a manipulation of data.
- There are three types of operators:
  - Arithmetic Operators (adding, subtracting, etc.)
  - Relational Operators (is equal to, is greater than, etc.)
  - Logical Operators (and, or, not, etc.)
  - Assignment Operators
  - Increment/Decrement Operators
  - Bitwise Operators (these are not in the scope of this class)

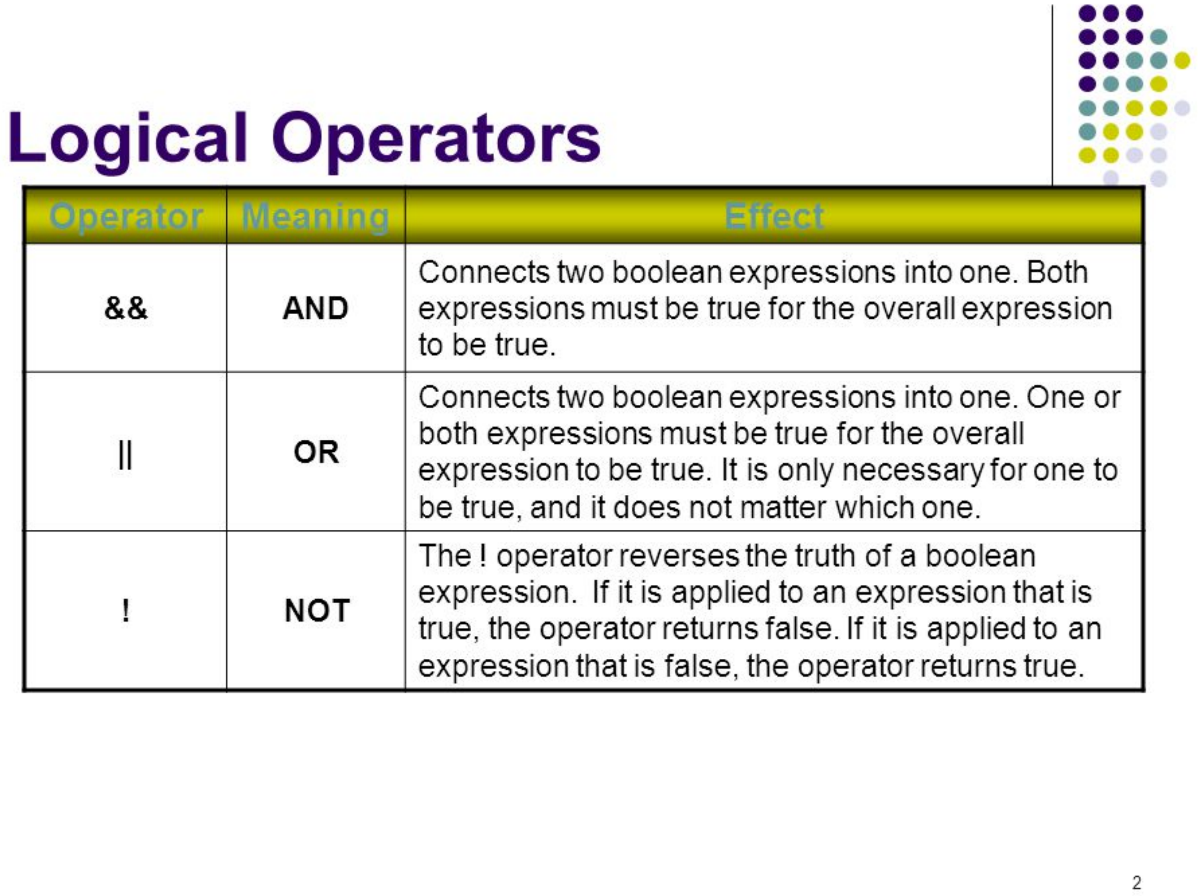
Arithmetic Operators

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



Operator	Use	Description
>	op1 > op2	op1 is greater than op2
>=	op1 >= op2	op1 is greater than or equal to op2
<	op1 < op2	op1 is less than op2
<=	op1 <= op2	op1 is less than or equal to op2
==	op1 == op2	op1 and op2 are equal
!=	op1 != op2	op1 and op2 are not equal

Relational Operators

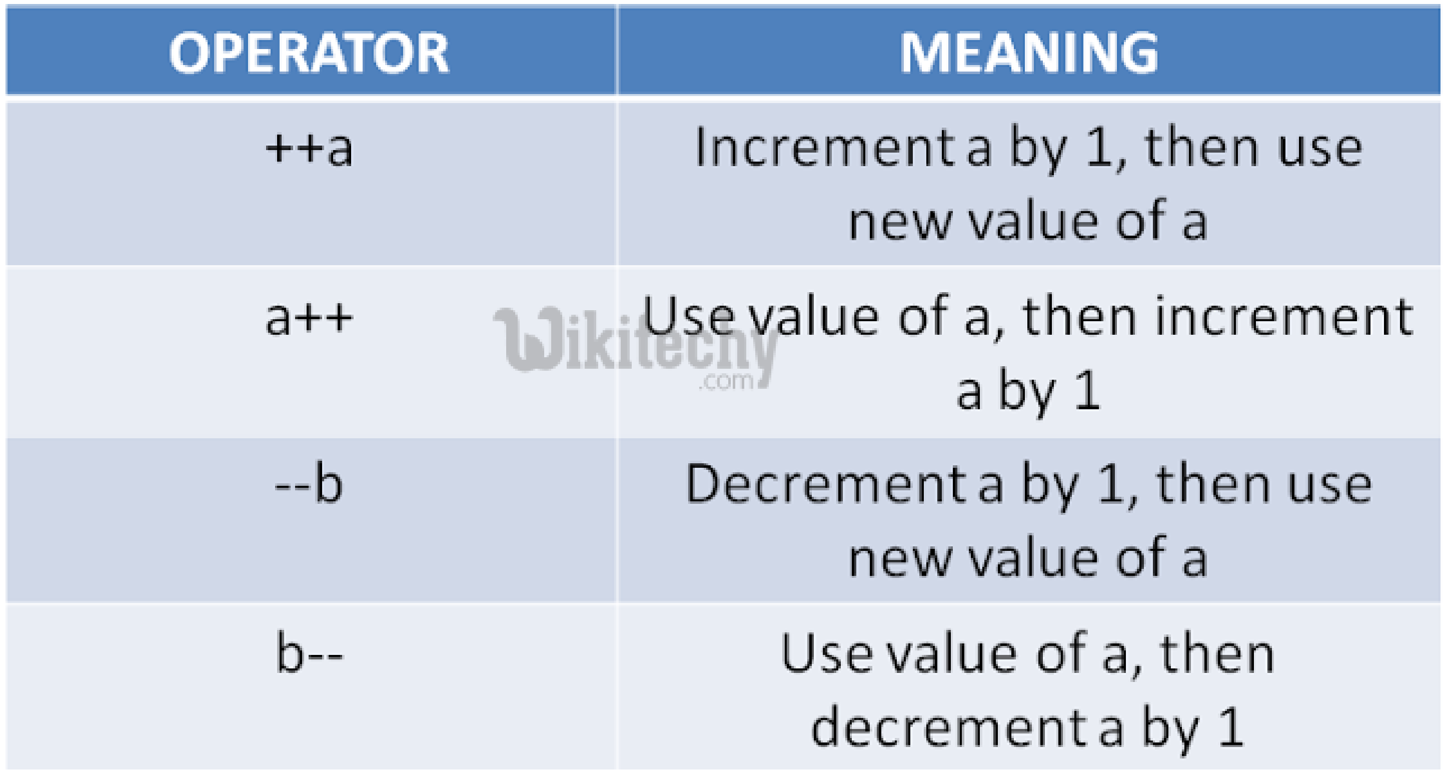


Logical Operators

Operator	Meaning	Effect
&&	AND	Connects two boolean expressions into one. Both expressions must be true for the overall expression to be true.
||	OR	Connects two boolean expressions into one. One or both expressions must be true for the overall expression to be true. It is only necessary for one to be true, and it does not matter which one.
!	NOT	The ! operator reverses the truth of a boolean expression. If it is applied to an expression that is true, the operator returns false. If it is applied to an expression that is false, the operator returns true.

Assignment Operators

Operator	Description	
=	assigns value from right side operands to left side operand	a=b
+=	adds right operand to the left operand and assign the result to left operand	a+=b is same as a=a+b
-=	subtracts right operand from the left operand and assign the result to left operand	a-=b is same as a=a-b
*=	mutiply left operand with the right operand and assign the result to left operand	a*=b is same as a=a*b
/=	divides left operand with the right operand and assign the result to left operand	a/=b is same as a=a/b
%=	calculate modulus using two operands and assign the result to left operand	a%=b is same as a=a%b

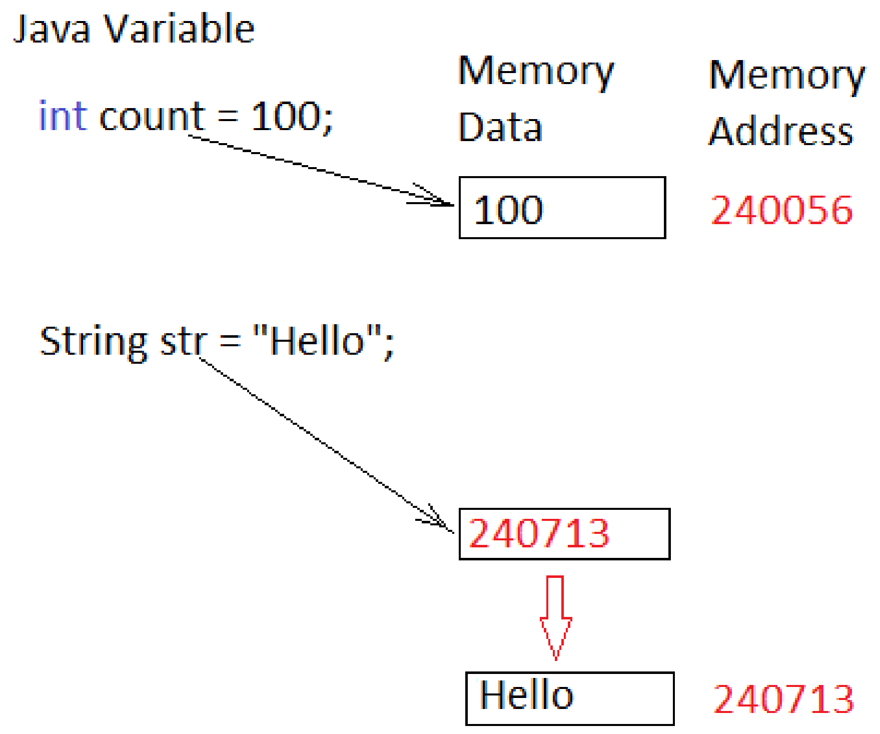


OPERATOR	MEANING
++a	Increment a by 1, then use new value of a
a++	Use value of a, then increment a by 1
--b	Decrement a by 1, then use new value of a
b--	Use value of a, then decrement a by 1

Increment and Decrement Operators

Strings

- Think of a string as a list of chars.
- Strings are non-primitive, making them different from the variables we have been looking at.
- The value at the variable in memory is not the string itself, but a **reference** to the string.
- String literals are double quotes surrounding characters.
- Strings are **immutable**, meaning their values cannot be changed, to update the value of a string, a new one must be created and the reference must be reassigned.



String Concatenation

- The String Concatenation Operator (+) is used to combine two strings together.
- Other data can also concatenated with a string, which is perfect for printing data to the console.
- Be careful, the concatenation operator has the same precedence as the addition operator, which can yield strange results
  - Example: `String s=50+30+"Hello"+40+40;`
    - s has a value of 80Hello4040
- Using parentheses can force desired results.

Any Questions?