# Lesson:

# Java Variables and Data types


PW SKILLS

# List of Concepts Involved:

- Statically typed vs Dynamically types PL
- Variables and data types
- Naming Convention
- Identifiers
- Operators in Java
- Incrementation and Decrementation

A computer program/code consists of various components viz. variables, data types, identifiers, keywords, etc which help us to build a successful program. Let us learn each one of them in detail and then move to our first program.

# Topic: Statically typed vs Dynamically typed

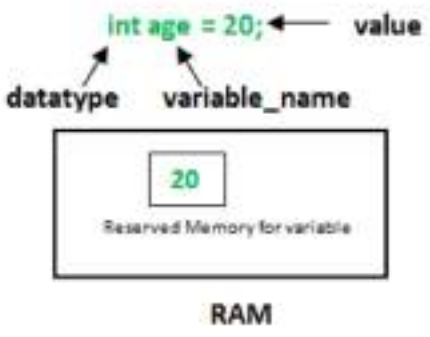
**Statically typed:** if the memory of the variable is given during the compilation time itself then such types of programming languages are called as "Statically typed".
Eg: C,C++,Java

**Dynamically typed:** if the memory of the variable is given during the execution time itself then such types of programming languages are called as "dynamically typed".
Eg: Python,PHP,JavaScript

# Topic: Variables

- A variable is the title of a reserved region allocated in memory. In other words, it may be referred to as the name of a memory location.
- It is a container that holds the value while the Java program is executed.
- Each variable should be given a unique name to indicate the storage area.
- A variable is assigned with a data type(we will learn about it after this topic).

**Syntax for Declaring a Variable:**
Type variable_name [= value];
The variable_name is the name of a variable. We can initialise the variable by specifying an equal sign and a value (initialization i.e. assigning an initial value, is optional). However, the compiler never assigns a default value to an uninitialized local variable in Java.



Here, rate is an int data type variable with the value 40 assigned to it.
In the example above, the variable can only hold integer values, as indicated by the int data type.

Here, we assigned a value to the variable during the declaration process. However, as stated before, it is optional.

Variables can be declared and assigned separately. Example,

```
int rate;
rate = 40;
```

**Changing values of variables**
Interestingly, a variable's value can also be changed in the program. Look at the example below :

```
int rate = 50;
System.out.println(rate);// 50
rate = 60;
System.out.println(rate);// 60
```

Initially, the value of rate was 50 but it has changed to 60 after the last update, rate=60.

**Naming Conventions for variables in Java**
Like us, all java components are identified with their names. There are a few points to remember while naming the variable. They are as follows -

- Variable names should not begin with a number.For example
  ```
  int 2var; // 2var is an invalid variable .
  ```

- White Spaces are not permitted in variable names. For example,
  ```
  int cricket score; // invalid variables.
  ```
  There is a gap/whitespace between cricket and score.

- A java keyword (reserved word) cannot be used as a variable name. For example, int float; is an invalid expression as float is a predefined keyword(we will learn about them) in java.

  As per the latest coding practices, for variable names with more than one word the first word has all lowercase letters and the first letter of subsequent words are capitalised. For example, cricketScore, codePracticeProgram etc. This type of format is called camelcase.

- While creating variables, it's preferable to give them meaningful names like- ‘age’, ‘earning’, ‘value’ etc. for instance, makes much more sense than variable names like a, e, and v.

- We use all lowercase letters when creating a one-word variable name. It's preferable(and in practice) to use physics rather than PHYSICS or pHYSICS.

# Topic 3: Identifiers

An identifier is a name given to a package, class, interface, method, or variable. All identifiers must have different names.
In Java, there are a few points to remember while dealing with identifiers :

- **Rule 1** - All identifiers should begin with a letter (A to Z or a to z), $ and _ and must be unique.
- **Rule 2** - After the first character/letter, identifiers can have any combination of characters.
- **Rule 3** - A keyword cannot be used as an identifier.
- **Rule 4** - The identifiers are case-sensitive.
- **Rule 5** - Whitespaces are not permitted.
- Examples of legal identifiers: rank, $name, _rate, __2_mark.
- Examples of illegal identifiers: 102pqr, -name.

These variables, identifiers etc. consume memory units. Before proceeding ahead, let us have a look at the memory unit concept too. Here, we will only focus on the relevant concept of memory.

**Basic Memory units:**
It refers to the amount of memory or storage used to measure data.
Basic memory units are:

**1. Bit**
A bit (binary digit 0 or 1) is the smallest unit of data that a computer can process and store
Symbols 0 and 1 are known as bits.Here, 0 indicates the passive state of signal and 1 indicates the active state of signal.
At a time, a bit can store only one value i.e 0 or 1. To have a greater range of value, we combine multiple bits.

**2. Byte**
A byte is a unit of memory/data that is equal to 8 bits.
You may think of a byte as one letter. For example, the letter 'f' is one byte or eight bits.
The bigger units are :

**3. Kilobyte**
A Kilobyte is a unit of memory data equal to 1024 bytes.

**4. Megabyte**
A Megabyte is a unit of memory data equal to 1024 kilobytes.

**5. Gigabyte**
A Gigabyte is a unit of memory data equal to1024 Megabytes.

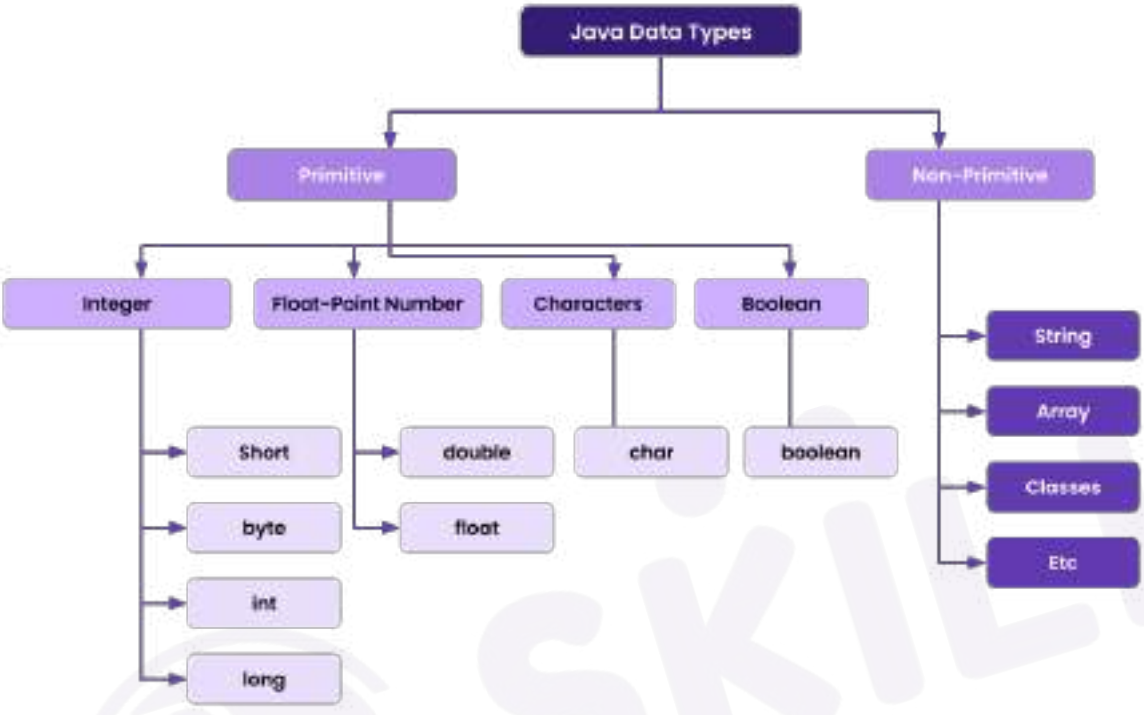
Lets us now move to the most important concept - data type

# Topic: Data Types

Data types specify the different sizes and values that can be stored in the variable. Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory. Hence, by assigning different data types to variables, we can store integers, decimals, or characters in these variables.

There are two types of data types in Java:

1. **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.
2. **Non-primitive data types:** Classes, Strings, Interfaces, and Arrays are examples of non-primitive data types.




# Primitive data types

A primitive type is predefined by the language and is named by a reserved keyword.

**1. Boolean Type**

- The Boolean data type can have two values– true or false and hence are typically used in true/false situations.

**For example,**

```
Boolean flag=true;
```

**2. Byte Type**

- Values for the byte data type range from -128 to 127 (8-bit signed two's complement integer, you will know more about it once we move to programs and applications).
- A byte type is used in place of an int to save memory when it is certain that the value of a variable will be between -128 and 127.

**For example,**

```
byte range=105;
```

**3. Short Type**

- The short data type can have values ranging from -32768 to 32767 (16-bit signed two's complement integer).
- If the value of a variable is certain to be between -32768 and 32767, short is used in place of other integer data types (int, long).

**For example,**

```
short loss=-50;
```

**4. Int Type**

- Values for the int data type range from $2^{31}$ to $2^{31}$-1(32-bit signed two's complement integer, you will know about it as we move to programs)
- In Java SE 8 and later, you can use the int data type to represent an unsigned 32-bit integer, which has a minimum value of 0 and a maximum value of $2^{32}$-1.

**For example,**

```
int profit=5000;
```

**5. Long Type**

- Values for the long data type range from - $2^{63}$ to $2^{63}$-1 (64-bit signed two's complement integer).
- You can use an unsigned 64-bit integer with a minimum value of 0 and a maximum value of $2^{64}$-1 if you're using Java 8 or later.

**For example:**

```
long profit=455559990;
```

**6. Double Type**

- The double data type is a 64-bit floating-point data type with double precision.
- It should never be used for exact values like currency.

**For example:**

```
double height=12.5;
```

**7. Float Type**

- The float data type is a 32-bit single-precision floating-point value. If you're curious, you can learn more about single-precision and double-precision floating-point.
- It should never be used for precise values like money.

**For example:**

```
float depth=-32.3f;
```

**8. Char Type**

- It's a Unicode (an international character encoding standard that provides a unique number for every character across languages and scripts) 16-bit characters.
- The char data type has a minimum value of 'u0000' (0) and a maximum value of 'uffff'.

**For example:**

```
char temp='a';
```

The **non-primitive** data types are a little advanced concepts which we will cover once we have mastered the primitives and are well versed with the programming principles of Java.

# Topic: Operators in Java

Operators in Java can be classified into 6 types:

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Assignment Operators
5. Unary Operators
6. Bitwise Operators

# 1.Java Arithmetic operators:

Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra. The following table lists the arithmetic operators:

Assume integer variable num1 holds 20 and variable num2 holds 30, then:

| Operator | Description and Example |
|---|---|
| + Addition | Adds values on either side of the operator.<br>*Example*: num1 + num2 will give 50 |
| - Subtraction | Subtracts right hand operand from left hand operand<br>*Example*: num1 - num2 will give -10 |
| * Multiplication | Multiplies values on either side of the operator<br>*Example*: num1 * num2 will give 600 |
| / Division | Divides left hand operand by right hand operand<br>*Example*: num1 / num2 will give 0.67 |
| % Modulus | Divides left hand operand by right operand and returns remainder<br>*Example*: num2 % num1 will give 10 |
| ++ Increment | Increases the value of operand by 1<br>*Example*: num2++ gives 31 |
| -- Decrement | Decreases the value of operand by 1<br>*Example*: num1-- gives 19 |

**Example:**

```
class Main {
      public static void main(String[] args) {

            // declare variables p and q
            int p = 20, q = 10;
            int result;

            // addition operator
            result=p+q;
            System.out.println(result);

            // subtraction operator
            System.out.println(p - q);

            // we can directly perform subtraction in print statement,no need to
            // use result variable here

                // multiplication operator
                System.out.println(p * q);

                // division operator
                System.out.println(p / q);

                // modulo operator
                System.out.println(p % q);
               }
        }
```

**Output:**

30
10
200
2
0

**Increment and Decrement Operators**

1. PreIncrement(++a)
2. PostIncrement(a++)
3. PreDecrement(--a)
4. PostDecrement(a–)

**Example:**

```
class Main {
      public static void main(String[] args) {
              int a = 5, int b = 6;
              int c = a++;//post increment
              int d = ++a;//pre increment
              int e = b--;//post decrement
              int f = -b;// pre decrement

              System.out.println(c);
              System.out.println(d);
              System.out.println(e);
              System.out.println(f);
             }
   }
```

**Output:**

```
5
7
6
4
```