Variables and Primitive Data Types

What Are Variables?

In programming, a variable is a container for storing data in memory. Think of it as a labeled box where you can keep values that you might want to reuse or modify later. Variables are essential for building programs because they let us work with and manipulate data.

Variables are defined by a name, which you choose, and a data type, which tells the computer what kind of information the variable will hold (like a number, a single character, a true/false value, and more).

Understanding Data Types

Data types specify the kind of data that can be stored in a variable. Different types require different amounts of memory and can represent different ranges of values. Let's dive into the basic, or “primitive,” data types commonly used in programming. We’ll use Java for our examples, but the ideas apply to other languages as well.

There are many “built-in” data types that come with most programming languages by default. These are called “primitive data types” or just “primitives”. Here are some common examples:

1. Integer (int):

- Purpose: Stores whole numbers (without decimals).
- Examples: 5, -3, 0.
- Memory: Uses 4 bytes (32 bits) of memory.
- Use Case: Perfect for counters, flags, or general arithmetic.

2. Character (char):

- Purpose: Stores a single Unicode character, like a letter.
- Examples: 'A', '9', '%', '愛'.
- Memory: Uses 2 bytes (16 bits) of memory.
- Use Case: For storing individual letters, digits, or symbols from any language.

3. Floating-Point (float):

- Purpose: Stores numbers with decimal points (fractional values).
- Examples: 3.14f, -0.75f, 1.0f.
- Memory: Uses 4 bytes (32 bits) of memory, with precision up to 6-7 significant decimal digits.
- Use Case: For when you need to store non-integer numbers, like temperatures or other measurements.

4. Boolean (boolean):

- Purpose: Stores a true or false value. That's it!
- Examples: true, false.
- Memory: Uses 1 bit (or the smallest possible unit). Usually a 0 represents "false" while 1 represents "true".
- Use Case: Vital for control flow (loops and branches).

Variable Declaration and Initialization

When you **declare** a variable, you're telling the computer to set aside space in memory for it. Declaration involves specifying the variable's data type and name. Here are some examples:

```
int age;

char grade;

boolean isStudent;
```

Initializing a variable means assigning it an initial value. This can happen in the same line as the declaration, or later in your code. In most programming languages we use the = sign to assign value to variables. The variable name goes on the left, and the value on the right.

You can either initialize a variable after defining it:

```
int score;

score = 95;
```

Or you can define and initialize variables on the same line:

```
int age = 25;

char grade = 'A';

boolean isStudent = true;
```

Here’s a quick example to show how variables might be used:

```
int age = 30;                  // Declare and initialize an integer

char initial = 'J';            // Declare and initialize a character

float height = 5.9f;           // Declare and initialize a float

boolean employed = true;       // Declare and initialize a boolean

System.out.println("Age: " + age);

System.out.println("Initial: " + initial);

System.out.println("Height: " + height);

System.out.println("Employed: " + employed);
```