**Java syntax rules, Variables, and Data Types**

**Java syntax rules:**

1. Case-Sensitive language.
2. Class name should start with a Capital letter.
3. File name = class name (public class)
4. Each statement ends with ;
5. Code blocks enclosed in { }

**Variables:**

-It is a container to store data in memory.

-Syntax: datatype variable\_name = value;

-Ex: int age = 21;

String name = “Mansi”;

**Types of Variables:**

1. **Local Variables:**

- These variables declared within a method, constructor or a block of code. (e.g. inside if statement or a for loop)

1. **Instance Variables:**

**-**These variables are declared within a class but outside of any method, constructor, or block.

**-**They belong to an instance(object) of the class, and each object has its own copy of instance variable.

1. **Static Variable:**

-These variables are declared within a class, similar to instance variables, but are marked with the static keyword.

**Data Types in Java:**

**Data type definition:** It is a java concept used to denote the types of data.

**There are two data types:**

1. Primitive data types
2. Non-Primitive data types
3. **Primitive data types:**  These are the most basic data types available within the java language. Primitive types are not contain complex behaviors and store actual values.

**Java has Eight Primitive data types:**

1. **byte:** Used to store very small whole numbers within the range (e.g. byte b=100;).

* Size: 8-bit
* Minimum Value: -128
* Maximum Value: 127

1. **short:** Used to store small whole numbers within the range (e.g. short temp=25;)

* Size: 16-bit
* Minimum Value: -32,768–
* Maximum Value: 32,767

1. **int:** Used to store whole numbers within the range (e.g. int age =21;)

* size: 32-bit
* Minimum Value: -2,147,483,648
* Maximum Value: 2,147,483,647

1. **long:** Used for verylarge integer values and append ‘L’ to the value

(e.g. long timestamp = 1234432654L;).

* Size: 64-bit
* Minimum Value: -9,223,372,036,854,775,808
* Maximum Value: 9,223,372,036,854,775,807

1. **float:** Used to store decimal values. Append ‘f’ to the value (e.g. float rate=3.14f;).

* Size: 32-bit
* Minimum Value: 1.4E-45
* Maximum Value: 3.4028235E+38

1. **double:** Used to store decimal values with high precision (e.g. double marks=8.10;)

* Size: 64-bit
* Minimum Value: 4.9E-324
* Maximum Value: 1.7976931348623157E+308

1. **boolean:**  It represents a logical value and stores true or false.

(e.g. boolean isAdmin= true; or boolean isValid = false;)

* Size: 1-bit
* Values: true or false

1. **char:**  It represents a single character (e.g. char initial = ‘A’; or char symbol =’@’;)

* Size: 16-bit
* Values: Unicode characters (letters, digits, symbols, etc)

1. **Non-Primitive data types:** These data types do not directly store values but instead store references to objects.

**Java has Four non-primitive data types:**

1. **Classes:** A blueprint for creating objects. Defines attributes and behaviors.
2. **Array:** Collections of elements of the same data type.
3. **String:** Sequence of characters, treated as objects in java.
4. **Interfaces:** Blueprints of a class, defining methods that implementing classes must provide.

* **What is a Nibble?**

A nibble is four bits or half of a byte. Nibbles are also useful in various encoding schemes, memory addressing and simplifying the representation of binary-coded data.

* **Visual Representation of a Nibble:**

One nibble is equivalent to 4 bits, i.e. (4 bits = Nibble), therefore nibble refers to four consecutive binary digits.

* As we all know that in a byte there are in total 8 bit, therefore (1 Byte = 8 Bits)
* Half of a byte, means ½ of a byte
* ½ Byte => 8/2 Bits
* ½ Byte => 4 bits
* ½ Byte => 1 Nibble (since we discovered above that 4 Bits = 1 Nibble)
* **Steps to work with Nibbles:**

1. **Converting Binary to Decimal**

* To convert a binary nibble to its decimal equivalent, the steps are as follows
* Write down the binary number.
* Assign positional values starting from the right (2^0, 2^1, 2^2, 2^3, 2^4, 2^5)
* Multiply each bit by its positional value.
* Sum the results.

1. **Converting Decimal to Binary**

* To convert a decimal number (0-15) to its binary equivalent:
* Divide the decimal number by 2.
* Write down the remainder.
* Repeat the process with the quotient until the quotient is 0 (zero).
* The binary number is the remainders read from bottom to top.