Data Types

What is a Datatype?

A datatype tells you what kind of data a variable can hold. It helps the computer understand what to expect and how to handle the data.

1. byte – (Small integer)

Stores integers between -128 to 127 (8-bit).

Example: Useful for memory-constrained situations like storing image pixel values.

```
byte smallNumber = 100;
```

2. short – (16-bit integer)

Stores integers between -32,768 to 32,767.

Example: Helpful when storing large sets of numbers with slightly higher precision than bytes.

```
short distance = 30000;
```

3. int – (Standard integer)

Stores integers up to 2^31-1. It's the most commonly used integer type.

Example: Storing a user's age or bank balance.

```
int age = 25;
```

4. long – (Large integer)

Stores large integers up to 2^63-1.

Example: Tracking high-precision values, like a unique customer ID.

```
long population = 7800000000L;
```

5. float – (Decimal with less precision)

A 32-bit floating point number.

Example: Storing approximate values like temperature or distance.

float temperature = 36.6f;

6. double – (High-precision decimal)

A 64-bit floating point number for more precision.

Example: Useful for scientific calculations.

double pi = 3.141592653589793;

7. char – (Single character)

Stores a single 16-bit character.

Example: Representing a grade or single-letter input.

char grade = 'A';

8. boolean – Logical value (`true/false)

Holds one of two values: true or false.

Example: Representing switches or flags.

boolean isLoggedIn = true;

9. String – (Sequence of characters)

A sequence of text or characters, immutable in Java.

Example: Storing names, messages, or documents.

String welcome message = "Hello, World!";

10. Array – (Collection of values)

A container that holds a fixed number of values of the same type.

Example: Storing multiple grades or names in a list.

```
int[] scores = {85, 90, 78};
```

11. Class/Object –(Custom data types)

Encapsulates data and operations (methods) into a blueprint.

Example: A class can represent complex entities like a `Student` with attributes and behavior.

```
class Student {
    String name;
    int age;

    Student(String name, int age) {
        this.name = name;
        this.age = age;
    }
}
```

Why are Datatypes Important?

- Memory Management: Helps the computer allocate the right amount of memory.
- Operations: Determines what operations can be performed (like adding numbers or combining strings).
- Error Prevention: Helps catch errors when you try to use the wrong type of data.

Example

```
int age = 25;              // Integer datatype
float height = 5.9f;       // Floating-point datatype
```

```
char grade = 'A';          // Character datatype
boolean student = true;  // Boolean datatype
String name = "Alice";     // String datatype
```