Data Types

Java is a statically typed language. It categorizes data types into two main groups.

- **Primitive Types**: Basic data types that store raw values directly in memory.
  - **`byte`**: 8-bit signed integer.
  - **`short`**: 16-bit signed integer.
  - **`int`**: 32-bit signed integer.
  - **`long`**: 64-bit signed integer (suffixed with `L`).
  - **`float`**: 32-bit floating-point number (suffixed with `f`).
  - **`double`**: 64-bit floating-point number.
  - **`char`**: 16-bit Unicode character.
  - **`boolean`**: Stores **`true`** or **`false`**.
- **Reference Types**: Objects that point to a memory location in the heap.
  - Includes arrays, classes, interfaces, and enums.
  - Default value for any reference variable is `null`.

Variables

Variables are named memory locations used to store data during program execution.

Declaration: Allocates memory based on the specified data type.

- **Initialization**: Assigns an initial value to the declared variable.
- **Scope**: Dictates where the variable can be accessed (Local, Instance, or Class level).

```
int userAge = 25; // Type, identifier, and initialization
```

Constants

Constants are variables whose values cannot change once assigned

- Declared using the `final` keyword.
- Usually named in all uppercase with underscores.
- Must be initialized immediately or within a constructor

java

```
final double MAX_SPEED = 120.5;
```

Operators and Expressions

Operators

Operators are special symbols used to perform operations on variables and values. [1, 2, 3, 4]

- **Arithmetic**: `+`, `-`, `*`, `/`, `%` (modulus).
- **Unary**: `++` (increment), `--` (decrement), `!` (logical NOT).
- **Assignment**: `=`, `+=`, `-=`, `*=`, `/=`.
- **Relational**: `==`, `!=`, `>`, `<`, `>=`, `<=`.
- **Logical**: `&&` (AND), `||` (OR), `!` (NOT).
- **Bitwise**: `&` (AND), `|` (OR), `^` (XOR), `~` (compliment), `<<` (left shift), `>>` (right shift).
- **Ternary**: Shorthand for if-else `condition ? value_if_true : value_if_false`. [1, 2, 3, 4, 5]

Expressions

An expression is a combination of variables, operators, and method calls that evaluates to a single value. [1, 2, 3]

java
```
int result = (basePrice * quantity) +
taxAmount; // Valid expression
```

Control Flow Statements

Decision Making

Directs program execution along specific paths based on conditions. [1, 2]

- `if` / `else if` / `else`: Executes code blocks based on boolean conditions.
- `switch`: Matches a variable against multiple constant values (supports `byte`, `short`, `char`, `int`, `String`, and enums).

java
```
switch (dayNumber) {
    case 1 -> System.out.println("Monday");
    case 2 -> System.out.println("Tuesday");
    default -> System.out.println("Other day");
}
```
Use code with caution.

Looping Statements

Repeats a block of code multiple times.

- `for`: Used when the exact number of iterations is known.
- `enhanced for` (for-each): Iterates through arrays or collections sequentially.
- `while`: Repeats as long as a condition remains true.
- `do-while`: Executes the code block once before checking the condition.

Branching Statements

Alters the standard execution flow inside loops or switches.

- `break`: Terminates the innermost loop or switch statement immediately.
- `continue`: Skips the current iteration and goes to the next loop pass.
- `return`: Exits from the current method, optionally returning a value.

Methods and Arrays

Methods

Methods are blocks of code designed to perform specific, reusable tasks.

- **Signature**: Combines the method name and its parameter list.
- **Return Type**: The data type of the value returned by the method (`void` if nothing is returned).
- **Parameters**: Inputs accepted by the method during execution.
- **Overloading**: Multiple methods sharing the same name but having different parameter lists.

```
public int calculateSum(int firstNumber, int
secondNumber) {
    return firstNumber + secondNumber;
}
```

Arrays

Arrays are fixed-size, ordered collections holding data items of the exact same type.

- **Indexing**: Elements are accessed via zero-based indexing (`0` to `length - 1`).
- **Creation**: Allocated on the heap using the `new` keyword.
- **Multidimensional**: Arrays of arrays used to store matrix-like data.

```
int[] lotteryNumbers = new int[5]; // Allocates
space for 5 integers
int[][] matrixGrid = {{1, 2},{3, 4}}; //2Darray
```

Classes, Objects, and Constructors

Classes and Objects

Java is built entirely around the Object-Oriented Programming (OOP) paradigm.

- **Class**: A blueprint or template defining the properties (fields) and behaviors (methods) of an entity.
- **Object**: A concrete instance of a class allocated in memory.

Constructors

Special methods called automatically during object creation to initialize instance fields.

- They share the exact name of the class.
- They have no return type (not even `void`).
- **Default Constructor**: Automatically provided by the compiler if no constructor is written.
- **Parameterized Constructor**: Written by the programmer to initialize fields with custom values.

```
public class Car {
    String model;

    // Parameterized Constructor
    public Car(String modelName) {
        this.model = modelName;
    }
}
```