Java Basics

1. Java Syntax

- **Syntax** refers to the rules that define how to write valid code in Java. It includes the structure of statements, the use of punctuation, and the proper use of keywords.
- **Case Sensitivity**: Java is case-sensitive, which means that `Variable` and `variable` are different identifiers.

Class Definition: Every Java application begins with a class definition.
java

```
public class Main {
    public static void main(String[] args) {
        // Code goes here
    }
}
```

-
- **Statements**: Each statement ends with a semicolon (`;`).

2. Variables

- **Variables** store data values. Each variable has a data type that determines the type of data it can hold.

Declaration and Initialization:
java

```
int number = 5;       // Integer
double pi = 3.14;     // Floating-point number
char letter = 'A';    // Character
String message = "Hello, World!"; // String
boolean isJavaFun = true; // Boolean
```

-
- **Types of Variables**:
    - **Local Variables**: Declared within a method.
    - **Instance Variables**: Declared within a class but outside any method.

- **Static Variables**: Declared with the `static` keyword, shared among all instances of a class.

3. Loops

- Loops allow repetitive execution of a block of code.
- **Types of Loops**:

`for` Loop: Used when the number of iterations is known.
java

```
for (int i = 0; i < 5; i++) {
    System.out.println(i);
}
```

`while` Loop: Used when the number of iterations is not known.
java

```
int i = 0;
while (i < 5) {
    System.out.println(i);
    i++;
}
```

`do-while` Loop: Similar to the `while` loop, but the code block is executed at least once.
java

```
int i = 0;
do {
    System.out.println(i);
    i++;
} while (i < 5);
```

4. Methods

- Methods are blocks of code that perform a specific task and can be reused.

Declaration:
java

```
public static void sayHello() {
    System.out.println("Hello!");
}
```

-

Calling a Method:
java

```
sayHello(); // Calls the sayHello method
```

-

Parameters: Methods can accept parameters to perform tasks based on input.
java

```
public static void greet(String name) {
    System.out.println("Hello, " + name);
}

greet("Alice");
```

-

Return Types: Methods can return values.
java

```
public static int add(int a, int b) {
    return a + b;
}

int sum = add(5, 3);
```

-

5. Control Structures

- Control structures manage the flow of the program.

`if-else` Statement: Used for conditional execution.
java

```java
int number = 10;
if (number > 0) {
    System.out.println("Positive");
} else {
    System.out.println("Non-positive");
}
```

-

`switch` Statement: Used for selecting one of many code blocks to execute.
java

```java
int day = 2;
switch (day) {
    case 1:
        System.out.println("Monday");
        break;
    case 2:
        System.out.println("Tuesday");
        break;
    // Other cases...
    default:
        System.out.println("Invalid day");
}
```

-

Ternary Operator: A shorthand for `if-else`.
java

```java
int num = 10;
String result = (num > 0) ? "Positive" : "Non-positive";
```

-

Summary

- **Syntax**: The structure and rules for writing Java code.
- **Variables**: Containers for storing data values, defined by types.
- **Loops**: Repeatedly execute a block of code (`for`, `while`, `do-while`).
- **Methods**: Encapsulate code that performs specific tasks, can take parameters, and may return values.
- **Control Structures**: Direct the flow of the program using `if-else`, `switch`, and the ternary operator.