# Class Lecture #3

## Use of Comments in java programs

In Java, comments are lines of text that are not executed as part of the program but are included in the code to provide explanations, notes, or annotations. They are used to make the code more readable and understandable for programmers.

**Single Line Comments:**

**// this is a single line java comment**

**Multiline Comments:**

**/* this is a**

**Multiline java**

**Comments */**

**There are three types of comments in Java:**

1. **Single-Line Comments: These comments begin with // and extend to the end of the line. They are often used for short explanations or notes on a specific line.**

**// This is a single-line comment int age = 25;**

**// This comment explains the variable's purpose**

2. **Multi-Line Comments: These comments are enclosed between /* and */. They can span multiple lines and are useful for longer explanations or temporarily "commenting out" blocks of code.**

**/* This is a multi-line comment.**

**It can span multiple lines. */**

**/***

**int x = 10;**

**int y = 20;**

***/**

3. **Documentation Comments (Javadoc Comments): These comments are also enclosed between /* and */, but they are used to generate documentation for the code. They often include information about classes, methods, and their purpose.**

# *** Understanding Java Variables, Data Types and Identifiers:

**Variables:**
In Java, a variable is like a data container, where you can store information. It holds different types of data, like numbers or words, and helps your program remember and use that data as needed. Think of it as a container for storing values that your program can work with.

In Java, every variable must have a data type associated with it. The data type defines the type of value the variable can hold and the operations that can be performed on it. This is a fundamental concept in Java's strong typing system. The data type ensures that the variable is used correctly, prevents type-related errors, and allows the compiler to allocate memory for the variable appropriately

**Data Types**: In Java, variables have data types that specify the type of data they can hold.

int is for integers, double is for floating-point numbers, String is for text, and boolean is for true/false.

**Identifiers**: Identifiers are names given to variables, classes, methods, etc. They must start with a letter, underscore, or dollar sign, and can include letters, digits, underscores, and dollar signs.

They are case-sensitive, so 'age' and 'Age' are different identifiers.

```
public class SimpleProgram {
    public static void main(String[] args) {
        // Declaration and Initialization of Variables
        int age = 25;
 // An integer variable named 'age' with a value of 25
        double height = 5.9;
// A double variable named 'height' with a value of 5.9
        String name = "John";
// A String variable named 'name' with a value "John"
        boolean isStudent = true;  // A boolean variable named 'isStudent' with a value of true
        // Using Variables
```

```
        System.out.println("Hello, my name is " + name + ".");
        System.out.println("I am " + age + " years old and " + height + " feet tall.");
        System.out.println("Am I a student? " + isStudent);
    }
}
```

**In this simple Java program:**

- We declare and initialize variables like **age**, **height**, **name**, and **isStudent** with their respective data types.
- Data types define what kind of values a variable can hold. For example, **int** is for whole numbers, **double** is for decimal numbers, **String** is for text, and **boolean** is for true/false values.
- Identifiers like **age**, **height**, and **name** are used to refer to the variables.
- We use these variables to display information about a person, like their name, age, height, and student status.
- The **System.out.println()** statements are used to print the information to the console.

Understanding variables, data types, and identifiers is fundamental in Java programming. They allow you to store and manipulate different types of information in your programs.

## Java Data Types In details

let's Explore the Java's data types:

Data types in Java classify the type of data a variable can hold. There are two categories of data types in Java:

1. **Primitive Data Types:** These are the most basic data types that Java provides. They represent single values and are built into the language. There are eight primitive data types in Java:
    1. **byte:** 8-bit signed integer. Range: -128 to 127.
    2. **short:** 16-bit signed integer. Range: -32,768 to 32,767.
    3. **int: 32-bit signed integer.** Range: -2^31 to 2^31 - 1. (^ stands for Power**)**
    4. **long**: 64-bit signed integer. Range: -2^63 to 2^63 - 1. (^ stands for Power)
    5. **float**: 32-bit floating-point number.

6. **double:** 64-bit floating-point number.
7. **char:** 16-bit Unicode character.
8. **boolean:** Represents true or false.

**Example:**

**Java code**

**int age = 25;**

**double salary = 55000.50;**

**char grade = 'A';**

**boolean isStudent = true;**

## 2. Reference Data Types:

In Java, reference types are data types that refer to objects stored in memory rather than holding the actual data itself. They point to the memory location where the object is stored. Reference types include classes, interfaces, arrays, and enums. When you create an object using a reference type, you're actually creating a reference to the memory location where the object's data is stored, allowing you to manipulate and interact with the object's properties and methods. This is different from primitive types (like int or char) which directly hold the actual data values. These data types are used to store references to objects. They don't hold the actual data but refer to memory locations where the data resides. Reference data types include:

- **Classes**
- **Interfaces**
- **Arrays**
- **Enums**

**Example:**

**java code**

**String name = "John";**

**// R**eference to a String object **int[] numbers =** {1, 2, 3, 4};

**// Reference to an array of integers**

It's crucial to understand the distinction between these data types, as primitive types are stored in memory directly, while reference types hold a reference to an object's memory location. Also, primitive types are faster in terms of memory access and manipulation compared to reference types.

Keep in mind that Java is a statically-typed language, which means you need to declare the data type of a variable when you declare it. This adds to the code's clarity and helps catch type-related errors during compilation.

## Difference Between Double and Float in Java with Example

In Java, both **double** and **float** are used to store floating-point numbers (numbers with decimal points). The main difference between them is their precision.

1. **float**: It is a 32-bit floating-point type, and it's used for numbers with less precision after the decimal point. For example: java code

float floatValue = 3.14159f; System.out.println(floatValue); // Output: 3.14159

2. **double**: It is a 64-bit floating-point type, and it's used for numbers with higher precision after the decimal point. For example:

double doubleValue = 3.14159; System.out.println(doubleValue); // Output: 3.14159

Since **double** has more bits to store the number, it can represent more precise decimal values than **float**. However, **float** uses less memory. It's essential to choose the appropriate type based on the precision you need for your calculations.

# Explain the rules of naming variables in java

In Java, variables must adhere to certain rules when it comes to naming. Here are the key rules for naming variables:

1. **Start with a Letter or Underscore:** Variable names must begin with a letter (uppercase or lowercase) or an underscore (_). They cannot start with a number.
2. **Followed by Letters, Digits, or Underscores:** After the initial character, variable names can consist of letters, digits, and underscores. They cannot contain spaces or special characters.
3. **Case Sensitivity:** Java is case-sensitive, so myVariable and MyVariable are considered different names.
4. **No Keywords or Reserved Words:** You cannot use Java keywords (e.g., int, if, while) or reserved words (e.g., public, static, class) as variable names.
5. **Meaningful Names:** Use descriptive and meaningful names that indicate the purpose of the variable. For example, age instead of a, totalSales instead of ts.
6. **CamelCase Convention**: It's common to use CamelCase for variable names where the first word is in lowercase and subsequent words start with an uppercase letter (e.g., myVariableName).
7. **Avoid Starting with Underscore:** While starting with an underscore is allowed, it's a convention to use it only for special purposes, as it might be confusing.

8. **No Special Characters:** Variable names cannot include special characters like @, #, $, %, etc.
9. **Length Limit**: Variable names can be of any length, but it's best to keep them reasonably concise for readability.
10. **Cannot Match Class Names:** A variable name cannot have the same name as a class name within the same scope.

**Remember, following these rules will not only make your code readable but also prevent errors due to incorrect variable naming.**