Java Object Oriented Programming

Course : Object Oriented Programming in Java

Lecture 3 : Java Basics

Instructor: Vivek Yadav, IIIT Bangalore

Today’s Agenda

1. **Data Types in Java**
2. Variable in Java
3. Decision Control in Java
4. Loop Control in Java
5. Function (Methods) in Java

Data Type in Java

What are Data Types?

- Data types specify the size and type of values that can be stored in variables.

Java has two categories of data types:

- **Primitive Data Types** (8 types)
- **Reference/Object Data Types**

Primitive Data Types in Java

- **byte**: 1 byte, stores whole numbers from -128 to 127
- **short**: 2 bytes, stores whole numbers from -32,768 to 32,767
- **int**: 4 bytes, stores whole numbers from -2^31 to 2^31-1
- **long**: 8 bytes, stores whole numbers from -2^63 to 2^63-1
- **float**: 4 bytes, stores fractional numbers (single precision)
- **double**: 8 bytes, stores fractional numbers (double precision)
- **char**: 2 bytes, stores a single character/letter/ASCII value
- **boolean**: 1 bit, stores true or false

Literals in Java

Definition: A literal is a fixed value that is directly represented in the source code.

Purpose: Used to assign constant values to variables of different data types.

Key Points to Remember

•**Integer Types:** Default to int. Use L for long (e.g., long num = 123L;).
•**Floating-point Types:** Defaults to double. Use f for float (e.g., float value = 3.14f
•**Escape Sequences in Characters:** Use \n, \t, \', \", etc., for special characters.
•**Null for Objects:** Can be assigned to reference types to represent 'no value'.

Types of Literals in Java

- **Integer Literals:** Used for numbers without a decimal.
    - Examples: int a = 100;, int b = 0x1A; // hexadecimal
    - **Formats:** Decimal (100), Binary (0b1010), Octal (014), Hexadecimal (0x1A)
- **Floating-point Literals:** Used for numbers with decimal points.
    - Examples: float f = 10.5f;, double d = 20.05;
    - **Note:** f or F suffix for float, d or D (optional) for double
- **Character Literals:** Represents a single character.
    - Examples: char ch = 'A';, char ch2 = '\n'; // newline character
    - **Unicode Characters:** Can use Unicode like char ch = '\u0041'; // 'A'
- **String Literals:** Represents sequences of characters.
    - Examples: String name = "Hello";
    - **Immutability:** Once created, string values can’t be changed.
- **Boolean Literals:** Represents true or false values.
    - Examples: boolean flag = true;, boolean check = false;

Primitive Data Types in Java

```
public class PrimitiveDataTypes {
    public static void main(String[] args) {
        byte byteVar = 100;
        short shortVar = 10000;
        int intVar = 100000;
        long longVar = 10000000000L;
        float floatVar = 5.75f;
        double doubleVar = 19.99;
        char charVar = 'A';
        boolean boolVar = true;
```

```
  System.out.println("byte: " + byteVar);
        System.out.println("short: " + shortVar);
        System.out.println("int: " + intVar);
        System.out.println("long: " + longVar);
        System.out.println("float: " + floatVar);
        System.out.println("double: " + doubleVar);
        System.out.println("char: " + charVar);
        System.out.println("boolean: " + boolVar);
    }
}
```

Reference (Object) Data Types in Java

Used to store objects.

Includes classes, arrays, and interfaces.

Example: String, arrays, custom objects.

Reference (Object) Data Types in Java

```
public class ReferenceDataTypes {
    public static void main(String[] args) {
        String str = "Hello, World!";
        int[] arr = {1, 2, 3, 4, 5};

        System.out.println("String: " + str);
        System.out.println("Array element at index 0: "
+ arr[0]);
    }
}
```

Today’s Agenda

1. Data Types in Java
2. **Variable in Java**
3. Decision Control in Java
4. Loop Control in Java
5. Function (Methods) in Java

Variables in Java

Variables act as containers to store data values.

Variable declaration: datatype variableName = value;

Variable types:

- **Local**: Declared inside methods.
- **Instance**: Declared inside a class but outside any method.
- **Static**: Declared using the static keyword.

```
class Student
{  string rollno;
   String name;
   static int count;

}
```

```
student a = new student();
student b =  "       ";
```

```
a          b
└ r        └ r
└ n        └ n
```

```
student.count.
 count
```

Variables in Java

```
public class VariablesDemo {
    // Instance variable
    int instanceVar = 50;

    // Static variable
    static int staticVar = 100;

    public static void main(String[] args) {
        // Local variable
        int localVar = 25;
        System.out.println("Local Variable: " +
localVar);

        VariablesDemo obj = new VariablesDemo();
        System.out.println("Instance Variable: " +
obj.instanceVar);
        System.out.println("Static Variable: " +
staticVar);
    }
}
```

Type Casting in Java

Type Casting: Converting a variable from one data type to another.

- **Widening Casting (automatic)**: Smaller to larger type (e.g., int to long).
- **Narrowing Casting (manual)**: Larger to smaller type (e.g., double to int).

{ variable -

{ variable

— insid

—

}

} —

Type Casting in Java

```
public class TypeCasting {
    public static void main(String[] args) {
        // Widening casting
        int intVal = 9;
        double doubleVal = intVal; // Automatic
conversion
        System.out.println("Widening Casting: " +
doubleVal);
```

```
 // Narrowing casting
        double doubleVar = 9.78;
        int intVar = (int) doubleVar; // Manual
conversion
        System.out.println("Narrowing Casting: " +
intVar);
    }
}
```

Today’s Agenda

1. Data Types in Java
2. Variable in Java
3. **Decision Control in Java**
4. Loop Control in Java
5. Function (Methods) in Java

Decision Control in Java

Helps make choices in code execution based on conditions.

- **if** statement
- **else if** and **else** statements
- **switch** statement

Decision Control in Java

```
public class DecisionControl {
    public static void main(String[] args) {
        int age = 20;

        // if-else if-else
        if (age < 18) {
            System.out.println("You are under 18.");
        } else if (age == 18) {
            System.out.println("You are exactly 18.");
        } else {
            System.out.println("You are over 18.");
        }
 // switch statement
        char grade = 'B';
```

```
        switch (grade) {
            case 'A':
                System.out.println("Excellent!");
                break;
            case 'B':
                System.out.println("Good job!");
                break;
            case 'C':
                System.out.println("Passed");
                break;
            default:
                System.out.println("Invalid grade");
        }
    }
}
```

Today’s Agenda

1. Data Types in Java
2. Variable in Java
3. Decision Control in Java
4. **Loop Control in Java**
5. Function (Methods) in Java

Loop Control in Java

Repeats a block of code multiple times.

- **for loop**: Iterates a fixed number of times.
- **while loop**: Executes as long as a condition is true.
- **do-while loop**: Executes at least once, then checks the condition

Loop Control in Java

```
public class LoopControl {
    public static void main(String[] args) {
        // For loop example
        for (int i = 0; i < 5; i++) {
            System.out.println("Iteration: " + i);
        }

        // While loop example
        int count = 0;
        while (count < 5) {
            System.out.println("Count: " + count);
            count++;
        }
```

```
  // Do-while loop example
        int num = 10;
        do {
            System.out.println("Number is: " + num);
            num--;
        } while (num > 5);
    }
}
```

Today's Agenda

1. Data Types in Java
2. Variable in Java
3. Decision Control in Java
4. Loop Control in Java
5. **Function (Methods) in Java**

Function (Methods) in Java

Reusable blocks of code that perform specific tasks.

- **Definition**: Defined once, can be called multiple times.
- **Return types**: Specify what type of data is returned (if any).
- **Arguments**: Inputs passed to the method.

Functions (Methods) in Java

```
public class FunctionsDemo {
    // Function to add two numbers
    public static int add(int a, int b) {
        return a + b;
    }

    // Function with no return type (void)
    public static void greet(String name) {
        System.out.println("Hello, " + name);
    }
```

```
public static void main(String[] args) {
        // Calling the add function
        int sum = add(5, 10);
        System.out.println("Sum is: " + sum);

        // Calling the greet function
        greet("Alice");
    }
}
```

Thank You