Computer Science by AP Tutorial

Data Types and Operators in Java

Java is a versatile programming language that provides various data types and operators to perform computations and manipulate data. Below is a comprehensive set of notes covering the topic.

---

1. Data Types in Java

Data types specify the type of data that can be stored and manipulated in a program. Java supports two categories of data types:

A. Primitive Data Types

Primitive data types are predefined by Java and serve as the building blocks for data manipulation. There are 8 primitive data types:

1. **byte**
    - Size: 1 byte (8 bits)
    - Range: -128 to 127
    - Example: `byte age = 25;`
2. **short**
    - Size: 2 bytes (16 bits)
    - Range: -32,768 to 32,767
    - Example: `short distance = 15000;`
3. **int**
    - Size: 4 bytes (32 bits)
    - Range: -2^31 to 2^31-1
    - Example: `int salary = 50000;`
4. **long**
    - Size: 8 bytes (64 bits)
    - Range: -2^63 to 2^63-1
    - Example: `long population = 7800000000L;`
5. **float**
    - Size: 4 bytes (32 bits)
    - Precision: Up to 7 decimal digits
    - Example: `float pi = 3.14f;`
6. **double**

- Size: 8 bytes (64 bits)
- Precision: Up to 15 decimal digits
- Example: `double gravity = 9.80665;`

7. **char**
   - Size: 2 bytes (16 bits)
   - Represents a single character using Unicode encoding.
   - Example: `char grade = 'A';`
8. **boolean**
   - Size: ~1 bit (true or false)
   - Represents logical values.
   - Example: `boolean isJavaFun = true;`

B. Non-Primitive Data Types

Non-primitive data types are user-defined and include classes, arrays, interfaces, etc.

1. **String** (a class used for text manipulation)

```
String name = "Java Programming";
```

2. **Array**

```
int[] numbers = {1, 2, 3, 4};
```

3. **Class/Objects**

```
class Student {
    int rollNumber;
    String name;
}
```

---

2. Operators in Java

Operators are special symbols or keywords used to perform operations on variables and values.

A. Types of Operators

1. Arithmetic Operators

Used for mathematical calculations:

- `+` (Addition): `a + b`
- `-` (Subtraction): `a - b`
- `*` (Multiplication): `a * b`
- `/` (Division): `a / b`
- `%` (Modulus): `a % b` (remainder)

2. Relational Operators

Used for comparison between two values:

- `==` (Equal to): `a == b`
- `!=` (Not equal to): `a != b`
- `` (Greater than): `a > b`
- `=` (Greater than or equal to): `a >= b`

3. Logical Operators

Used for logical operations:

- `&&` (Logical AND): `(a > b && c > d)`
- `||` (Logical OR): `(a > b || c > d)`
- `!` (Logical NOT): `!(a > b)`

4. Bitwise Operators

Operate at the bit level:

- `&` (Bitwise AND)
- `|` (Bitwise OR)
- `^` (Bitwise XOR)
- `~` (Bitwise Complement)
- `>` (Right shift)

5. Assignment Operators

Used to assign values:

- `=` (Assign): `x = y`
- Compound assignment operators:

```
x += y; // x = x + y
x -= y; // x = x - y
x *= y; // x = x * y
```

6. Unary Operators

Operate on a single operand:

- Increment (++) and Decrement (--)

```
int x = 5;
x++; // x becomes 6
x--; // x becomes 5
```

- Positive (+) and Negative (-)

```
int y = +10;
int z = -10;
```

7. Ternary Operator

Shorthand for if-else conditions:

```
int result = (condition) ? valueIfTrue : valueIfFalse;
```

8. Instanceof Operator

Checks whether an object is an instance of a specific class or subclass:

```
if(obj instanceof ClassName) {
    // Code block
}
```

B. Precedence and Associativity of Operators

Operator precedence determines the order in which operations are performed in an expression.

1. Highest precedence operators:
    - Parentheses ( )
2. Multiplication, Division, Modulus (*, /, %)
3. Addition, Subtraction (+, -)
4. Relational operators (``, etc.)
5. Logical operators (&&, ||)
6. Assignment operators (=, etc.)

Associativity determines the direction of execution:

- Left-to-right for most operators.
- Right-to-left for assignment operators.

Summary Table

Data Type	Size	Example
byte	1 byte	byte age = 25;
short	2 bytes	short s = 100;
int	4 bytes	int num = 50;
long	8 bytes	long l = 500L;
float	4 bytes	float f = 3.14f;
double	8 bytes	double d = 9.8;
char	2 bytes	char c = 'A';
boolean	~1 bit	boolean flag = true;

Key Points to Remember

1. Primitive data types are predefined by Java and have fixed sizes.
2. Non-primitive data types are more flexible and user-defined.
3. Operators are essential for performing calculations, comparisons, and logical operations.
4. Understanding operator precedence helps avoid bugs in complex expressions.
