Core Java

AJ

Data Types in Java

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

1.Primitive data types: The primitive data types include boolean, char, byte, short, int, long, float and double.

2.Non-primitive data types: The non-primitive data types include Classes, Interfaces, and Arrays

Java Primitive Data Types

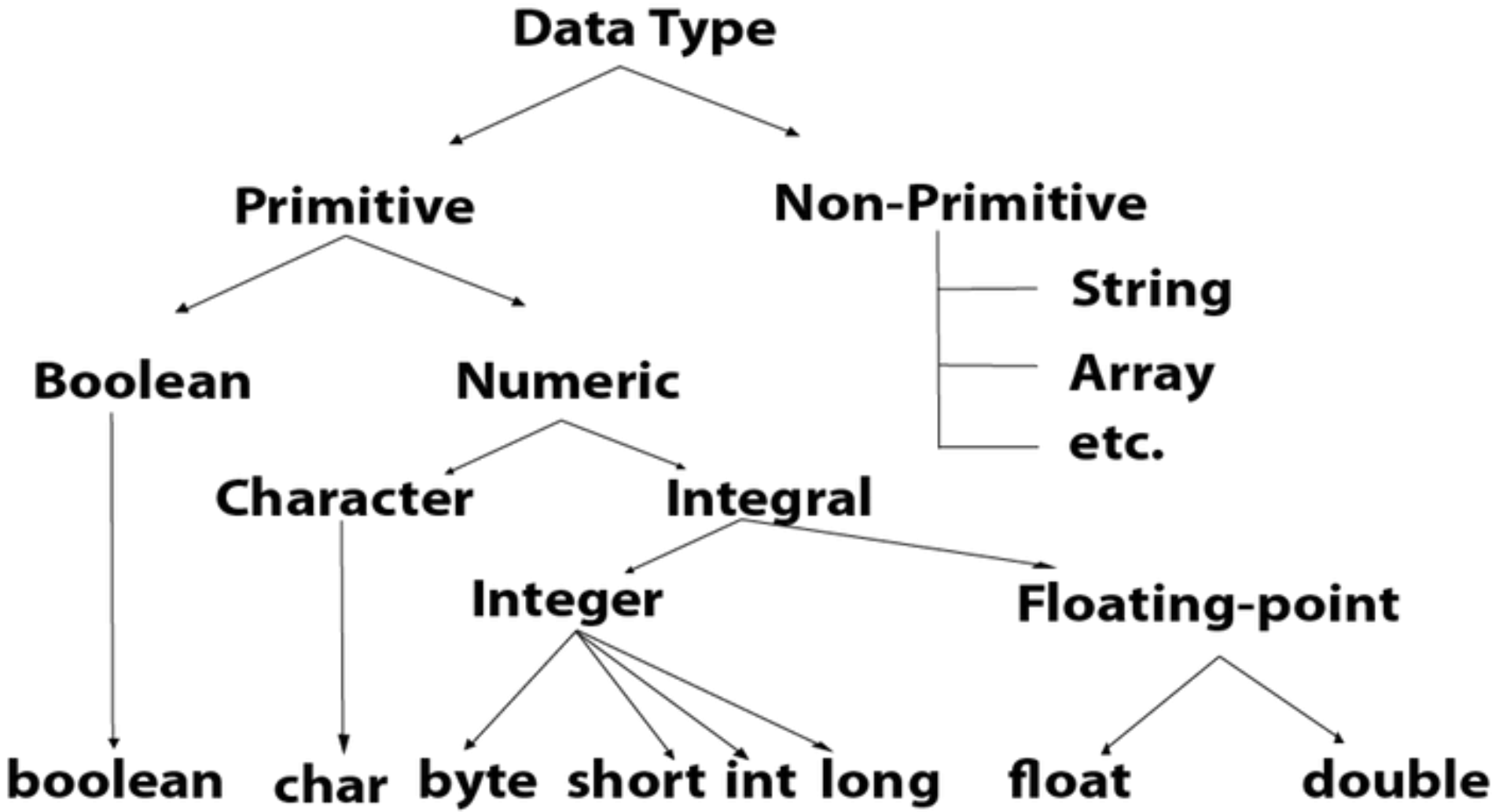
In Java language, primitive data types are the building blocks of data manipulation. These are the most basic data types available in Java language.

Java is a statically-typed programming language. It means, all variables must be declared before its use. That is why we need to declare variable's type and name.

There are 8 types of primitive data types:

- boolean data type
- byte data type
- char data type
- short data type
- int data type
- long data type
- float data type
- double data type

Data Types in Java



Data Types in Java

Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

Operators in java

Operator in java is a symbol that is used to perform operations. For example: +, -, *, / etc.
There are many types of operators in java which are given below:

- Unary Operator,
- Arithmetic Operator,
- Shift Operator,
- Relational Operator,
- Bitwise Operator,
- Logical Operator,
- Ternary Operator and
- Assignment Operator.

Java Unary Operator

The Java unary operators require only one operand. Unary operators are used to perform various operations i.e.:

- incrementing/decrementing a value by one
- negating an expression
- inverting the value of a boolean

Java Unary Operator Example: ++ and --

Java Unary Operator Example: ~ and !

```
1.class OperatorExample{
2.public static void main(String args[]){
3.int a=10;
4.int b=-10;
5.boolean c=true;
6.boolean d=false;
7.System.out.println(~a);//-
11 (minus of total positive value which starts from 0)
8.System.out.println(~b);//9 (positive of total minus, positive st
arts from 0)
9.System.out.println(!c);//false (opposite of boolean value)
10.System.out.println(!d);//true
11.}}
```

Operators in java

Java Arithmetic Operators

Java arithmatic operators are used to perform addition, subtraction, multiplication, and division. They act as basic mathematical operations.

```
1.class OperatorExample{
2.public static void main(String args[]){
3.int a=10;
4.int b=5;
5.System.out.println(a+b);//15
6.System.out.println(a-b);//5
7.System.out.println(a*b);//50
8.System.out.println(a/b);//2
9.System.out.println(a%b);//0
10.}}
```

Operators in java

Java Left Shift Operator

The Java left shift operator << is used to shift all of the bits in a value to the left side of a specified number of times.

Java Left Shift Operator Example

```
1.class OperatorExample{
2.public static void main(String args[]){
3.System.out.println(10<<2);//10*2^2=10*4=40
4.System.out.println(10<<3);//10*2^3=10*8=80
5.System.out.println(20<<2);//20*2^2=20*4=80
6.System.out.println(15<<4);//15*2^4=15*16=240
7.}}
```

Java Right Shift Operator

The Java right shift operator >> is used to move left operands value to right by the number of bits specified by the right operand.

Java Right Shift Operator Example

```
1.class OperatorExample{
2.public static void main(String args[]){
3.System.out.println(10>>2);//10/2^2=10/4=2
4.System.out.println(20>>2);//20/2^2=20/4=5
5.System.out.println(20>>3);//20/2^3=20/8=2
6.}}
```

Operators in java

Java AND Operator Example: Logical && and Bitwise &

The logical && operator doesn't check second condition if first condition is false. It checks second condition only if first one is true.

The bitwise & operator always checks both conditions whether first condition is true or false.

```
1.class OperatorExample{
2.public static void main(String args[]){
3.int a=10;
4.int b=5;
5.int c=20;
6.System.out.println(a<b&&a<c);//false && true = false
7.System.out.println(a<b&a<c);//false & true = false
8.}}
```

Java AND Operator Example: Logical && vs Bitwise &

```
1.class OperatorExample{
2.public static void main(String args[]){
3.int a=10;
4.int b=5;
5.int c=20;
6.System.out.println(a<b&&a++<c);//false && true = false
7.System.out.println(a);//10 because second condition is not ch
ecked
8.System.out.println(a<b&a++<c);//false && true = false
9.System.out.println(a);//11 because second condition is check
ed
10.}}
```

Operators in java

Java OR Operator Example: Logical || and Bitwise |

The logical || operator doesn't check second condition if first condition is true. It checks second condition only if first one is false.

The bitwise | operator always checks both conditions whether first condition is true or false.

```
1.class OperatorExample{
2.public static void main(String args[]){
3.int a=10;
4.int b=5;
5.int c=20;
6.System.out.println(a>b||a<c);//true || true = true
7.System.out.println(a>b|a<c);//true | true = true
8.//|| vs |
9.System.out.println(a>b||a++<c);//true || true = true
10.System.out.println(a);//10 because second condition is not checked
11.System.out.println(a>b|a++<c);//true | true = true
12.System.out.println(a);//11 because second condition is checked
13.}}
```

Operators in java

Java Ternary Operator

Java Ternary operator is used as one liner replacement for if-then-else statement and used a lot in java programming. it is the only conditional operator which takes three operands.

Java Ternary Operator Example

```
1.class OperatorExample{
2.public static void main(String args[]){
3.int a=2;
4.int b=5;
5.int min=(a<b)?a:b;
6.System.out.println(min);
7.}}
```

Java Assignment Operator

Java assignment operator is one of the most common operator. It is used to assign the value on its right to the operand on its left.

```
class OperatorExample{
public static void main(String args[]){
int a=10;
int b=20;
a+=4;//a=a+4 (a=10+4)
b-=4;//b=b-4 (b=20-4)
System.out.println(a);
System.out.println(b);
}}
```

Operators in java

Java Assignment Operator Example: Adding short

```
class OperatorExample{
public static void main(String args[]){
short a=10;
short b=10;
//a+=b;//a=a+b internally so fine
a=a+b;//Compile time error because 10+10=20 now int
System.out.println(a);
}}
```

Compile time error

After type cast:

```
1.class OperatorExample{
2.public static void main(String args[]){
3.short a=10;
4.short b=10;
5.a=(short)(a+b);//20 which is int now converted to short
6.System.out.println(a);
7.}}
```

Output:

20