CORE JAVA

Ideal Computer Center

Data Types in Java

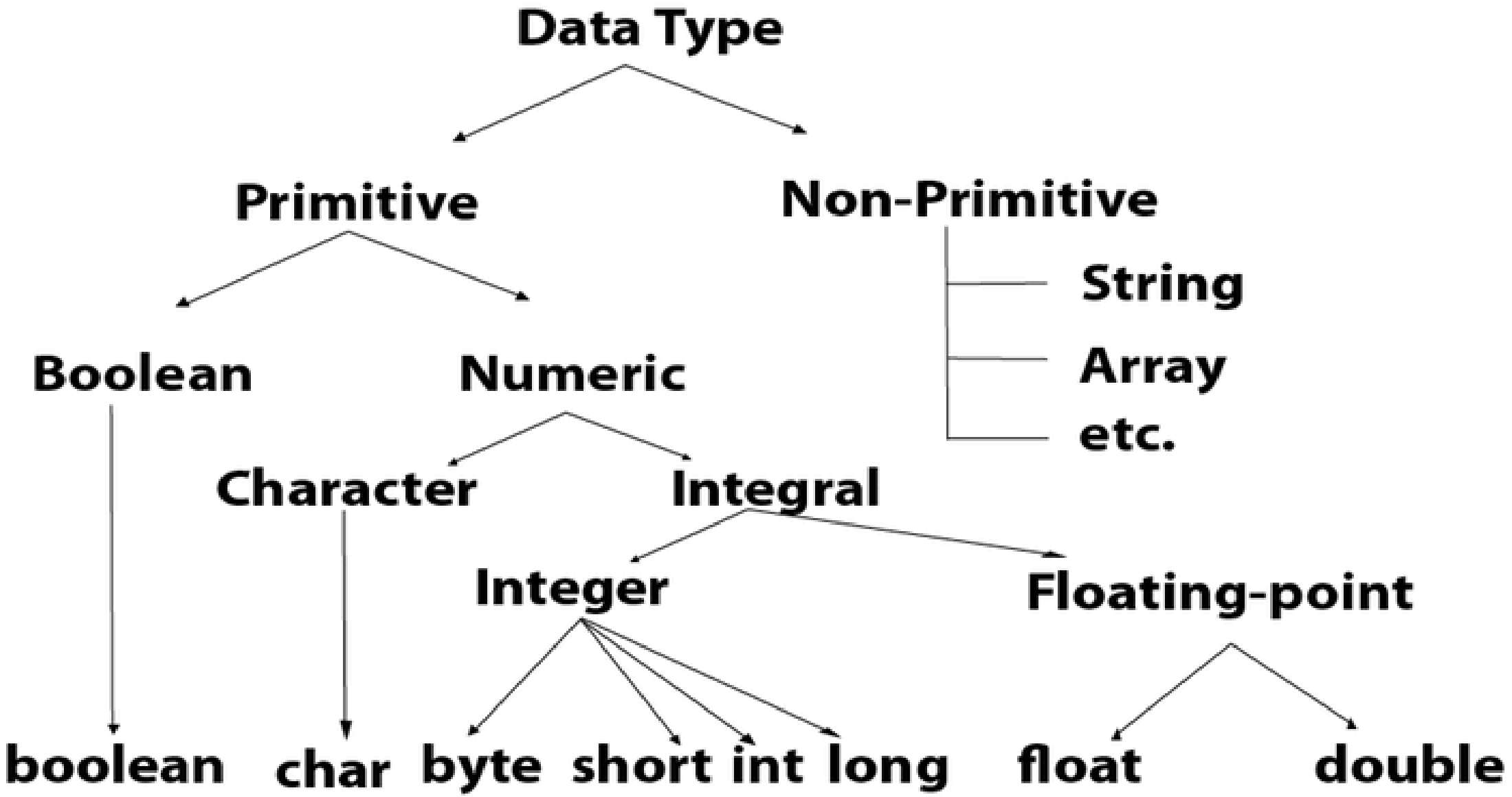
- Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

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

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

Java Primitive Data Types

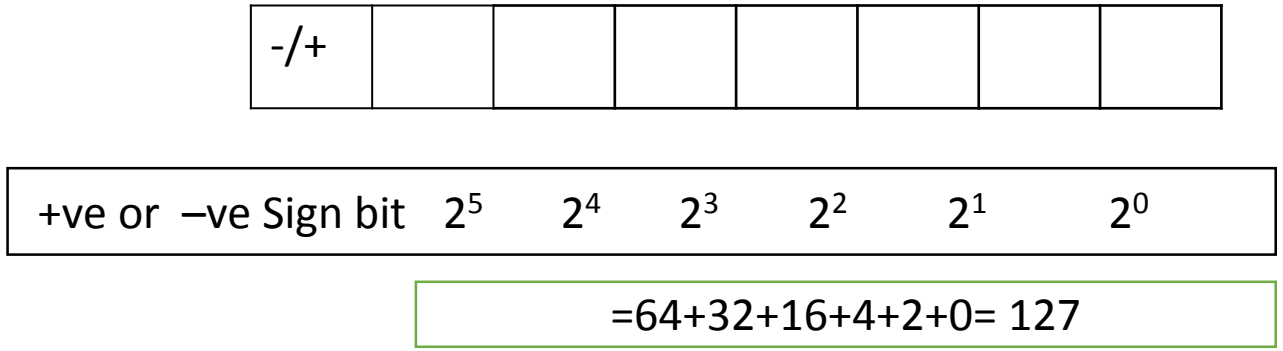
1. boolean data type
2. byte data type
3. char data type
4. short data type
5. int data type
6. long data type
7. float data type
8. double data type


Data Type
Primitive
Non-Primitive
String
Array
etc.
Boolean
Numeric
Character
Integral
Integer
Floating-point
boolean
char
byte
short
int
long
float
double

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

1 Byte Data Type

- The byte data type is an example of primitive data type.
- **Size-**: 1 byte
- **Max Value-**: +127
- **Min Value-**: -128
- **Range** -: -128 to +127
- **default value** -: 0
- **Example:**
- **byte** a = 10, **byte** b = -20
- The byte data type is used to save memory in large arrays where the memory savings is most required. It saves space because a byte is 4 times smaller than an integer. It can also be used in place of "int" data type.



Find output-

```
public class Main {
  public static void main(String[] args) {
    byte b=128;
    System.out.println(b);
  }
}
```

- Output-: CE -: incompatible types: possible lossy conversion from int to byte byte b=128;

Which of following byte declaration are correct

- byte b=10;
- byte b=127;
- byte b=128;
- byte b=true;
- byte b=27;
- byte b=-35;
- Byte b= “yash”;

2. Short Data Type

- The short most rarely used data type in java
- **Size** - 16-bit
- **Max value** is 32,767
- **Min value** is (-32,768)
- **Range** -: -32,768 to 32,767 (inclusive).
- **default value** -: 0.
- The short data type can also be used to save memory just like byte data type. A short data type is 2 times smaller than an integer.

Find output-

```
public class Main {
  public static void main(String[] args) {
    short s=128;
    System.out.println(s);
  }
}
```

- Output-:

3. Int Data Type

- The int data type is a 32-bit signed two's complement integer. Its value-range lies between
- **Size-:**4 byte
- **Max** -: 2,147,483,647.
- **Min** -: 2,147,483,648
- **Range-**: - 2,147,483,648 (-2^31) to 2,147,483,647 (2^31 -1) (inclusive).
- **default value** -: 0.

- E.g int a=20;

Find output-

```
public class Main {
  public static void main(String[] args) {
    int i=18;
    System.out.println(i);
  }
}
```

- Output-:

4. Long Data Type

- The long data type is used when you need a range of values more than those provided by int
- **Size-**: 8 byte
- **Range** -: -9,223,372,036,854,775,808(-2^63) to 9,223,372,036,854,775,807(2^63 -1).
- **Min** - : -9,223,372,036,854,775,808
- **Max** -: 9,223,372,036,854,775,807.
- **default value** -: 0.
- System.out.println(10.0/3);
- Eg long l=26000*40*60*60*24*20

```
public class Main {
  public static void main(String[] args) {
    long l = 19658765765;
    System.out.println(l);
  }
}
```

5.Float Data Type

- The float data type is a single-precision point.
- **Size-4** byte
- **Range** -: $-3.4e^{38}$ to $3.4e^{38}$
- Its value range is unlimited.
- Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
- The float data type should never be used for precise values, such as currency.
- Its default value is 0.0F.
- **E.g float** f = 12.3;

```
public class Main {
  public static void main(String[] args) {
    float myNum = 5.75f;
    System.out.println(myNum);
  }
}
```

6.Double Data Type

- The double data type is a double-precision
- Its value range is unlimited.
- The double data type is generally used for decimal values just like float.
- Stores fractional numbers. Sufficient for storing 15 decimal digits
- The double data type also should never be used for precise values, such as currency.
- Its default value is 0.0d.
- **E.g double** d1 = 12.3

```
public class Main {
  public static void main(String[] args) {
    double myNum = 19.99d;
    System.out.println(myNum);
  }
}
```

Output-:19.99

Which of following double declaration are correct?

- double d=456.423;
- double d=00$4654;
- double d=“a”;
-

7 Char Data Type

- The char data type is a single 16-bit Unicode character with in single quotes(‘ ’)
- Size- 2byte
- Range –: 0 to 65535;
- Range -: '\u0000' (or 0) to '\uffff‘
- Stores a single character/letter or ASCII values
- The char data type is used to store characters.

```
public class Main {
  public static void main(String[] args) {
    char myGrade = 'B';
    System.out.println(myGrade);
  }
}
```

Which of following boolean declaration are correct?

- char a=‘x’;
- char a=‘ab’;
- char ch=“ab”;
- char c=‘A’
- char c=‘a’

8 Boolean Data Type

- The Boolean data type is used to store only two possible values: true and false. This data type is used for simple flags that track true/false conditions.
- The Boolean data type specifies one bit of information
- Default value -false

```
public class Main {
  public static void main(String[] args) {
    boolean isJavaFun = true;
    boolean isMangoTasty = false;
    System.out.println(isJavaFun);
    System.out.println(isMangoTasty);
  }
}
```

Which of following boolean declaration are correct?

- boolean b=true;
- boolean b="True";
- boolean b="true";
- boolean b=0;
- boolean b=false;

```
public class Main {
  public static void main(String[] args) {
    String greeting = "Hello World";
    System.out.println(greeting);
  }
}
```

Add the correct data type for the following variables:

-  `myNum = 9;`
-  `myFloatNum = 8.99f;`
-  `myLetter = 'A';`
-  `myBool = false;`
-  `myText = "Hello World";`

Escape Character

A character preceded by a backslash (\) is an escape sequence and has a special meaning to the compiler.

	Description
\t	Inserts a tab in the text at this point.
\b	Inserts a backspace in the text at this point.
\n	Inserts a newline in the text at this point.
\r	Inserts a carriage return in the text at this point.
\f	Inserts a form feed in the text at this point.
\'	Inserts a single quote character in the text at this point.
\"	Inserts a double quote character in the text at this point.
\\	Inserts a backslash character in the text at this point.

- E.g
- System.out.println(“**Write Once, Run Anywhere. ” + ” \l” + ”Java”);**
- System.out.println(“**Write Once, Run Anywhere. ” + ” \n” + ”Java”);**
- System.out.println(“**Write Once, Run Anywhere. ” + ” \b” + ”Java”);**
- System.out.println(“**Write Once, Run Anywhere. ” + ” \r” + ”Java”);**
- System.out.println(“**Write Once, Run Anywhere. ” + ” \f” + ”Java”);**
- System.out.println(“**Write Once, Run Anywhere. ” + ” \’” + ”Java”);**
- System.out.println(“**Write Once, Run Anywhere.” + ” \””** + ”Java”);
- System.out.println(“**Write Once, Run Anywhere.” + ” \\” + ”Java”);**

Strings Literal

Any Sequence of Character with in double quote(“”)can be treated as String
e.g
String s=“Ideal Computer Center”

```
public class Main {
  public static void main(String[] args) {
    String greeting = "Hello World";
    System.out.println(greeting);
  }
}
```

Java Comments

1. Single-line Comments

Single-line comments start with two forward slashes (//).

Any text between // and the end of the line is ignored by Java (will not be executed).
This example uses a single-line comment before a line of code:

```
// This is a comment
 System.out.println(“Happy New Year");
```

2. Java Multi-line Comments

Multi-line comments start with /* and ends with */.

Any text between /* and */ will be ignored by Java.

This example uses a multi-line comment (a comment block) to explain the code

```
/* New year planning
Exam Plan */


System.out.println("Hello World");
```