Java Identifier:

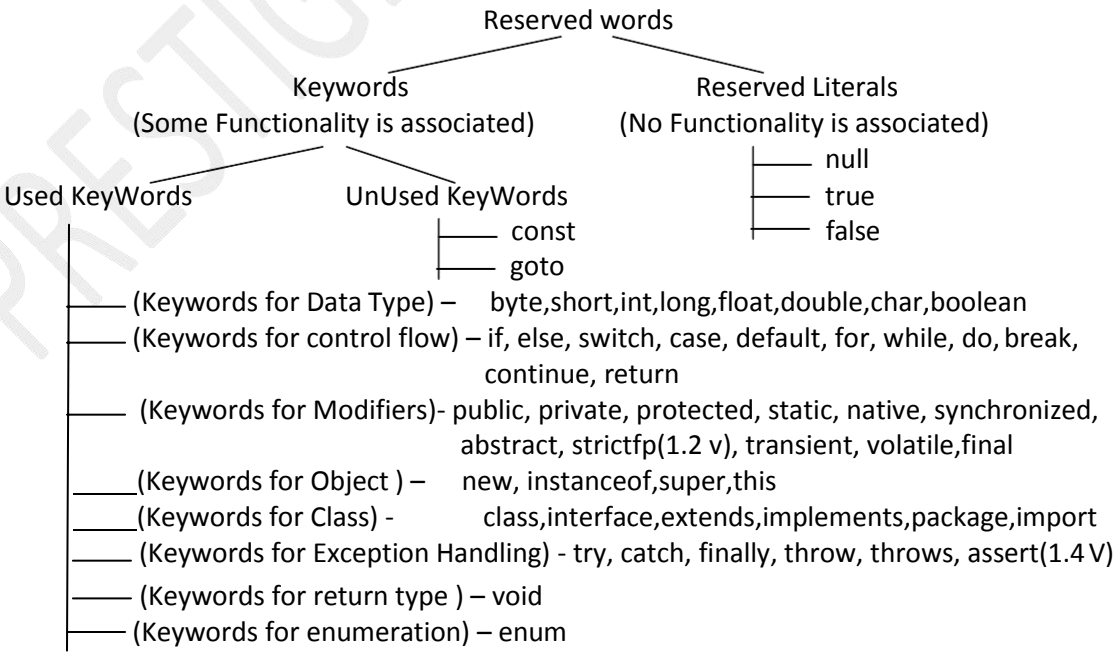
Identifiers are the names of variables, methods, classes, packages and interfaces. Unlike literals they are not the things themselves, just ways of referring to them.

Rules :

1. Allowed characters are :
   - a to z
   - A to Z
   - 0 to 9
   - _
   - $
2. First character of identifier should be a non-digit character. Other than first character of identifier, can be digits.
3. Java identifiers are case sensitive, of course java language itself treated as case sensitive language.
4. There is no length limit for java identifiers .
5. Reserved words are not available for identifiers. i.e. we can use reserved words as identitfiers.
6. Pre-defined class names and interfaces names are available for identifiers.

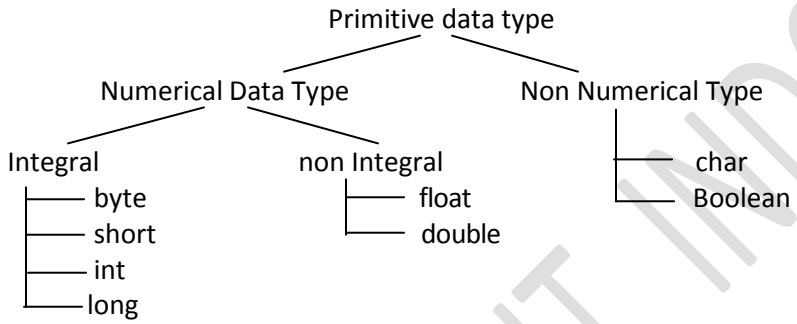
Java Reserved Words:

A reserved word in Java is a special word that gets recognized by the compiler. When the compiler sees a reserved word, it is prompted to do a unique task. Reserved words cannot be used for any other purpose in Java, besides their unique task.



Primitive Data Type:

Primitive types are the most basic data types available within the Java language.
There are : **boolean , byte , char , short , int , long , float and double .**
These types serve as the building blocks of data manipulation in Java. Such types serve only one purpose — containing pure, simple values of a kind.



byte :

- The byte data type is an 8-bit signed two's complement integer.
- It has a minimum value of -128 and a maximum value of 127 (inclusive).
- The byte data type can be useful for saving memory in large arrays, where the memory savings actually matters.

Bit position-

7	6	5	4	3	2	1	0

Bit no 7 is called Most Significant bit and it is used for sign.
If number is positive (e.g. byte b = 20) then this bit contain 0.
If number is negative (e.g. byte b = -20) then this bit contain 1.
Negative numbers are always stored in 2's complement form in the memory .

byte data type is best suitable if we are handling data in terms of streams either from the file the network

short:

- The short data type is a 16-bit signed two's complement integer.
- It has a minimum value of -32,768 and a maximum value of 32,767 (inclusive).
- As with byte, the same guidelines apply: you can use a short to save memory in large arrays, in situations where the memory savings actually matters.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

The most significant bit (bit no 15) is used for sign bit.
If number is positive (e.g. short s = 20) then this bit contain 0.
If number is negative (e.g. short s = -20) then this bit contain 1.
Negative numbers are always stored in 2's complement form in the memory .

short data type is best suitable for 16-bit processors like 8086 but these processors are completely outdated and hence the corresponding short data data type is also rarely used.

int :

- By default, the int data type is a 32-bit signed two's complement integer, which has a minimum value of -2^{31} or -2147483648 and a maximum value of $2^{31}-1$ or 2147483647.

- In Java SE 8 and later, you can use the int data type to represent an unsigned 32-bit integer, which has a minimum value of 0 and a maximum value of $2^{32}-1$.

The most significant bit (bit no 31) is used for sign bit.
If number is positive (e.g. int i = 20) then this bit contain 0.
If number is negative (e.g. int i = -20) then this bit contain 1.

Negative numbers are always stored in 2's complement form in the memory .

This is the most commonly used data type in java.

long :

- The long data type is a 64-bit two's complement integer.
- The signed long has a minimum value of -2^{63} and a maximum value of $2^{63}-1$.
- In Java SE 8 and later, you can use the long data type to represent an unsigned 64-bit long, which has a minimum value of 0 and a maximum value of $2^{64}-1$.

The most significant bit (bit no 63) is used for sign bit.
If number is positive (e.g. long l = 20) then this bit contain 0.
If number is negative (e.g. long l = -20) then this bit contain 1.

Negative numbers are always stored in 2's complement form in the memory .

Whenever int is not enough to hold big values then we should use long data type.

float:

- If we want to represent real numbers, then we can use float or double data type. The float data type is a single-precision 32-bit IEEE 754 floating point.

- A single precision floating point representation uses a word of 32 bit.
  - 1 bit for the sign, S
  - 8 bits for the exponent, 'E'
  - 24 bits for the fraction, also called mantissa, or coefficient (even though just 23 are represented).

Let's call it 'M' (for mantissa, I prefer this name as "fraction" can be misunderstood).
Representation:

```
          S  EEEEEEEE   MMMMMMMMMMMMMMMMMMMMMMM
bits:     31 30         23 22                 0
```

Range of floating point number is -3.4E38 to 3.4E38.

After performing arithmetic operations using float data type, we can get accuracy upto 5 to 6 digits after decimal point.

double:

- Range of double data type is -1.7E308 to 1.7E308.

- The precision indicates the number of decimal digits that are correct, i.e. without any kind of representation error or approximation. In other words, it indicates how many decimal digits one can safely use.

- A double precision floating point representation uses a word of 64 bit.
  - 1 bit for the sign, S
  - 11 bits for the exponent, 'E'
- 53 bits for the fraction / mantissa / coefficient (even though only 52 are represented), 'M'

Representation:

```
      S  EEEEEEEEEEE   MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
bits: 63 62           52 51                                                0
```

In fact, the mantissa is a number represented without all its non-significative 0. For example,
0.000124 becomes 0.124 × 10−3
237.141 becomes 0.237141 × 103

After performing arithmetic operations using double data type, we can get accuracy upto 15 to 16 digits after decimal point.

boolean:

- The boolean data type has only two possible values: true and false.
- Use this data type for simple flags that track true/false conditions. This data type represents one bit of information, but its "size" isn't something that's precisely defined.
- Size : not applicable (Java Virtual dependent)

char:

- The char data type is a single 16-bit Unicode character.
- It has a minimum value of '\u0000' (or 0) and a maximum value of '\uffff' (or 65535 inclusive).
- All characters are internally stored in form of integer values.

In Java we are allowed to use any worldwide alphabets character.

Primitive Data Type default values:

Data Type	Default Value
byte	0
short	0
int	0
long	0
float	0.0f
double	0.0
char	0 or space
boolean	false

Literals:

Any constant value which can be assigned to the variable is called literal.

Integral Literals:
For the integral data types (byte, short, int, long) we can specify literal values in the following ways:
1. Decimal Literals : Allowed digits are 0 to 9. For example
   int i = 20;
2. Octal Literals: Allowed digits are 0 to 7. Literal value should be prefixed with zero.
   int o = 010;
3. Hexa Decimal Literal : Allowed digits are 0 to 9 and alphabets from A to F (or a to f)
   int h = 0x10A ;
   int f = 0X20;

There are two enhancements with respect to integral literal from java 1.7 version onwards
1. Binary Literal – all integral can be represented in binary literal
Int x = 0b1111; or int x = 0B1111;
2. usage of underscore in numeric literals
e.g.
double d = 1234567.789; can be written as double d = 1_23_45_6_7.7_89;
int I = 4_45_5_67_9;
At the time of compilation underscore(_) will be removed automatically. We can use _ between digits only.

By default every integral is int type but we can specify explicitly as long type by suffixing with small 'l' or capital "L".

There is no direct way to specify byte and short literals explicitly. But whenever we are assigning integral literal to the byte variables and its value within the range of byte compiler automatically treats as byte literal. Similarly short literal also.

Floating Point Literals:

Floating point literals are by default double type but we can specify explicitly as float type by suffixing "f" or "F"
We can specify explicitly floating point literal as double type by suffixing with "d" or "D".

We can assign integral literal directly to the floating point data types and that integral literal can be specified in octal and hexa decimal form also.
double d = 0xEEF;

But we can't assign floating point literal directly to the integral type.

We can specify floating point literal even in exponential form also.
double d = 23e10;
float f = 3e10f;

Boolean Literals :

The only allowed value for the boolean data type are 'true' and 'false' where case is important.

Char Literals:

A char literal can be represented as single character within single quotes.
char ch = 'z';

We can specify a char value as integral literal which represents Unicode of that character.

char ch = 97;

We can represent a char literal by Unicode representation which is nothing but '\Uxxxx' where xxxx represents hexadecimal numbers.

char ch = '\u0097';

Java Naming Convention:

Java is a case sensitive language so the way of writing code is important.

1. All Java classes, Abstract classes and Interface names should start with uppercase letter ,if any class contain more than one word every inner word also start with capital letters.
   Ex: String , StringBuffer , FileInputStream
2. All java methods should start with lower case letters and if the method contains more than one word every innerword should start with capital letters. Ex :- post() toString() toUpperCase()
3. All java variables should start with lowercase letter and inner words start with uppercase letter.
   Ex:- pageContent bodyContent
4. All java constant variables should be in uppercase letter.
5. All java packages should start with lower case letters only.
   Ex: java.awt Java.io

NOTE:-

The coding standards are applicable for predefined library not for user defined library. But it is recommended to fallow the coding standards for user defined library also.