Naming convention OR Coding Standard in Java :

Naming convention provides two important characteristics:

a) Standard Code (Industry accepted code)
b) Readability of the code will enhance.

1) How to write a class in java:

While writing a class in java, we should follow pascal naming convention, According to this each word, first character must be capital and it should not contain any space. In java a class represents noun.

Example :
ThisIsExampleOfClass

System
String
Integer
CommandAddition
ArrayIndexOutOfBoundsException
DataInputStream.

2) How to write a method in java :

While writing a method in java we should follow camel case naming convention, According to this naming convention first word will be in small and 2nd word onwards, each word first character must be capital. In java a method represents verb.

Example :
thisIsExampleOfMethod()
read()
readLine()
parseInt()
charAt()
toUpperCase()

3) How to write a field/variable in java :

While writing a variable/field we should follow camel case naming convention but unlike method variable does not have () symbol.

Example :

rollNumber
customerName
customerBill
studentName
playerName

4) How to write a final and static field:

While writing the final and static field we should follow snake_case naming convention. Each character must be capital and in between two words we must have _ symbol.

Example :
SIZE = 8 [SIZE is final and static field]
MIN_VALUE = -128 [MIN_VALUE is final and static field]
MAX_VALUE = 127 [MAX_VALUE is final and static field]

5) How to write a package :

A package must be written in lower case only. Generally it is reverse of company name.

com.nit.basic
com.tcs.chating
com.wipro.shopping

Tokens in Java :

Token is the smallest unit of the program which is identified by the compiler.

Without token we can't complete statement or expression in java.

Token is divided into 5 types in java

1) Keyword
2) Identifier
3) Literal
4) Punctuators (Separators)
5) Operator

Keyword

A keyword is a predefined word whose meaning is already defined by the compiler.

In java all the keywords must be in lowercase only.

A keyword we can't use as a name of the variable, name of the class or name of the method.

true, false and null look like keywords but actually they are literals.

As of now, we have 67+ keywords in java.

goto and const are the reserved words.

Identifiers:

A name in java program by default considered as identifiers.

Assigned to variable, method, classes to uniquely identify them.

We can't use keyword as an identifier.

Example

```
class Fan
{
  int coil ;

  void switchOn()
  {
  }
}
```

Here Fan(Name of the class), coil (Name of the field) and switchOn(Name of the Method) are identifiers.

Rules for defining an identifier :

1) Can consist of uppercase(A-Z), lowercase(a-z), digits(0-9), $ sign, and underscore (_)
2) Begins with letter, $, and _
3) It is case sensitive
4) Cannot be a keyword
5) No limitation of length

Literals:

Any constant which we are assigning to field/variable is called Literal.

In java we have 5 types of Literals:

1) Integral Literal
2) Floating Point Literal
3) Boolean Literal
4) Character Literal
5) String Literal

Note : null is also a literal

Integral Literal :

* If a numeric litreal **does not contain decimal OR fraction** then It is called Integral Literal.
  Example :
  12, 90, 45, 119

* In integral literal, we have 4 data types

  a) byte (8 bits)
  b) short (16 bits)
  c) int (32 bits)
  d) long (64 bits)

* In Java, An integral literal we can represent in four different forms

  a) Decimal Literal (Base 10) [It will accept 10 digits 0 to 9]
  b) Octal Literal (Base 8) [It will accept 8 digits 0 to 7]
  c) Hexadecimal Literal (Base 16) [It will accept 16 digits i.e. 0 to 9 and A - F OR a - f]
  d) Binary Litera (Base 2) [It will accept 2 digits 0 and 1] [Available from JDK 1.7V]

Decimal Literal:

By default, our numeric literals are decimal literal. Here base is 10 so, It accepts 10 digits i.e. from 0-9.

Example:
int x = 20;
int y = 123;
int z = 234;

Octal Literal:

If any Integer literal starts with 0 (Zero) then it will become octal literal. Here base is 8 so it will accept 8 digits i.e. 0 to 7.

Example:
int a = 018; //Invalid because it contains digit 8 which is out of range
int b = 0777; //Valid
int c = 0123; //Valid

Hexadecimal Literal:

If any integral literal starts with 0X or 0x (Zero capital X Or 0 small x) then it is hexadecimal literal. Here base is 16 so it will accept 16 digits i.e 0 to 9 and A to F OR [a to f]

Example :

int x = 0X12; //Valid
int y = 0xadd; //Valid
int z = 0Xface; //valid
int a = 0Xage; //Inavlid because character 'g' out of range

Binary Literal :

If an Integral literal starts with 0B (Zero capital B) or 0b (Zero small b) then it will become Binary literal. Binary literal is available from JDK 1.7v.
Here base is 2 so it will accept 2 digits i.e 0 and 1 only.

Example :

int x = 0B101; //valid
int y = 0b111; //Valid
int z = 0B112; //Invalid [2 is out of range]

* As we know by default our literal is **decimal literal so, As a programmer we can represent an integral literal in decimal, octal, hexadecimal and binary but to get the outout JVM will always convert the literals into decimal literal only.**

Octal
Hexa **decimal**
Binary

Example : 523

5 **2** **3** **decimal number**

$5 \times 10^{2} + 2\times 10^{1} + 3 \times 10^{0}$

//program

```
void main()
{
    int x = 015;
    IO.println(x);
}
```

* In 015, 0 describes that It is octal literal

1 0
(1 5)
8

$1 \times 8^{1} + 5 \times 8^{0}$

8 + 5

13