Literals

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

int x = 10;

keyword/ identifier/ literal/

datatype name of variable constant value

**Integral literal**

For integral datatypes (byte, short, int, long) we can specify value in the following ways

1. Decimal literals

Allowed digits are 0 to 9

Ex. int x = 10;

1. Octal literals (base - 8)

Allowed digits are 0 to 7, literal value should be prefixed with zero (0).

Ex. Int x = 010;

1. Hexadecimal literals (base -16)

Allowed digits are 0 to 9, a to f, for extra digit (a to f) we can use lower and upper case characters.

This is one of area, where java is not case sensitive.

Literal value should be prefixed with zero (ox) or (0X)

Ex. Int x = 0x10; int x = 0X10;

These are the only ways to specify literal value for integral datatypes

**FAQ**

Which of the following are valid variable declarations?

1. int x = 10;
2. int x= 0786;
3. int x = 0777;
4. int x = 0xFace;
5. int x = 0XBeef;
6. int x = 0Xbeer;

What is output of below program?

|  |
| --- |
| **public** **class** Intliteral {  **public** **static** **void** main(String args[]) {  **int** i1 = 10;  **int** i2 = 010;  **int** i3 = 0X10;  System.***out***.println(i1 + "..." + i2 + "..." + i3);  }  } |

Note

1. By default, every integral literal is of int type, but we can specify explicitly as long type by suffixed with l or L.

EX.

int i1 = 10;

long l1 = 10L;

long l2= 10l;

int i2 = 10l; //CE : PLP found long required int

1. Whenever we are assigning integral literal to the byte or short variable and if the value with in the range of byte or short, then compiler automatically treats it as byte literal

--

**Floating Literal**

By default, every float literal is of double type, and hence we can’t assign directly to float literal to double variable.

We can specify float literal explicitly as float type by suffixed with f or F.

We can specify explicitly float literal as double type by suffixed with d or D of course this convention is not required

EX

|  |
| --- |
| **public** **class** FloatLiteral {  **float** f1 = 123.456;//CE : can't convert from double to float  **float** f2 = 123.456f;  **double** d1 = 123.456;  **double** d2 = 123.456f;  **double** d3 = 123.456d;  } |

We can specify float literals only in decimal form. Octal and hexadecimal forms are not allowed.

|  |
| --- |
| **public** **class** FloatLiteral {  **double** d1 = 123.456;  **double** d2 = 0123.456;// not valid  **double** d3 = 0x123.456; //not valid  } |

We can assign int literal directly to the float variable and that int can be specified either in decimal or octal or hexadecimal form.

|  |
| --- |
| **public** **class** FloatLiteral {  **double** d1 = 786;  **double** d2 = 0777;  **double** d3 = 0xface;  **double** d4 = 0xface.0; //not valid  **double** d5 = 0786.0;  } |

We can’t assign float literals to the int types

|  |
| --- |
| **public** **class** FloatLiteral {  **int** x = 10.5;  } |

We can specify floating point literal even in exponential form (also known as scientific notation)

|  |
| --- |
| **public** **class** FloatLiteral {  **public** **static** **void** main(String args[]) {  **double** d1 = 1.2e3;  System.***out***.println(d1); // 1200.0  **float** f1 = 1.2e3; // possible loss of precision  **float** f2 = 1.2e3f;  }  } |

Boolean literal

The only allowed values for Boolean datatypes are true or false, where case should be in lower case.

|  |
| --- |
| **public** **class** Booleanliteral {  **public** **static** **void** main(String[] args) {  **boolean** b1;  **boolean** b2 = **false**;  **boolean** b3 = **true**;  **boolean** b4 = 0; // not valid  **boolean** b5 = “**true**”;  }  } |

|  |  |
| --- | --- |
| Int x = 0;  If (x){  Sop(“hello”);  }  else {  sop(“Hi”);  } | While (1) {  Sop(“Hello”);  } |

// here if, while expects only Boolean type, either true or false.

Char literal

A char literal can be specified as single character with in single quotes

char c1 = ‘a’;

char c2 = “a”;

char c3 = a;

char c4 = ‘ab’;

we can specify char literal as integral literal which represents UNICODE value of that character.

The integral can be specified either in decimal, octa, hexadecimal forms.

The allowed range is 0 to 65535

char c1 = 97;

sop(c1) // a

char c2 = 65535;

char c3 = 65536;

char c4 = oxFace;

char c5 = 0777;

char c6 = 0xbeer;

char c7 = 0786;

Unicode representation of char literal

We can represent char literal even in Unicode representation which is nothing but‘\uxxxx’

Ex

char c1 = ‘\u0061’;

char c2 = ‘\iface’;

char C3 = ‘\ubeef’;

Every escape character in java is a char literal

char nl = ‘\n’;

char nt = ‘\t’;

char c1 = ‘\m’

String Literal

Enhancements with respect to literals