**byte**

Size : 1 byte (8 bits)

Range : - 128 to 127

MSB LSB

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| X |  |  |  |  |  |  |  |

**MSB** - Most Significant Bit (Highest Order Bit)

**LSB** - Least Significant Bit (Lowest Order Bit)

1) X = Reversed

Signed bit - 0 is positive and 1 is negative

2) Remaining 7 bits represents the value

**Maximum value = 127**

MSB LSB

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

26 25 24 23 22 21 20

64 + 32 + 16 + 8 + 4 + 2 + 1 **= 127**

**Minimum value = -128**

MSB LSB

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

* **(10000000)2  = - (128)10**

**2) short**

**Size : 2 bytes (16 bits)**

**Range : - 215 to 215 – 1**

**- 32768 to 32767**

**3) int**

**Size : 4 bytes (32 bits)**

**Range : - 231 to 231 – 1**

**- 2147483648 to 2147483647**

**4) long**

**Size : 8 bytes (64 bits)**

**Range : - 263 to 263 – 1**

**Keywords**

**- In any language, some words are reserved for some meaning or functionality.**

**- If reserved words are associated with some functionality, it is called as keywords.**

**- If reserved words is representing only values, it is called as reserved literals.**

**Used Keywords (48)**

**1) Keywords for datatypes (8)**

**byte int float boolean**

**short long double char**

**2) Keywords for flow control (11)**

**if switch default do**

**break return else case**

**while for continue**

**3) Keywords for modifiers (11)**

**public private protected**

**static abstract final native**

**synchronized (For objects)**

**strictfp (Restricts floating point calculations - 1.2V)**

**transient (Marks a member variable not to be serialized)**

**volatile (The value of the variable will never be cached thread locally)**

**4) Keywords for exception handling (6)**

**try finally throws**

**catch throw assert (Defines statement)**

**5) class related keywords (6)**

**class extends package**

**interface implements import**

**6) object related keywords (4)**

**new super**

**this instanceof**

**7) Return type keyword (1)**

**void**

**8) Group of named constants (1)**

**enum**

**Unused Keywords (2)**

**1) goto - Using it creates several problems in old langauages. Hence, banned in Java.**

**2) const - We can use final keyword instead of const.**

**NOTE - If we use unused keywords then COMPILE TIME ERROR is thrown by JVM.**

**Reserved Literals (3)**

**- true & false are values for boolean datatype**

**- null is a default value for object reference**

**Intergal Data type**

**1) byte**

**Size : 1 byte (8 bits)**

**Range : - 128 to 127**

**2) short**

**Size : 2 bytes (16 bits)**

**Range : - 32768 to 32767**

**3) int**

**Size : 4 bytes (32 bits)**

**Range : - 2147483648 to 2147483647**

**4) long**

**Size : 8 bytes (64 bits)**

**Range : Refer MS-Word Document**

**1) float - Single Precision (5 to 6 decimal place accuracy)**

**Example : 10.0/3 = 3.333333.......**

**Accuracy : Less**

**Size : 4 bytes (32 bits)**

**Range : -3.4e38 to 3.4e38**

**2) double - Double Precision (14 to 15 decimal place accuracy)**

**Example : 10.0/3 = 3.333333333333333.......**

**Accuracy : More**

**Size : 8 bytes (64 bits)**

**Range : -1.7e38 to 1.7e38**

**1) boolean (true/false)**

**Size - N.A. (Not Applicable)**

**Range - N.A. (Not Applicable)**

**2) char**

**Size - 2 bytes (16 bits) - Java**

**Range - 0 to 65535**

**(NOTE - Size of char in C/C++ is 1 byte) - Interview Question**

**Q. Why the size of char is different in Java and C/C++?**

**Ans.**

**1) C/C++ - ASCII code based language (Less than or equals 256 characters)**

**2) Java - Unicode based language (Greater than 256 characters)**

**Looping staements**

**1) do loop / do-while loop**

**initialization;**

**do**

**{**

**Body of the loop**

**}**

**while (<TestCodition>);**

**2) while loop**

**initialization;**

**while (<TestCodition>)**

**{**

**Body of the loop**

**}**

**3) for loop**

**for (<Initialization>;<TestCodition>;<Operation>)**

**{**

**Body of the loop**

**}**

**Branching statements**

**Branching/ Decision Making Statements/ Controls Statements**

**1) Conditional or Ternary Operator**

**<Condition> ? <Exp1> : <Exp2>**

**2) if Statements**

**a) Simple if statement**

**if(<Test>)**

**Statement;**

**OR**

**if(<Test>)**

**{**

**Statements;**

**}**

**b) if .... else statement**

**if(<Test>)**

**{**

**Statements; // TRUE Block**

**}**

**else**

**{**

**Statements; // FALSE Block**

**}**

**c) Nesting of if .... else statements**

**if(<Test>)**

**{**

**if(<Test>)**

**{**

**Statements;**

**}**

**else**

**{**

**Statements;**

**}**

**}**

**else**

**{**

**Statements;**

**}**

**d) else if Ladder**

**if(<Test1>)**

**{**

**Statement(s)\_1;**

**}**

**else if (<Test2>)**

**{**

**Statement(s)\_2;**

**}**

**else if (<Test3>)**

**{**

**Statement(s)\_3;**

**}**

**else if (<TestN>)**

**{**

**Statement(s)\_N;**

**}**

**else**

**{**

**Default Statement(s);**

**}**

**3) switch statement**

**switch(<Expression>)**

**{**

**case <Value1>:**

**Statement1;**

**break;**

**case <Value2>:**

**Statement2;**

**break;**

**case <ValueN>:**

**StatementN;**

**break;**

**default:**

**Statement;**

**break;**

**}**