*Literals in Java*

1. **Explain about Literal?**

* Any constant value which can be assigned to the variable is called as literal/constant.
* // Here 100 is a constant/literal.
* int x = 100;

1. **How many types of literals in java?**

* There are 5 different types of literals in java they are
  + Integral literals
  + Floating point literals
  + Character literals
  + Boolean literals
  + String literals

1. **Explain about Integral literals?**

* For Integral data types (byte, short, int, long), we can specify literals in 4 ways:-

**Decimal literals (Base 10) :**

* + In this form the allowed digits are 0-9.
  + **Example**: int x = 101;

**Octal literals (Base 8) :**

* + In this form the allowed digits are 0-7.
  + The octal number should be prefix with 0.
  + **Example**: int x = 0146;

**Hexa-decimal literals (Base 16) :**

* + In this form the allowed digits are 0-9 and characters are a-f.
  + We can use both uppercase and lowercase characters.
  + As we know that java is a case-sensitive programming language but here java is not case-sensitive.
  + The hexa-decimal number should be prefix with 0X or 0x.
  + **Example**: int x = 0X123Face;

**Binary literals:**

* + From 1.7 onward we can specify literals value even in binary form also, allowed digits are 0 and 1. Literals value should be prefixed with 0b or 0B.
  + **Example**: int x = 0b1111;

1. **Write a java program using integral literals?**

|  |
| --- |
| // Java program to illustrate the application of Integer literals  public class Test {      public static void main(String[] args)      {          int a = 101; // decimal-form literal          int b = 0100; // octal-form literal          int c = 0xFace; // Hexa-decimal form literal          int d = 0b1111; // Binary literal          System.out.println(a);          System.out.println(b);          System.out.println(c);          System.out.println(d);      }  }  OUTPUT |

101

64

64206

15

* **NOTE :**
  + By default, every literal is of int type, we can specify explicitly as long type by suffixed with l or L.
  + There is no way to specify byte and short literals explicitly but indirectly we can specify.
  + Whenever we are assigning integral literal to the byte variable and if the value within the range of byte then the compiler treats it automatically as byte literals.

1. **Explain about Floating-Point literal**

* For Floating-point data types, we can specify literals in only decimal form and we cant specify in octal and Hexa-decimal forms.  
  **Decimal literals(Base 10) :** 
  + In this form the allowed digits are 0-9.
  + **Example:** double d = 123.456;

1. **Write a program using Floating point literals?**

|  |
| --- |
| // Java program to illustrate the application of  // floating-point literals  public class Test {      public static void main(String[] args)      {          float a = 101.230; // decimal-form literal          float b = 0123.222; // It also acts as decimal literal          float c = 0x123.222; // Hexa-decimal form          System.out.println(a);          System.out.println(b);          System.out.println(c);      }  }  OUTPUT: |

101.230

123.222

Error: malformed floating point literal

* **NOTE:** 
  + By default every floating point literal is of double type and hence we cant assign directly to float variable.
  + But we can specify floating point literal as float type by suffixed with f or F.
  + We can specify explicitly floating point literal as double type by suffixed with d or D. Of course this convention is not required.

1. **Explain about Char literal?**

* For char data types we can specify literals in 4 ways:

**Single quote :**

* + We can specify literal to char data type as single character within single quote.
  + **Example** :char ch = 'a';

**Char literal as Integral literal :**

* + we can specify char literal as integral literal which represents Unicode value of the character and that integral literals can be specified either in Decimal, Octal and Hexadecimal forms. But the allowed range is 0 to 65535.
  + **Example** :char ch = 062;

**Unicode Representation :**

* + We can specify char literals in Unicode representation ‘\uxxxx’. Here xxxx represents 4 hexadecimal numbers.
  + **Example** :char ch = '\u0061';// Here /u0061 represent a.

**Escape Sequence :**

* + Every escape character can be specify as char literals.
  + **Example**: char ch = '\n';

1. **Write an example program using char literal?**

|  |
| --- |
| // Java program to illustrate the application of char literals  public class Test {  public static void main(String[] args)      {          char ch = 'a'; // signle character literl within signle quote          char b = 0789; // It is an Integer literal with octal form          char c = '\u0061'; // Unicode representation          System.out.println(ch);          System.out.println(b);          System.out.println(c);          // Escape character literal          System.out.println("\"  is a symbol");      }  }  OUTPUT: |

a

error:Integer number too large

a

" is a symbol

1. **Explain about String literal?**

* Any sequence of characters within double quotes is treated as String literals.
  + Example: String s = "Hello";
* String literals may not contain unescaped newline or linefeed characters. However, the Java compiler will evaluate compile time expressions, so the following String expression results in a string with three lines of text:
  + Example:

String text = "This is a String literal\n"

+ "which spans not one and not two\n"

+ "but three lines of text.\n";

1. **Write a program using String literals?**

|  |
| --- |
| // Java program to illustrate the application of String literals  public class Test {      public static void main(String[] args)      {          String s = "Hello";            // If we assign without "" then it treats as a variable          // and causes compiler error          String s1 = Hello;            System.out.println(s);            System.out.println(s1);      }  }  **OUTPUT:** |

Hello

error: cannot find symbol

symbol: variable Hello

location: class Test

1. **Explain about boolean literals?**

* Only two values are allowed for Boolean literals i.e. true and false.

boolean b = true;

|  |
| --- |
| // Java program to illustrate the application of boolean  // literals  public class Test {      public static void main(String[] args)      {          boolean b = true;          boolean c = false;          boolean d = 0;          boolean b = 1;          System.out.println(b);          System.out.println(c);          System.out.println(d);          System.out.println(e);      }  } |

true

false

error: incompatible types: int cannot be converted to boolean

error: incompatible types: int cannot be converted to boolean

* **NOTE :** 
  + When we are performing concatenation operations, then the values in brackets are concatenated first.
  + Then the values are concatenated from the left to the right.
  + We should be careful when we are mixing character literals and integers in String concatenation operations and this type of operations are known as Mixed Mode operation.

1. **Write a program using Literals?**

|  |
| --- |
| // Java program to illustrate the behaviour of  // char literals and integer literals when  // we are performing addition  public class Test {      public static void main(String[] args)      {          // ASCII value of 0 is 48          int first = '0';            // ASCII value of 7 is 55          int second = '7';          System.out.println("Geeks!" + first +                                  '2' + second);      }  } |

Output:

Geeks!48255

* **Explanation :**
* Whenever we are performing addition between a string and integer then the overall result is converted into string.
* In the above program evaluation is done in the following way:

"Geeks!" + first + '2' + second

"Geeks! " + 48 + '2' + 55

"Geeks!48" + '2' + 55

"Geeks!482" + 55

"Geeks!48255"