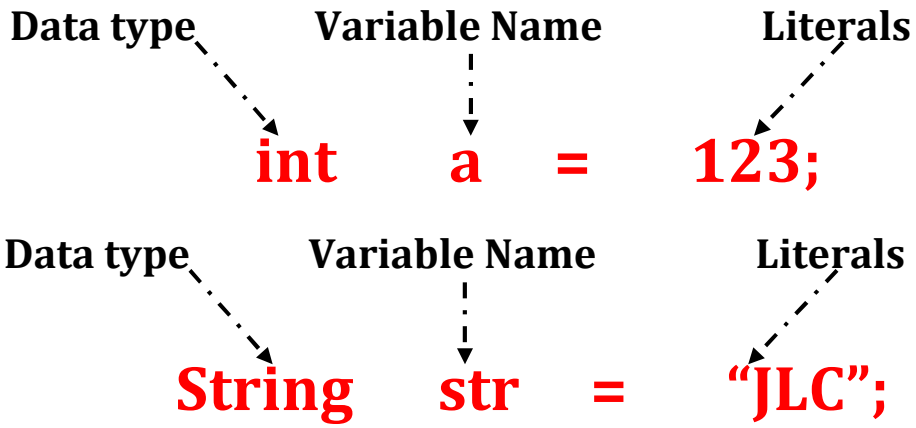
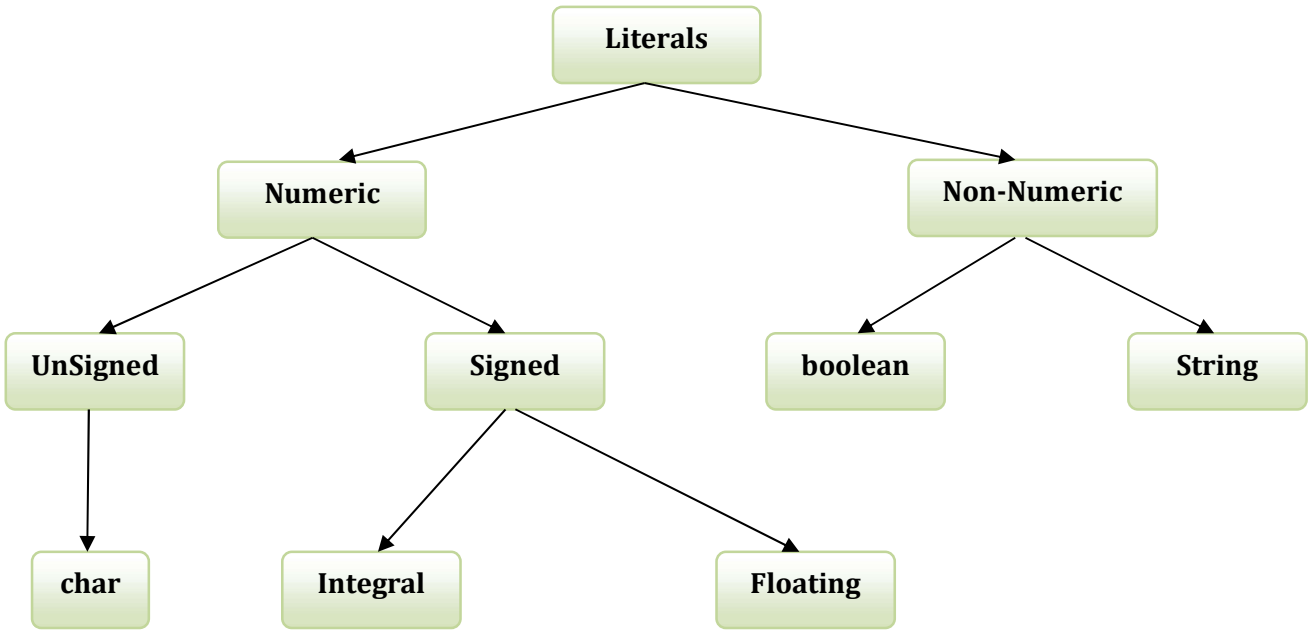
2.7 Literals

- Literals are the actual values.
- Literals will be assigned to variables or constants.
- Literals are also used to perform any operations.
- Constant values in programs are called as Literals.





Types of Literals

1) Boolean Literals
2) Character Literals
3) String Literals
4) Integral Literals
5) Floating Literals
6) null Literal

2.7.1 Boolean Literals

- Boolean Literal can be assigned to a boolean type variable.
- There are two boolean literals
  1) true 2) false

Lab18.java

```
class Lab18{
public static void main(String as[]) {
boolean b1 = true;
boolean b2 = false;
System.out.println(b1);
System.out.println(b2);
}
}
```

Lab19.java

```
class Lab19{
public static void main(String as[]) {
boolean b = 1;
System.out.println(b);
}
}
```

Lab20.java

```
class Lab20{
public static void main(String as[]) {
boolean True = false;
boolean b = True;
System.out.println(b);
}
}
```

2.7.2 Character Literals

- Character Literal can be assigned to char type variable.
- A char type variable can hold following:
  - Single character enclosed in single quotation marks
  - Escape Sequence
  - ASCII Value
  - UNICODE Character
  - Octal Character

Lab21.java

```
class Lab21{
public static void main(String as[]) {
char ch1 = 'a';                 char ch2 = 'J';
char ch3 = '5';                 char ch4 = '*';
System.out.println(ch1);
System.out.println(ch2);
System.out.println(ch3);
System.out.println(ch4);
} }
```

Lab22.java

```
class Lab22{
public static void main(String as[]) {
char ch1='';        // Empty
char ch2='  ';      // Two Space
char ch3='AB';
char ch4 = '\';
char ch5='';
}
}
```

Lab23.java

```
class Lab23{
public static void main(String as[]) {
char ch1 = ' ';                 // One Space
char ch2 = '        ';          // Tab Key
char ch3 = '/';
System.out.println(ch3);
} }
```

2.7.2.1 Escape Sequence

- **ESCAPE SEQUENCE is a special notation which is used to represent some special characters which can't be represented as it is.**

Escape Sequence	Description
\t	Tab Space.
\b	Backspace.
\n	Newline.
\r	Carriage return.
\f	Formfeed.
\'	Single quote character.
\"	Double quote character.
\\	Backslash character.

Lab24.java

```
class Lab24{
public static void main(String as[]) {
char ch1 = '\'';
char ch2 = '\\';
System.out.println(ch1);
System.out.println(ch2);
} }
```

Lab25.java

```
class Lab25{
public static void main(String str[]) {
char ch1='\+';
char ch2='\p';
char ch = '\a';
System.out.println(ch);
}  }
```

Lab26.java

```
class Lab26{
public static void main(String str[]) {
char ch1='"';      // DOUBLE QUOTES in Single Quotes
char ch2 = '\"';   // slash followed by  Double Quotes
System.out.println(ch1);
System.out.println(ch2);
} }
```

Lab27.java

```
class Lab27{
public static void main(String str[]) {
String str="Welcome to "JLC"";
System.out.println(str);
}
}
```

Lab28.java

```
class Lab28{
public static void main(String as[]) {
String str="Welcome to \"JLC\"";
System.out.println(str);
} }
```

Lab29.java

```
class Lab29{
public static void main(String str[]) {
char ch='\u';
System.out.println(ch);
} }
```

2.7.2.2 ASCII Character Sets

- **ASCII stands for American Standard Code for Information Interchange.**
- **Every character enclosed in single quotation marks will have an integer equivalent value called as ASCII value.**
- **ASCII Value Range is 0 – 255.**
- **ASCII Value can be assigned to a char type variable.**

Lab30.java

```
class Lab30{
public static void main(String str[]) {
char ch='1';
System.out.println(ch);
} }
```

Lab31.java

```
class Lab31{
public static void main(String str[]) {
char ch=1;
System.out.println(ch);
} }
```

Lab32.java

```
class Lab32{
public static void main(String str[]) {
char ch1=65;      char ch2=48;      char ch3=97;
System.out.println(ch1);
System.out.println(ch2);
System.out.println(ch3);
} }
```

Lab33.java

```
class Lab33{
public static void main(String str[]) {
char ch1=0;                  char ch2=65535;
System.out.println(ch1);
System.out.println(ch2);
}
}
```

Lab34.java

```
class Lab34{
public static void main(String str[]) {
char ch1=-1;      char ch2=65536;
System.out.println(ch1);
System.out.println(ch2);
} }
```

2.7.2.3 UNICODE Characters

- UNICODE stands for UNIversal CODE.
- Every character will have UNICODE value.

UNICODE Notation

Syntax:

\uXXXX - X will be hexadecimal digit

Starts with \u followed by four hexadecimal digits.

UNICODE Range

\u0000 (0) to \uFFFF (65535)

Lab35.java

```
class Lab35{
public static void main(String str[]) {
char ch1 = '\u0061';
char ch2 = '\u0062';
System.out.println(ch1);
System.out.println(ch2);
char ch3 = '\u0041';
char ch4 = '\u0042';
System.out.println(ch3);
System.out.println(ch4);
} }
```

2.7.2.4 Octal Value as char type

- Octal value can be assigned to char type.

OCTAL Notation

Syntax:

\DDD - D will be octal digit

Starts with \ followed by three octal digits.

OCTAL Range

Range in Decimal	0	-	255
Range in Octal	\0	-	\377

Lab36.java

```
class Lab36{
public static void main(String str[]) {
char ch1 = 'A';              char ch2 = 65;
System.out.println(ch1);
System.out.println(ch2);
char ch3 = '\u0041';         char ch4 = '\101';
System.out.println(ch3);
System.out.println(ch4);
} }
```

Character	ASCII	OCTAL	UNICODE	Character	ASCII	OCTAL	UNICODE
0	48	060	\u0030	5	53	065	\u0035
1	49	061	\u0031	6	54	066	\u0036
2	50	062	\u0032	7	55	067	\u0037
3	51	063	\u0033	8	56	070	\u0038
4	52	064	\u0034	9	57	071	\u0039

Character	ASCII	OCTAL	UNICODE	Character	ASCII	OCTAL	UNICODE
A	65	101	\u0041	a	97	141	\u0061
B	66	102	\u0042	b	98	142	\u0062
C	67	103	\u0043	c	99	143	\u0063
D	68	104	\u0044	d	100	144	\u0064
E	69	105	\u0045	e	101	145	\u0065
F	70	106	\u0046	f	102	146	\u0066
G	71	107	\u0047	g	103	147	\u0067
H	72	110	\u0048	h	104	150	\u0068
I	73	111	\u0049	i	105	151	\u0069
J	74	112	\u004A	j	106	152	\u006A
K	75	113	\u004B	k	107	153	\u006B
L	76	114	\u004C	l	108	154	\u006C
M	77	115	\u004D	m	109	155	\u006D
N	78	116	\u004E	n	110	156	\u006E
O	79	117	\u004F	o	111	157	\u006F
P	80	120	\u0050	p	112	160	\u0070
Q	81	121	\u0051	q	113	161	\u0071
R	82	122	\u0052	r	114	162	\u0072
S	83	123	\u0053	s	115	163	\u0073
T	84	124	\u0054	t	116	164	\u0074
U	85	125	\u0055	u	117	165	\u0075
V	86	126	\u0056	v	118	166	\u0076
W	87	127	\u0057	w	119	167	\u0077
X	88	130	\u0058	x	120	170	\u0078
Y	89	131	\u0059	y	121	171	\u0079
Z	90	132	\u005A	z	122	172	\u007A

Lab37.java

```
class Lab37{
public static void main(String str[]) {
char ch1 = '\101';
char ch2 = '\103';
System.out.println(ch1);
System.out.println(ch2);
}
}
```

Lab38.java

```
class Lab38{
public static void main(String str[]) {
char ch1=\101;
char ch2='\420';
}
}
```

2.7.3 String Literals

- String Literal is a collection of zero or more characters enclosed between double quotation marks.
- String Literals can be assigned to reference variable of type String.
- String Literal will be represented as char array in memory.

Lab39.java

```
class Lab39{
public static void main(String str[]) {
String str1 = null;
System.out.println(str1);
int x=str1.length();
System.out.println(x);
}
}
```

Lab40.java

```
class Lab40{
public static void main(String str[]) {
String str1 ="";
System.out.println(str1);
int x=str1.length();
System.out.println(x);
String str2 ="JLCINDIA";
System.out.println(str2);
int y=str2.length();
System.out.println(y);
String str3 ="JLC99@india";
System.out.println(str3);
int z=str3.length();
System.out.println(z);
}
}
```

Lab41.java

```
class Lab41{
public static void main(String str[]) {
String dir1 = "D:\new\test\batch";
System.out.println(dir1);
String dir2 = "D:\\new\\test\\batch";
System.out.println(dir2);
}
}
```

SUMMARY

Boolean Literals

1) A boolean type variable can hold either true or false.
2) A boolean type variable can't hold numeric value.
    a. 0 not equals to false
    b. Non Zero values not equals to true
3) Boolean literals - true and false are reserved words and must be in lower case.
    a. true not same as TRUE, True, TrUe etc
    b. false not same as FALSE, False, FaLsE etc

Character Literals

4) A char type variable can hold following:
    a. Single character enclosed in single quotation marks
    b. Escape Sequence
    c. ASCII Value
    d. UNICODE Character
    e. Octal Character
5) SPACE is a valid character so it can be placed in single quotation marks.
6) TAB is a valid character so it can be placed in single quotation marks.
7) Backslash character (\) is used to form escape sequence so only backslash character can't be placed in single quotation marks.
8) Single Quote is used to form character literal so only single quote can't be placed in single quotation marks.
9) There are some characters like Backslash, Single quote which can't be placed in single quotes as it is. Those special characters have to be represented as Escape Sequence.
10) Every character enclosed in single quotation marks will have an integer equivalent value called as ASCII value.
11) char type variable can hold integer value ranging from 0 - 65535 only.
12) char type variable can hold UNICODE value ranging from \u0000 - \uFFFF only.
13) char type variable can hold OCTAL value ranging from \0 - \377 only.
14) When you print the char type variable which holds Integer, UNICODE or Octal value then it displays corresponding character representation.

15) If Integer, UNICODE value don't have character representation then it displays ? which indicates no character defined for that value.

16) When you assign character literals to int type variable then internally ASCII value of that character will be assigned.

String Literals

17) Empty String Literal is allowed whereas empty character literal is not allowed.

18) If reference variable of String type contains null value then it is called as NULL String.

19) You can't refer any members with NULL String. If you do so it results in NullPointerException.

20) If reference variable of String type contains zero character in double quotations marks then it is called as EMPTY String.

21) You can use escape sequence to represent UNICODE/Octal value as it is in String literals.

Assignment #2

Q1) What is a literal? How many types of literals available?

Q2) How many boolean literals are available?

Q3) What will happen when I assign 1 to boolean type variable?

Q4) Can we store empty character in char type variable?

Q5) How to store single quote in char variable?

Q6) What is Escape Sequence?

Q7) Which of the following are valid escape character in Java?

A) \\ B) \a C) \p D) \d

E) \n F) \' G) \j H) \?

Q8) What is ASCII value and What is the range of ASCII value?

Q9) What are the ASCII values of Upper case Alphabets?

Q10) What are the ASCII values of Lower case Alphabet?

Q11) What are the ASCII values of Digits?

Q12) Can I store negative value in char type variable?

Q13) What is the format of UNICODE character?

Q14) Can I store UNICODE Character in the char type variable?

Q15) Can I declare the variable using UNICODE representation?

Q16) What are the UNICODE values of Upper case Alphabets?

Q17) What are the UNICODE values of Lower case Alphabet?

Q18) What are the UNICODE values of Digits?

Q19) What is the difference between ASCII and UNICODE Character sets?

Q20) Can I store Octal representation in the char type variable?

Q21) What is the format of Octal representation that can be stored in char type variable?

Q22) What is the range of Octal representation that can be stored in char type variable?

Q23) What are the Octal representations of Upper case Alphabets?

Q24) What are the Octal representations of Lower case Alphabet?

Q25) What are the Octal representations of Digits?

Q26) What are the various types of representations can be stored in char type variable?

Q27) What is a String Literal?

Q28) Can I represent String Literal using Single quotes?

Q29) What is empty String literal ?

Q30) What is the difference between Empty string and null String?

Q31) How to get the length of String Literal?

Q32) What will be displayed when UNICODE value is found in String Literal?

```
String str="UNICODE of A is \u0041";
```

Q33) What will be displayed when Octal representation is found in String Literal?

```
String str="Octal of A is \101";
```

Q34) What will happen when Escape Sequence is found in String Literal?

```
String str="Hello\nGuys";
```

Practice Test #2

Sr. No	Question	Options	Answer

1	public class Mobile{
	public static void main(String[] args) {
	double price;
	String model;
	System.out.println(model + price);
	}
	}
		A) null0
		B) null0.0
		C) 0
		D) 0.0
		E) Compilation error	

2	class Test2{
	public static void main(String as[]) {
	boolean b = 0;
	System.out.println('0');
	}
	}
		A) 0
		B) false
		C) true
		D) Runtime Error
		E) Compilation error	

3	class Test3{
	public static void main(String as[]) {
	boolean faLse = true;
	boolean b = faLse;
	System.out.println(b);
	}
	}
		A) 0
		B) false
		C) true
		D) Runtime Error
		E) Compilation error	

4	class Test4 {
	public static void main(String[] args) {
	char ch='/';
	System.out.println(ch);
	}
	}
		A) 0
		B) /
		C) //
		D) Runtime Error
		E) Compilation error	

5	class Test5 {
	public static void main(String[] args) {
	char ch='\';
	System.out.println(ch);
	}
	}
		A) 0
		B) \
		C) \\
		D) Runtime Error
		E) Compilation error	

6	class Test6 {
	public static void main(String[] args) {
	char ch='A';
	System.out.println(ch+1);
	}
	}
		A) A
		B) Compilation error
		C) B
		D) 65
		E) 66	

7	class Test7 {
	public static void main(String[] args) {
	char ch=65535;
	System.out.println(ch+1);
	}
	}
		A) 0
		B) Compilation error
		C) 65535
		D) 65536
		E) None of above	

8	class Test8 {
	public static void main(String[] args) {
	char ch=0;
	System.out.println(ch-1);
	}
	}
		A) 0
		B) Compilation error
		C) -0
		D) -1
		E) None of above	

9	class Test9 {
	static String name;
	public static void main(String str[]) {
	System.out.println(name);
	System.out.println(name.length());
	}
	}
		A) null 0
		B) Compilation error
		C) null
		and then Runtime Exception
		D) Nothing will be Displayed
		E) null 4	

10	class Test10 {
	static String name="null";
	public static void main(String str[]) {
	System.out.println(name);
	System.out.println(name.length());
	}
	}
		A) null 0
		B) Compilation error
		C) null
		and then Runtime Exception
		D) Nothing will be Displayed
		E) null 4	

11	class Test11 {
	public static void main(String str[]) {
	String name="UNICODE OF A IS
	\u0041";
	System.out.println(name);
	} }
		A) UNICODE OF A IS \u0041
		B) UNICODE OF A IS A
		C) UNICODE OF A IS 65
		D) Compilation error	

12	class Test12 {
	public static void main(String str[]) {
	String data="Value is \61";
	System.out.println(data);
	} }
		A) Value is 1
		B) Value is 61
		C) Value is 65
		D) Value is 49
		E) Compilation error	

13	class Test13 {
	public static void main(String str[]) {
	String dir="D:/java/Hello.java";
	System.out.println(dir);
	} }
		A) Compilation error
		B) D:/java/Hello.java
		C) D:\java\Hello.java
		D) Runtime error	

2.7.4 Integer Literals

- Integer Literals can be assigned to one of the integral data type -byte, short, int, long.
- Integer Literal size is depending on integral data type you are using.
- Default type of Integer Literal is int.
- There are four types of Integer Literals:
  - Decimal Literals
  - Octal Literals
  - Hexadecimal Literals
  - Binary Literals (From Java 7)

2.7.4.1 Decimal Literals

- Decimal Literal is a valid number from decimal number system.
- The base or radix of decimal number system is 10 i.e only 10 digits ranging from 0 to 9 are allowed to form the decimal literals.
- Decimal literals must not start with 0.
  - 12345
  - 98787898

Lab42.java

```
class Lab42{
public static void main(String str[]) {
byte b= 65;
System.out.println(b);
short s = 1234;
System.out.println(s);
int i = 123431;
System.out.println(i);
}
}
```

Lab43.java

```
class Lab43{
public static void main(String str[]) {
byte b= 128;
short s = 32768;
int i = 2147483648;
long a= 2147483648;
}
}
```

Lab44.java

```
class Lab44{
public static void main(String str[]) {
long a= 2147483648L;
System.out.println(a);
}
}
```

Lab45.java

```
class Lab45{
public static void main(String str[]) {
System.out.println(9880979999);
}
}
```

Lab46.java

```
class Lab46{
public static void main(String str[]) {
System.out.println(9880979999L);
}
}
```

Lab47.java

```
class Lab47{
public static void main(String str[]) {
int a=099;
System.out.println(a);
} }
```

2.7.4.2 Octal Literals

- Octal Literal is a valid number from octal number system.
- The base or radix of octal number system is 8 i.e only 8 digits ranging from 0 to 7 are allowed to form the octal literals.
- Octal Literal must start with 0 (ZERO).
    - 0101
    - 0377
    - 0345245

Lab48.java

```
class Lab48{
public static void main(String str[]) {
int a=1010;
int b=0101;
System.out.println(a);
System.out.println(b);
}
}
```

2.7.4.3 Hexadecimal Literals

- Hexadecimal Literal is a valid number from hexadecimal number system.
- The base or radix of hexadecimal number system is 16 i.e only 16 digits ranging from 0 to 9 and A/a to F/f are allowed to form the hexadecimal literals.
- Hexadecimal Literal must start with 0X / 0x.
    - 0x123af

Lab49.java

```
class Lab49{
public static void main(String str[]) {
int a=0xA;
int b=0X61;
int c=0Xface;
System.out.println(a);
System.out.println(b);
System.out.println(c);
} }
```

2.7.4.4 Binary Literals (From Java 7)

- Binary Literal is a valid number from binary number system.
- The base or radix of binary number system is 2 i.e. only 2 digits 0 and 1 are allowed to form the binary literals.
- Binary Literal must start with 0B / 0b.
    - 0b1010
    - 0B1111001101

Lab50.java

```
class Lab50{
public static void main(String str[]) {
int a = 101;
int b = 0b101;
System.out.println(a);
System.out.println(b);
} }
```

Lab51.java

```
class Lab51{
public static void main(String str[]) {
int a = 0b102;
System.out.println(a);
}
}
```

Lab52.java

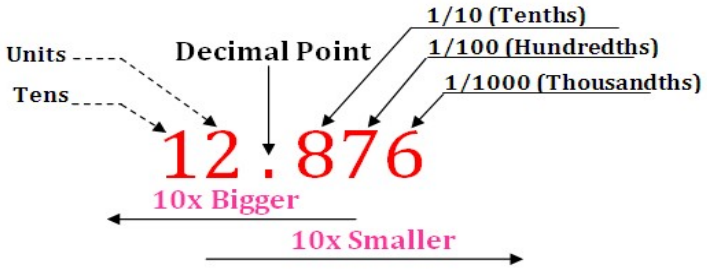
```
class Lab52{
public static void main(String str[]) {
int a = 101;        int b=0101;
int c=0X101;        int d=0B101;
System.out.println(a);
System.out.println(b);
System.out.println(c);
System.out.println(d);
} }
```

2.7.5 Floating Point Literals

- Floating Point Literal can be assigned to one of the floating point data type –float, double.
- Floating Point Literal size is depending on floating point data type you are using.
- Default type of Floating Point Literal is double.
- You can use D/d as a suffix for the double value optionally.
- You must use F/f as a suffix for float value.
- Floating Point Literals can be represented using following notations
    - Standard Floating Point Notation
    - Exponent/Scientific Notation
    - Hexadecimal Notation (From Java 5)

- It consists of a whole number followed by a decimal point and fractional part.

  12987.63
  129.8763



2.7.5.1 Standard Floating Point Notation

Lab53.java

```
class Lab53{
public static void main(String str[]) {
double d1 = 1234.567;
double d2 = 123.4567;
System.out.println(d1);
System.out.println(d2);
} }
```

Lab54.java

```
class Lab54{
public static void main(String str[]) {
float f1=12987.63;
System.out.println(f1);
}
}
```

Lab55.java

```
class Lab55{
public static void main(String str[]) {
float f1=12987.63F;
System.out.println(f1);
} }
```

2.7.5.2 Scientific / Exponent Notation

- **The exponent is indicated by an E or e followed by a decimal number, which can be positive or negative.**

123.4567e2	-> $123.4567 * 10^{2}$
123.4567E+2	-> $123.4567 * 10^{2}$
12345.67e-2	-> $12345.67 * 10^{-2}$

Lab56.java

```
class Lab56{
public static void main(String str[]) {
float f = 129.8763e2F;
double d1 = 129.8763e+2;
double d2 = 12987.63e-2;
System.out.println(f);
System.out.println(d1);
System.out.println(d2);
} }
```

2.7.5.3 Hexadecimal floating-point literals (FROM JAVA 5)

- It allows literals of the float and double types to be written primarily in base 16 rather than base 10.
  double x = 0XaP0;
  a -> 10
  // 10 * 2^0 = 10.0

Lab57.java

```
class Lab57{
public static void main(String str[]) {
double d1 = 0XaP0;
System.out.println(d1);
double d2 = 0XaP1;
System.out.println(d2);
double d3 = 0XaP2;
System.out.println(d3);
double d4=0XA9.0P0;
double d5=0XA9.P0;
System.out.println(d4);
System.out.println(d5);
}
}
```

Lab58.java

```
class Lab58{
public static void main(String str[]) {
double d1 = 0Xa12.12;
System.out.println(d1);
}
}
```

2.7.6 null literal

- null is a value.
- It is default value for any reference variable.
- If the value of reference variable is null then it indicates that address/reference is not available in the variable.

Lab59.java

```
class Lab59{
static String str1;
public static void main(String str[]) {
String str2=null;
System.out.println(str1);
System.out.println(str2);
}
}
```

2.7.7 Underscore in Numeric Literals

- To represent the unit of the values you can use comma (,) symbol in number representation.

  Ex:

  2,45,12,452

- You can do the same task with UNDERSCORE (_) from Java 7 onwards.

  Ex:

  2_45_12_452

Lab60.java

```
class Lab60{
public static void main(String str[]) {
int a=2_45_12_452;
System.out.println(a);
}
}
```

Lab61.java

```
class Lab61{
public static void main(String str[]) {
int _12 = 9898;
System.out.println(_12);
int a= _12;
System.out.println(a);
} }
```

Lab62.java

```
class Lab62{
public static void main(String str[]) {
int a= _234;
System.out.println(a);
}
}
```

Lab64.java

```
class Lab64{
public static void main(String str[]) {
int a= 234_;
int b=0b_101;
int c=0X_1_A_F;
double d1=123_.45;
double d2=123._45;
}
}
```

Lab63.java

```
class Lab63{
public static void main(String str[]) {
int a=0_77;
System.out.println(a);
int b=0b1_01_01;
System.out.println(b);
int c=0X1_A_F;
System.out.println(c);
double d=1_2_3.4_5_6;
System.out.println(d);
}
}
```

SUMMARY

Integer Literals

1) There are four types of Integer Literals
   a. Decimal Literals
   b. Octal Literals
   c. Hexadecimal Literals
   d. Binary Literals (From Java 7)
2) Default type of Integer Literal is int.
3) Integer value ranging from -2147483648 to 2147483647 can be represented directly.
4) If you want to represent the integer value out of int range then you need to use L/l suffix.
5) Decimal literals must not start with 0.
6) Octal Literal must start with 0 (ZERO).
7) Hexadecimal Literal must start with 0X / 0x.
8) Binary Literal must start with 0B / 0b.
9) When Java compiler encounters Octal, Binary or Hexadecimal Literals then that will be replaced with decimal equivalent value.

Floating Point Literals

10) Default type of Floating Point Literal is double.
11) You can use D/d as a suffix for the double value optionally.
12) You must use F/f as a suffix for float value.
13) When Integer value starts with 0 then it will be considered as Octal Literals.

UNDERSCORE in Numeric Literals

14) You can use UNDERSCORE in numeric Literals to represent unit of values from Java 7.
15) When UNDERSCORE is used before first digit then it will be considered as a variable.
16) You can't use UNDERSCORE after the last digit.
17) In the case of Octal literal UNDERSCORE can be used just after the 0.
18) In the case of Binary literal UNDERSCORE can't be used just after the 0b/0B.
19) In the case of Hexadecimal literal UNDERSCORE can't be used just after the 0x/0X.
20) You can use more than one UNDERSCORE side by side between two digits.
21) You can use UNDERSCORE with floating point literals but not allowed just before or after the decimal point.

Assignment #3

Q1) What is Integral Literal ? How many types of Integral Literals are available?

Q2) What is the default data type of Integer Literal?

Q3) What is the Integer Literal newly added in Java 7?

Q4) What is the base/radix of Decimal Number System?

Q5) What are the digits allowed in Decimal Literal?

Q6) What is the range of int data type?

Q7) How can I store a value in long type of variable whose range is greater than int range?

Q8) What is the base/radix of Octal Number System?

Q9) What are the digits allowed in Octal Literal?

Q10) What will happen when I use 8 or 9 digits in Octal Literal?

Q11) What is Octal Literal representation?

Q12) What is the base/radix of Hexadecimal Number System?

Q13) What are the digits allowed in Hexadecimal Literal?

Q14) What is Hexadecimal Literal representation?

Q15) What is the base/radix of Binary Number System ?

Q16) What are the digits allowed in Binary Literal?

Q17) What will happen when I use other then 0 and 1 digits in Binary Literal?

Q18) What is Binary Literal representation?

Q19) What are the floating data types in Java?

Q20) What is the default floating data type?

Q21) When I need to use D/d suffix with floating values?

Q22) When I need to use F/f suffix with floating values?

Q23) What are the notations of Floating Point Literals?

Q24) What is the Floating Point notation newly added in Java 5?

Q25) Where can I use null literal?

Q26) What is the use of UNDERSCORE in digits?

Q27) Can I use UNDERSCORE before first digit? If so what it represents?

Q28) Can I use UNDERSCORE after last digit?

Q29) Can I use underscore after first 0 in Octal Literal?

Q30) Can I use underscore after 0B in Binary Literal?

Q31) Can I use underscore after 0X in Hexadecimal Literal?

Q32) Can I use two underscore side by side between two digits?

Q33) Can I use underscore with Floating Point Literals?

Q34) Can I use underscore just before or after decimal point in Floating Point Literals?

Q35) What are the new features added from Java 7 in Literals?

Practice Test #3

Sr. No	Question	Options	Answer

1	class Test1 {
	public static void main(String[] args) {
	float f1=11;
	float f2=11.f;
	f2=f1+f2;
	System.out.println(f2);
	}
	}
		A) 22
		B) 22.0
		C) 22.0f
		D) Compile error	

2	class Test2 {
	public static void main(String[] args) {
	byte b=016;
	short s=0x16;
	char c='1';
	int a=b+s+c;
	System.out.println(a);
	}
	}
		A) Compiler error
		B) Runtime exception
		C) 48
		D) 50
		E) 85	

3	class Test3 {
	public static void main(String[] args) {
	float f=12e-1F;
	final long l=12L;
	f=f+l;
	System.out.println(f);
	}
	}
		A) 13.2
		B) 18.9
		C) Compiler error
		D) Runtime exception	

4	class Hello {
	public static void main(String[] args) {
	int Hello = 13;
	int main = 34;
	System.out.println(Hello + main);
	}
	}
		A) 83
		B) Compilation fail
		C) 47
		D) Exception thrown at runtime	

5	class Test5{
	public static void main(String args[]){
	int a=7;
	System.out.println(a);
	}
	}
		A) 7
		B) Compilation fail
		C) 55
		D) Exception thrown at runtime	

6	class Test6{
	public static void main(String args[]){
	int a='7';
	System.out.println(a);
	}
	}
		A) 7
		B) Compilation fail
		C) 55
		D) Exception thrown at runtime	

7	class Test7{
	public static void main(String jlc[]){
	int ab=A;
	System.out.println(ab);
	}
	}
		A) A
		B) Compilation fail
		C) 65
		D) Exception thrown at runtime	

8	class Test8{
	public static void main(String jlc[]){
	int ab='A';
	System.out.println(ab);
	}
	}
		A) A
		B) Compilation fail
		C) 65
		D) Exception thrown at runtime	

9	class Test9{
	public static void main(String jlc[]){
	int ab=\u0037;
	System.out.println(ab);
	}
	}
		A) 7
		B) Compilation fail
		C) 55
		D) Exception thrown at runtime
		E) 37	

10	class Test10{
	public static void main(String jlc[]){
	int ab='\u0037';
	System.out.println(ab);
	} }
		A) 7
		B) Compilation fail
		C) 55
		D) Exception thrown at runtime
		E) 37	

11	class Test11{
	public static void main(String str[]) {
	double d = 061;
	System.out.println(d);
	}
	}
		A) 61
		B) Compilation fail
		C) 49
		D) 49.0
		E) 61.0	

12	class Test12{
	public static void main(String str[]) {
	double d = 061.0;
	System.out.println(d);
	}
	}
		A) 61
		B) Compilation fail
		C) 49
		D) 49.0
		E) 61.0	

13	class Test13{
	public static void main(String str[]) {
	double d = 0XE.P0;
	System.out.println(d);
	}
	}
		A) 0
		B) Compilation fail
		C) 14
		D) 14.0
		E) 61.0	

14	class Test14{
	public static void main(String str[]) {
	double d = 077;
	System.out.println(d);
	}
	}
		A) 63.0
		B) 77
		C) 77.0
		D) 63
		E) Compilation error	

15	class Test15{
	public static void main(String str[]) {
	double d = 077.0;
	System.out.println(d);
	}
	}
		A) 63.0
		B) 77
		C) 77.0
		D) 63
		E) Compilation error	

16	class Test16{
	public static void main(String str[]) {
	double d=0XA9;
	System.out.println(d);
	}
	}
		A) Runtime Error
		B) 169.0
		C) 169
		D) 19
		E) Compilation error	