Values and Data Types: 5, 6 and 7

Lesson 5

Literals

Literals are **values** that never change themselves during program run. E.g., **5, 2.5, "Alisha", 'A'**.
Literals are also known as **constants, values**, and **data**.

Various Kinds of Literals: General Concept

1) Integer Literal

Integer literals are **whole numbers** without any fractional parts, which means they do not contain decimal points. E.g.: **25**

In a program usually an integer literal is defined with **int** keyword. E.g.: **int** n = 25;

2) Floating Point Literal / Real Literal

Floating point literals are numbers having **fractional parts.** E.g.: **2.5**, **2.0** etc.
It is also known as **real literals**. In a program usually a real literal is defined with **double** keyword.
E.g.: **double** n = 2.5;

Rules for Writing Numeric Literals

Any characters including comma, space, Rs. etc. should not be used with a numeric literal. But for real literal a **.** (dot) can be used as decimal point.
E.g1: double a=30,000.0; is invalid because there is a comma.
E.g2: double rate=30%; is invalid because % sign should not be used.
E.g3: double amt=Rs.300000; is invalid because Rs. cannot be used.

Correct the errors (Write corrected statement):

double amt = Rs. 2,00,000.0;
double rate = 20%;

Answer:

double amt = 200000.0;
double rate=20.0; or double rate=20;

3) Character Literal

A character literal is **single character** enclosed in **single quotation** marks. E.g.: 'a', '1', '+', **';',** **' '**

In a program character literal is represented by the **char** keyword. E.g.: **char** c = 'A';

3. a) Non-graphic Characters / Non-printable / Non-initializable

Non-graphic characters are that cannot be initialized or printed directly. E.g.: **" "** **\ tab space enter space**.

The **System.out.println("Hello");** prints only **Hello**. The " " are not printed.
To print **"Hello"** (in double quotes) the System.out.println(**""Hello""**); is invalid.
To print **" "** it requires System.out.println("**\"**Hello**\"**");. Its output is **"Hello"**. The **** (back slash) should be preceded by both **"** and **"**. The **\"** is known an **escape sequence** character.
The String a = "Hello"; stores only **Hello** in the variable **a**. The " " are not stored. To store the **"Hello"** it requires String a = "**\"**Hello**\"**";.

3. b) Escape Sequence Character

An **escape sequence character** is a **sequence of characters start with backslash**. Escape sequence characters are used to represent non-graphic characters.

E.g.: **'\"'**.

Common Escape Sequence Characters

\" To use double quotation mark

E.g.: System.out.println("**\"Hello\"**");
Output: **"Hello"**

\' To use the single quotation mark

E.g.: System.out.println("**\'Hello\'**");

Output: 'Hello'

\n New line or Next line or line break

E.g.: System.out.println("Hello**\n**dear")

Output:

Hello

dear

\t Horizontal **tab space**

A tab space in Java is a group of **8** spaces. E.g.:

System.out.println("Anu**\t**99.0");

System.out.println("Nimisha**\t**98.5");

System.out.println("**\t**97.0");

Output:

Anu 99.0 (There are five spaces after Anu (8-3=5 spaces))

Nimisha 98.5 (There is 1 space after Nimisha (8-7=1 space))

97.0 (There are 8 spaces)

\u To print Unicode character

E.g.: System.out.println('**\u0041**');

Output: **A**

\0 To represent null character

E.g.: char c='**\0**'; (char c=''; two single quotes is invalid)

Its output is nothingness

**** To use backslash

E.g.: System.out.println("**C:**"); //System.out.println("C:\"); is error

Output: **C:**

Write output:

1) char a = '\u0042';
   System.out.println(a);
2) String s = "D:\\Birthday\\Cakecutting.jpg";
   System.out.println(s);
3) String s = "\"Johan\"";
   System.out.println(s);

Output:

1) B
2) D:\Birthday\Cakecutting.jpg
3) "Johan"

Write valid or invalid:

1) String a = ""Hello"";
2) String a = "\"Hello\"";
3) String a = "\"Hello"\";

Answer:

1) Invalid
2) Valid
3) Invalid

Write value of s:

1) String s = "\"Java\"";
2) String s = "E:\\Picture\\Flower.jpg";

Answer:

1) "Java"
2) E:\Picture\Flower.jpg

Answer the following:

1. Write statement to initialize: C:\Photos

2. Write statement to print: **"**Anu**"**

Answer:

1. String s = “C:\\Photos”;

2. System.out.println("**"**Anu**"**");

4) String Literal

A string literal is a **sequence** of **zero** or **more characters** enclosed by **double quotation** marks.

E.g.: "Hello", "", " ", "a", "1", "12.3", "+", "**;**"

Answer the following:

1) Write difference between **a** and **“a”**

2) Write difference between **2** and **“2”**

3) Write difference between **+** and **“+”**

4) Write difference between **‘n’** and **“n”**

5) Write difference between **;** and **“;”**

6) Write difference between **“”** and **“ ”**

Answer:

1) The **a** is a variable and the **“a”** is a string literal
2) The **2** is an integer literal and the **“2”** is a string literal
3) The **+** is an operator and the **“+”** is a string literal
4) The **‘n’** is a character literal and the **“n”** is a string literal
5) The **;** is a punctuator and the **“;”** is a string literal
6) The **“”** and **“ ”** are string literals

5) Boolean Literal

It is a literal with two kinds of values either **true** or **false**.

The output of a relational and a logical expressions is a boolean literal.

E.g: System.out.println(5>3); The output is **true.**

Write output:

```
int x=9,y=8;
System.out.println(x<y);
```

Output:
false

6) Null Literal

Null literal is the literal with no value. The keyword **null** can be assigned with String variables.

E.g.1: String name = **null**; The **String** is a built-in class. The **name** is its object.
E.g.2: String name = **“”**; The **“”** with no value in between is also a null literal.
E.g.3: char a = ‘\0’; The **‘\0’** is a null character in ASCII.
E.g.4: char a = ‘\u0000’; The **‘\u0000’** is a null character in Unicode.

Write statement for the following:

a) Initialize a null character to c;
b) Initialize a null string to st.

Output:

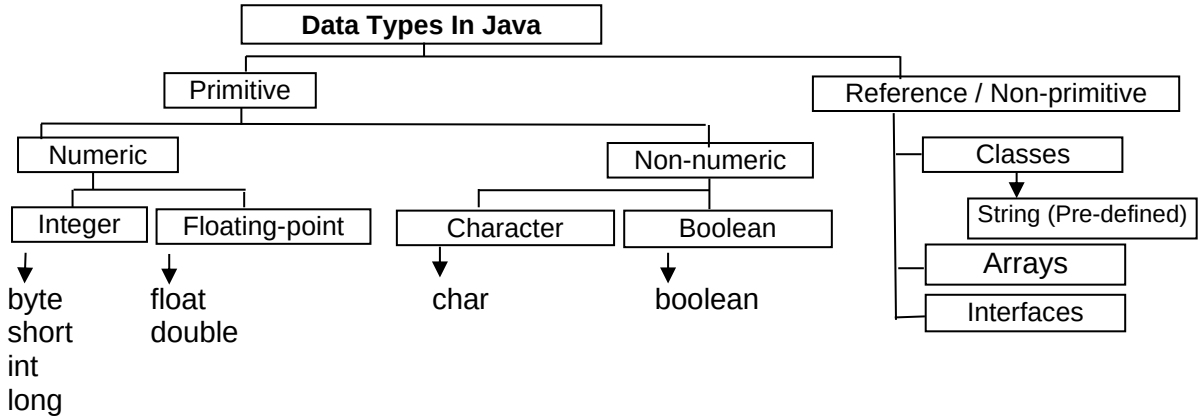
a) char c = ‘\0’; or char c = ‘\u0000’;
b) String st = “”; or String st = null;

Lesson 6

Data Types in Programs and Type Conversions

Data means values or literals or constants. We have seen that there are six kinds of literals in general concept. To use them in programs these are divided as in the given diagram.

Diagram of Various Data Types at a Glance



Defintion and Use

Data type is the way to identify the type of data in a program. E.g.: int, double, char , String etc.

Data Types are Categorized Broadly into Two

1) **Primitive** (basic, fundamental or intrinsic): These are the basic data types defined by Java developers. It is a single cell memory.

2) **Reference data types**: These are **classes**, **arrays** and **interfaces**. These are **composite data types** of primitives and String etc. It can contain more than one memory cells.

The **String** is a pre-defined class; so it is reference data type. It is composed of multiple **char** memory cells. String **s=**"Anu"; The **s** is composed of three **char** memory cells.

Primitive Data Types

Numeric data types

The numeric data types are: 1) byte 2) short 3) int 4) long 5) float and 6) double. Sometimes the **char** also is considered as **numeric** data type, because each character has a numeric value. The value of character **A** is **65**, **B** is **66** etc. These values start from **0** to **65535**. The **char c=65;** is valid, Value of **c** is **A**.

Non-numeric data type

The non-numeric data type is **boolean** and **char**

Integer Data Types

These are for integers: 1) byte 2) short 3) int 4) long

Floating Point (Real) Data Types

Data types for **numbers with decimal point** are **floating point** data types. These are: 1) float 2) double

Character Data Type

Data type for **single character** is **character** datatype. It is **char**. (The **String** is used for multiple characters).

Reference Data Type (Frequently Used)

Pre-defined classes String, System, Scanner etc are reference data types. Users also can define reference data types.

Correct the errors:

1) int a = 2.5;
2) char b = "Java";
3) double c = "2.5";
4) String d = 'Computer'

Answer:

1) double a = 2.5;
2) String b = "Java";
3) String c = "2.5"; or double c=2.5;
4) String d = "Computer";

Size and Range

Capacity (memory space**) of a variable** to store a value is known as **size**. It is measured in bits or bytes.
E.g.: byte a; The memory cell named **a** is formed with **8 bits** i.e., **1 byte**.
The **quantity of value that can be stored** in a variable ()memory cell is known as **range**.
E.g.: byte **a**; The memory cell named **a** can store an integer ranging from **-128** to **127**.

Diagrammatic Representation of Memory *(Only for knowledge)*

ASCII of **5** is **53**. Its binary is **110101**. It is stored as binary digits in memory cells.
byte **a**=5; short **b**=5; int **c**=5; long **d**=5;

0	0	1	1	0	1	0	1
0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	1

a 1 byte (8 bits) is used

b 2 bytes (16 bits) are used

c 4 bytes (32 bits) are used

d 8 bytes (64 bits) are used

Size and Range of Various Primitives at a Glance

Type	Size	Range	
byte	1 bytes (8 bits)	-128 to +127 (**-2^7 to 2^7-1**)	
short	2 bytes (16 bits)	-32768 to +32767 (**-2^{15} to $2^{15}-1$**)	
int	4 bytes (32 bits)	around -2 billion to +2 billion (**-2^{31} to $2^{31}-1$**)	
long	8 bytes (64 bits)	around below -2 billion and 2 billion or above **-2^{63}** to **$2^{63}-1$**	
float	4 bytes (32 bits)	-3.4×10^{38} to $3.4\times10^{38}-1$	Precision up to 8 digits (3.3333333)
double	8 bytes (64 bits)	-1.7×10^{308} to $1.7\times10^{308}-1$	Precision up to 17 digits (3.3333333333333335)
boolean	1 byte (8 bits)	true and false	
char	2 bytes (16 bits)	0 to 65535	

Answer the following:

1) Write the minimum and maximum value that can be stored in a byte variable
2) Initialize the number 9446748197 in a variable named mob
3) Initialize the lowest number and highest number in two char data type variables
4) Arrange in ascending order in size: float, double, byte, long, short, int, char, boolean
5) Write the smallest and largest value that can be initialized to a short data type variable

Answer:

1) mimimum: -128 maximum: 127
2) long mob = 9446748197**L**;
3) char a=0, b=65535;
4) boolean, byte, char, short, int, float, long, double
5) smallest: -32768 largest: 32767

Lesson 7

Type Conversion

1. Define type conversion.

   Conversion of one type of data into another type is called type conversion.
   E.g.: int a = 'A'; The **'A'** is converted to **65** when it is stored into the variable **a**.

Type Conversion At A Glance

Type Conversion

Implicit Conversion (Automatic by the compiler) **(Type Promotion / Coercion)**	**Explicit Conversion** (By the user) **(Type Casting)**
E.g.1: System.out.println('A'+'B'); The output is **131** (promoted to **int**)	E.g.1: System.out.println(**(double)**65); **65** is converted to **double**. Result is **65.0**
E.g.2: double a=5; The value of a is **5.0** (promoted to **double**)	E.g.2: System.out.println(**(char)**65); **65** is converted to **char**. So result is **A**

2. What are the **two types of conversions**? Explain briefly.
   The two types of conversions are **implicit** and **explicit type conversions**.
   Implicit type conversion: An implicit type conversion is an automatic conversion done by the compiler without programmer's intervention. It is known as **type promotion** or **coercion**.
   Explicit type conversion: An explicit type conversion is done by a user into a specific type. It is known as **type casting**.

Type Promotion

3. Define **type promotion** / **coercion**. Write an example also.

   Type promotion is the **implicit type conversion** performed by the **compiler** when **different types of data** occur in an expression. In order **not to lose data**, the compiler **promotes** the data type of the expression to the **appropriate type** in the expression. E.g.: doube a=**5**; Its output is **5.0**.

Rules of Type Promotion

1) In an expression with **various integers** then the result will be largest **integer** data type.
   E.g.: byte a=3; short b=4; int c=5; long d=6;
   ------ s=a+b+c+d; The **s** should be **long**.
   long s=a+b+c+d;
2) In an expression with **various real types** then the result will be largest **real** type.
   E.g.: float a=2; double b=3.0;
   ---------- s=a+b; The **s** should be **double**.
   double s=a+b;
3) In an expression with **integer** and **real** then the result will be **real** type.
   E.g1: long b=3; float a=2F;
   ------ s=a+b; The **s** should be **float**.
   float s=a+b; Output is **5.0**
   E.g2: long b=3; double a=2;
   ---------- s=a+b; The **s** should be **double**.
   double s=a+b; Output is **5.0**
4) In an expression with **character** and **integer** then the result will be **integer**.
   E.g.: char a= 'A'; int b=1;
   --- c=a+b; The **c** should be **int**.
   int c=a+b; Output is **66**
5) In an expression with **character** and **character** then the result will be **integer**.
   E.g.: char a= 'A'; char b= 'B';
   ---- c=a+b; The **c** should be **int**.
   int c=a+b; Output is **131**
6) In an expression with **String** and **int/char/double/boolean/String** etc then the result will be **String**.
   E.g.: String a= "Value" + int b = ':' + int c = 5; + double d = 2.5; + boolean e =true;
   --------- s = a+b+c+d+e; The **a** should be **String**.
   String s= a+b+c+d+e; Output is **Value:52.5true**

4. Write output

```
long l=9; float f=4;
System.out.println(l/f);
System.out.println(l%f);
```

Output:
2.25
1.0

5. Write output:

```
byte a=1; char b= 'A';
System.out.println(a+b);
System.out.println((char)(b+a));
```

Output:
66
B

Data Type Priority: Higher to Lower

6. Write priority of operators higher to lower regarding to type conversion

1.	double	8 bytes	
2.	float	4	Real is geater than integer, float can hold long values.
3.	long	8	
4.	int	4	
5.	char	2	char can hold integers upto 65535.
6.	short	2	short can hold integers upto 32767 only.
7.	byte	1	
8.	boolean	1	

7. What is the **need of type promotion**?

When there are different types of data in an expression, in order to not lose data, compiler converts it to the appropriate type which is in the expression. 5/2 is 2 and 5/2.0 is 2.5. The 0.5 is saved here.

Type Casting

6. Define **type casting**. Write an example.

The explicit type conversion done forcefully by the user is known as type casting.

E.g.: **(char)**65. The integer 65 will be converted to character; so output is A.

7. Write output and state which kind of type conversion

```
int a='B';
System.out.println(a);
```

Output:
66
Type promotion

8. Write output and state which kind of type conversion

```
System.out.println((char)66);
System.out.println((double)66);
```

Output:
B
66.0
Type casting

Type Casting Does Not Change Original Value

The **(double)n** does not change the data type of **n**. It is **int** itself.

E.g.: **int n=2;**

```
System.out.println((double)n);
System.out.println(n);
```

The output is:
2.0
2

9. Write output and state which kind of type conversion

```
int a=7;
System.out.println((float)a);
System.out.println(a);
```

Output:
7.0
7
Type casting

10. Write output:

```
int a=7,b=2;
System.out.println(a/(double)b);
System.out.println((float)a%b);
```

Output:
3.5
1.0

3. Write output:
   1.
```
int a=7,b=2;
System.out.println(a/b);
```
Answer:
3
   2.
```
int a=9, b=2;
double c=a/b;
System.out.println(c);
```
Answer:
4.0
   3.
```
int a=7; float b=2;
System.out.println(a/b);
```
Answer:
3.5
   4.
```
long a=7; float b=2;
System.out.println(a/b);
```
Answer:
3.5
   5.
```
char a='A'; int b=5;
System.out.println(a-b);
```
Answer:
60
   6.
```
char a='a', b='A';
System.out.println(a-b);
```
Answer:
32
   7.
```
char a='A', b='a';
System.out.println(a-b);
```
Answer:
-32
   8.
```
char a=5, b=3;
System.out.println(a<b);
```
Answer:
false
   9.
```
int a=5,b=3,c=6;
System.out.println(a<b||a<c);
```
Answer:
true
   10.
```
String a= "Result"; char b= ':';
int c=40; double d=6.75;
System.out.println(a+b+c+d);
```
Answer:
Result:406.75