Data types:

Every variable has a type, every expression has a type

and all types are strictly define more over every assignment

should be checked by the compiler by the type compatibility

hence java language is considered as strongly typed programming language.

Java is pure object oriented programming or not?

Java is not considered as pure object oriented programming language

because several oops features

(like multiple inheritance, operator overloading) are not supported by

java moreover we are depending on primitive data types

which are non objects.

Diagram: primitive-data-type.png

Except Boolean and char all remaining data types are

considered as signed data types

because we can represent both "+ve" and"-ve" numbers.

Integral data types :

Byte:

Size: 1byte (8bits)

Maxvalue: +127

Minvalue:-128

Range:-128to 127[-27 to 27-1]

Example:

byte b=10;

byte b2=130;//C.E:possible loss of precision

found : int

required : byte

byte b=10.5;//C.E:possible loss of precision

byte b=true;//C.E:incompatible types

byte b="ashok";//C.E:incompatible types

found : java.lang.String

required : byte

byte data type is best suitable if we are handling

data in terms of streams either from

the file or from the network

Short:

The most rarely used data type in java is short.

Size: 2 bytes

Range: -32768 to 32767(-2

15 to 215-1)

Example:

short s=130;

short s=32768;//C.E:possible loss of precision

short s=true;//C.E:incompatible types

Short data type is best suitable for 16 bit processors

like 8086 but these processors are completely outdated and hence

the corresponding short data type is also out data type.

Int:

This is most commonly used data type in java.

Size: 4 bytes

Range:-2147483648 to 2147483647 (-2

31 to 231-1)

Example:

int i=130;

int i=10.5;//C.E:possible loss of precision

int i=true;//C.E:incompatible types

long:

Whenever int is not enough to hold big values then we should go for long data type.

Example:

To hold the no. Of characters present in a big file int may not enough hence the return

type of length() method is long.

long l=f.length();//f is a file

Size: 8 bytes

Range:-2

63 to 263-1

Note: All the above data types (byte, short, int and long) can be used to represent whole

numbers. If we want to represent real numbers then we should go for floating point

data types.

Floating Point Data types:

Float double

If we want to 5 to 6 decimal places of

accuracy then we should go for float.

If we want to 14 to 15 decimal places of

accuracy then we should go for double.

Size:4 bytes. Size:8 bytes.

Range:-3.4e38 to 3.4e38. -1.7e308 to1.7e308.

float follows single precision. double follows double precision.

boolean data type:

Size: Not applicable (virtual machine dependent)

Range: Not applicable but allowed values are true or false.

Example 1:

boolean b=true;

boolean b=True;//C.E:cannot find symbol

boolean b="True";//C.E:incompatible types

boolean b=0;//C.E:incompatible types

Char data type:

In old languages like C & C++ are ASCII code based the no.Of

ASCII code characters are < 256 to represent these 256 characters 8 - bits

enough hence char size in old languages 1 byte.

In java we are allowed to use any worldwide alphabets character and java is Unicode based and no.Of unicode characters are > 256 and <= 65536 to represent all these characters one byte is not enough compulsory we should go for 2 bytes.

Size: 2 bytes

Range: 0 to 65535

Example:

```
char ch1=97;
char ch2=65536;//C.E:possible loss of precision
```

Summary of java primitive data type:

Summary of java primitive data type.png

The default value for the object references is "null".

Literals:

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

Example:Litrals.png

Integral Literals:

For the integral data types (byte, short, int and long)

we can specify literal value in the following ways.

1) Decimal literals: Allowed digits are 0 to 9.

Example: int x=10;

2) Octal literals: Allowed digits are 0 to 7. Literal value should

be prefixed with zero.

Example: int x=010;

3) Hexa Decimal literals:

 The allowed digits are 0 to 9, A to Z.

 For the extra digits we can use both upper case and lower case characters.

 This is one of very few areas where java is not case sensitive.

 Literal value should be prefixed with ox(or)oX.

Example: int x=0x10;

These are the only possible ways to specify integral literal.

Which of the following are valid declarations?

1. int x=055;

2. int x=0867;

3. int x=0xFACE;

4. int x=0xdog;

5. int x=0xBeer;

6. int x=0xabb2cd;

Example:

```
int x=10;
int y=010;
int z=0x10;
System.out.println(x+"----"+y+"----"+z); //10----8----16
```

By default every integral literal is int type but we can specify explicitly as long type by suffixing with small "l" (or) capital "L".

Example:

int x=10;(valid)

long l=10L;(valid)

long l=10;(valid)

int x=10l;//C.E:possible loss of precision(invalid)

found : long

required : int

There is no direct way to specify byte and short literals explicitly.

But whenever we are assigning integral literal to the byte variables

and its value within the range of byte compiler automatically

treats as byte literal. Similarly short literal also.

Example:

byte b=127;(valid)

byte b=130;//C.E:possible loss of precision(invalid)

short s=32767;(valid)

short s=32768;//C.E:possible loss of precision(invalid)

Floating Point Literals:

Floating point literal is by default double type but we can specify

explicitly as float type by suffixing with f or F.

Example:

float f=123.456;//C.E:possible loss of precision(invalid)

float f=123.456f;(valid)

double d=123.456;(valid)

We can specify explicitly floating point literal as

double type by suffixing with d or D.

Example:

double d=123.456D;

We can specify floating point literal only in decimal form and we can't specify in octal and hexadecimal forms.

Example:

double d=123.456;(valid)

double d=0123.456;(valid) //it is treated as decimal value but not octal

double d=0x123.456;//C.E:malformed floating point literal(invalid)

Which of the following floating point declarations are valid?

1. float f=123.456;
2. float f=123.456D;
3. double d=0x123.456;
4. double d=0xFace;
5. double d=0xBeef;

We can assign integral literal directly to the floating point data types and that integral literal can be specified in decimal , octal and Hexa decimal form also.

Example:

double d=0xBeef;

System.out.println(d);//48879.0

But we can't assign floating point literal directly to the integral types.

Example:

int x=10.0;//C.E:possible loss of precision

We can specify floating point literal even in exponential

form also(significant notation).

Example:

double d=10e2;//==>10*102(valid)

System.out.println(d);//1000.0

float f=10e2;//C.E:possible loss of precision(invalid)

float f=10e2F;(valid)

Boolean literals:

The only allowed values for the boolean type are true (or)

false where case is important.

i.e., lower case

Example:

1. boolean b=true;(valid)
2. boolean b=0;//C.E:incompatible types(invalid)
3. boolean b=True;//C.E:cannot find symbol(invalid)
4. boolean b="true";//C.E:incompatible types(invalid)

Char literals:

1) A char literal can be represented as single character within

single quotes.

Example:

1. char ch='a';(valid)
2. char ch=a;//C.E:cannot find symbol(invalid)
3. char ch="a";//C.E:incompatible types(invalid)
4. char ch='ab';//C.E:unclosed character literal(invalid)

2) We can specify a char literal as integral literal which represents

Unicode of that character.

We can specify that integral literal either in decimal or octal or

hexadecimal form but allowed values range is 0 to 65535.

Example:

1. char ch=97; (valid)

2. char ch=0xFace; (valid)

System.out.println(ch); //?

3. char ch=65536; //C.E: possible loss of precision(invalid)

3) We can represent a char literal by Unicode representation which is nothing but

'\uxxxx' (4 digit hexa-decimal number) .

Example:

1. char ch='\ubeef';

2. char ch1='\u0061';

System.out.println(ch1); //a

3. char ch2=\u0062; //C.E:cannot find symbol

4. char ch3='\iface'; //C.E:illegal escape character

5. Every escape character in java acts as a char literal.

Example:

1) char ch='\n'; //(valid)

2) char ch='\l'; //C.E:illegal escape character(invalid)

Escape Character and Description

\n New line

\t Horizontal tab

\r Carriage return

\f Form feed

\b Back space character

\' Single quote

\" Double quote

\\ Back space

Which of the following char declarations are valid?

1. char ch=a;
2. char ch='ab';
3. char ch=65536;
4. char ch=\uface;
5. char ch='/n';
6. none of the above.

String literals:

Any sequence of characters with in double quotes is treated

as String literal.

Example:

String s="Ashok"; (valid)

1.7 Version enhansements with respect to Literals :

The following 2 are enhansements

1. Binary Literals
2. Usage of '_' in Numeric Literals

Binary Literals :

For the integral data types untill 1.6v we can specified literal value in the following ways

1. Decimal

2. Octal

3. Hexa decimal

But from 1.7v onwards we can specified literal value in binary form also.

The allowed digits are 0 to 1.

Literal value should be prefixed with Ob or OB .

```
int x = 0b111;
System.out.println(x); // 7
```

Usage of _ symbol in numeric literals :

From 1.7v onwards we can use underscore(_) symbol in numeric literals.

```
double d = 123456.789; //valid
double d = 1_23_456.7_8_9; //valid
double d = 123_456.7_8_9; //valid
```

The main advantage of this approach is readability of the code will be improved At the time of compilation ' _ ' symbols will be removed automatically , hence after compilation the above lines will become double d = 123456.789

We can use more than one underscore symbol also between the digits.

Ex : double d = 1_23_ _456.789;

We can use more than one underscore symbol also between the digits.

Ex : double d = 1_23_ _456.789;

We should use underscore symbol only between the digits

```
double d=_1_23_456.7_8_9; //invalid
double d=1_23_456.7_8_9_; //invalid
double d=1_23_456_.7_8_9; //invalid
double d='a';
```

System.out.println(d); //97

integral data types

float f=10L;

System.out.println(f); //10.0

floating-point data types

Diagram: type-promotion.png