

- 1) **Elucidate the following concepts: 'Statically Typed Language', 'Dynamically Typed Language', 'Strongly Typed Language', and 'Loosely Typed Language'? Also, into which of these categories would Java fall?"**

Statically Typed language:

Type of language in which the variable type are checked at the compile time (type of the variable is known at the compile time). This is a compile or intermediate type language and is used at enterprise applications.

Dynamically Typed Language:

The type of the variables is checked during the run-time

Java is a both statically and dynamically typed language.

Strongly Typed Language

the type of the data (Ex: integer, character, decimal, ...etc) is specified and highly considered.

Loosely Typed Language

the loosely typed language is a programming language that does not require a variable to be defined.

Java is a strongly typed language

- 2) **"Could you clarify the meanings of 'Case Sensitive', 'Case Insensitive', and 'Case Sensitive-Insensitive' as they relate to programming languages with some examples? Furthermore, how would you classify Java in relation to these terms?"**

Case Sensitive

The ability to distinguish uppercase or lowercase letters in a computer system or program. This types of languages identify the lowercase and upper case letters as two different letters. C#, Java, XML, C++, Python...etc are case sensitive languages.

Case Insensitive

the ability to ignore the difference between upper case and lower case versions of a letter. Ada, SQL, Pascal are example for case insensitive languages.

Case Sensitive- Insensitive

Java is case sensitive.

- 3) **Explain the concept of Identity Conversion in Java? Please provide two examples to substantiate your explanation.**

Java programming language permits for any type to convert from a type to that same type. In other word assigning two instance of same type is identity conversion.

Ex:

```
int int1 = 2;
```

```

int int2 = 3;
int1 = int2;

String str1;
String str2 = "NAME";
str1 = str2;

```

4) Explain the concept of Primitive Widening Conversion in Java with examples and diagrams.

Primitive widening conversion

Primitive widening conversion is done automatically. following specific data type conversions are included in widening primitive type conversions

- byte to short, int, long, float, double
- short to int, long, float, double
- char to int, long, float, double
- int to long , float, double
- long to float or double

Following conversions are done automatically as storing a smaller size type in a large size type.

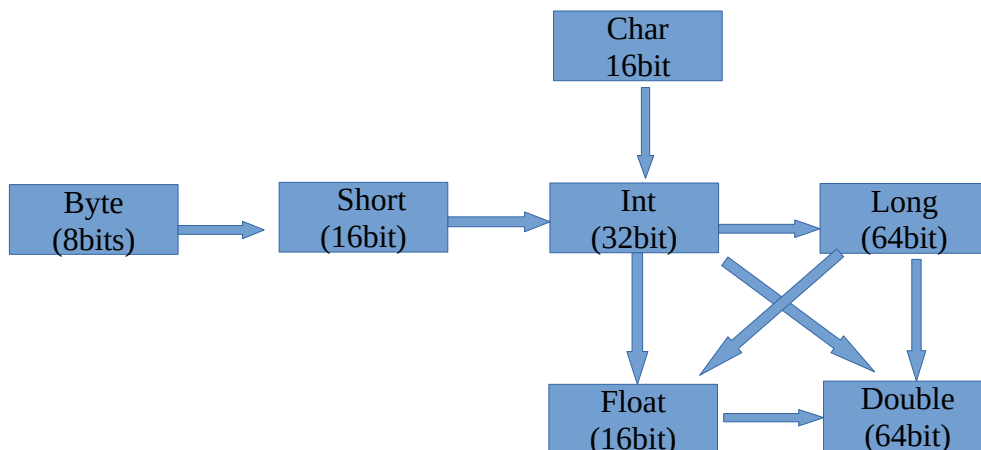
- byte to short, int, long, float, double
- short to int, long, float, double
- char to int, long, float, double
- int to long , double

In this case, information about the overall magnitude of a numeric value are not lost, where the numeric value preserved exactly as storing a smaller size type in a large size type

- byte, short, char to floating
- int to double
- float to double

But following widening primitive conversions may result lost of some of the least significant bits of the value. Those conversions have low accuracy and slow conversion.

- Int to float
- long to float (This conversion occurs due to the IEEE 754 standard structuring)
- long to double



- 5) **Explain the the difference between run-time constant and Compile-time constant in java with examples.**

Run-time constant

Run time constant can only be computed while the application is running. The value is unknown at the beginning. But JVM knows the value.

Compile-time constant

If a primitive type or a string is defined as a constant and the value is known at compile time, the compiler replaces the constant name everywhere in the code with its value. This is called a compile-time constant. A compile-time constant will have the same value each time an application runs. In the narrow primitive conversion, the values need to be compile time constant and the number should be in the relevant bits range.

- 6) **Explain the difference between Implicit (Automatic) Narrowing Primitive Conversions and Explicit Narrowing Conversions (Casting) and what conditions must be met for an implicit narrowing primitive conversion to occur?**

Implicit (Automatic) Narrowing Primitive Conversions

this is the conversion of data type having larger range to a smaller range data type. This will result in loss of information. This occurs automatically without the need for explicit instruction from the programmer. There are two conditions to be satisfied to this conversion.

1- the value should be a compile time constant

2- the data type with larger range should be in the range of target data type(data type having smaller range)

Ex : final int myInt1 = 23;

short myShort1 = myInt1;

Explicit Narrowing Conversions (Casting)

In the explicit narrowing conversion also a data type having larger range is converted in to a data type having smaller range. This type of conversion requires the programmer to specify the desired data type explicitly during the conversion to indicate that they are aware of the potential loss of data. This will lead to incorrect result as the type range to be converted is exceeds the target data type.

Ex: long Value1 = 1242002L;

short Value2 = (short) Value1;

- 7) **How can a long data type, which is 64 bits in Java, be assigned into a float data type that's only 32 bits? Could you explain this seeming discrepancy?"**

The data representation of floating point numbers in the memory follows the standard of IEEE 754. when converting long data type in to float data type, the conversion is independent from the the bit size range. But the conversion type is widening primitive conversion. Long to float conversion occurs due to the IEEE 754 structure of floats. But this may result in loss of precision (that is, the result may lose some of the least significant bits of the value.) In this case, the resulting floating-point value will be a correctly rounded version of the integer value, using IEEE 754 round-to-nearest mode.

8) Why are int and double set as the default data types for integer literals and floating point literals respectively in Java? Could you elucidate the rationale behind this design decision?

By default, the int data type is a 32-bit signed two's complement integer, which has a minimum value of -2^{31} and a maximum value of $2^{31}-1$. Byte, Short, Char data types are also included in this range. But the long data type exceeds the maximum int size. Then the programmer should adopt a special syntax "L" when assigning a value to a long that exceeded the maximum value.

Ex: long x = 2147483647; // Compiles
 long y = 2147483648; // Does not compile
 long z = 2147483648L; // Compiles

Then the "Long" wrapper class is being used, programmer must always use the "L" literal notation.

Ex: Long y = 214; // Does not compile
 Long z = 214L; // Compiles

Therefore int data type is used as the default data type in java.

Double and floats are two primitive data types of floating point numbers. In java, 32bits are allocated for a float and 64bits size is allocated for a double. Then the double provides higher range of values than the floats. And also, in the case of floating point literals suffix "f" have to be used with the value. Therefore double is set as the default datatype in java.

9) Why does implicit narrowing primitive conversion only take place among byte , char , int , and short ?

Narrowing conversion among byte ,char, int, and short can be carried out without losing the information but the conditions need to be satisfied.(value needs to be compile constant and must be in the target data type range.) long is a data type having large range ($2^{(-63)} - [2^{(-63)}-1]$). then converting such type of data in to smaller range data type will lead to loss of information or precision errors. Therefore these conversions are not implicit and required explicit casting.

10) Explain "Widening and Narrowing Primitive Conversion". Why isn't the conversion from short to char classified as Widening and Narrowing Primitive Conversion?

Both conversions take place in this conversion. First the lower data type is converted into higher data type by using widening primitive conversion. Then that data type is converted into another lower datatype using narrowing conversion.

Byte to char conversion is a widening and narrowing primitive conversion. First, the byte is converted to an int via widening primitive conversion, and then the resulting int is converted to a char by narrowing primitive conversion.

Technically short to char conversion is narrowing primitive conversion. The unsigned range of shorts from 0 to 32768 is within the size range of char data type(0-65535). Therefore the conversion can be carried out explicitly without losing information.