1. Statically typed language - Statically typed language is a programming language which every variable and expression has a type and those are explicitly declared and checked at compile time of the program rather that at runtime. In statically typed languages data type of variables must be specified when its declared.

Dynamically typed language - Dynamically typed language is a programming language which types of variables are determined at the runtime of the program. This means unlike in statically typed languages, we don’t need to specify the type of variable when declaring it.

Strongly typed language - Strongly typed language is a programming language which we need to explicitly declare variable types and type errors are caught by compiler or interpreter before program is executed.

Loosely typed language - Loosely typed language is programming language which we do not need to explicit type declaration and it does not enforce strict type checking.

**Java considered strongly typed, statically typed as well as dynamically typed programming language.**

1. Case Sensitive - Case sensitivity related to programming language means its distinguished between uppercase and lowercase letters in identifiers.

for example, if we define a variable called **myVariable**, in a case sensitive language **myvariable** considered different variable.

Case Insensitive - Case insensitive means the programming language is not differentiate between uppercase and lowercase in identifiers.

for example, if we define a variable as **myVariable** and referring it to **myvariable** treat as the same variable.

Case sensitive-intensitive - Case sensitive-intensitive language means its provide option to choose between whether they are case sensitive or case insensitive.

Java programming language is considered as case sensitive language meaning it differentiate between uppercase and lowercase letters in identifier.

1. Identity conversion in Java is a way of type conversion where a value of a data type is assigned to a variable of same data type. Which means assigned variable of specific data type, assigning value of the same data type to that variable does not require any data type conversion.

ex 1:-

*public class Main {*

*public static void main(String[] args) {*

*int number= 30;*

*int newNumber = number; // Identity conversion*

*}*

*}*

ex 2:-

*public class Main {*

*public static void main(String[] args) {*

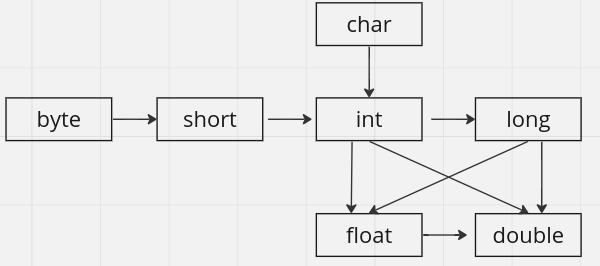
*String message = “Hello Java”;*

*int newMessage = “Hello Java”; // Identity conversion*

*}*

*}*

1. Primitive widening conversion - Primitive widening conversion is an automatic type conversion that happens when smaller data type is assigned to a variable of a larger data type.



for example -

*public class Main {*

*public static void main(String[] args) {*

*int myInt= 100;*

*long myLong = myInt;*

*float myFloat = 3.14F;*

*double myDouble = myFloat ;*

*}*

*}*

1. A run-time constant is a value that is determined when the execution of program. It’s value remain a constant during runtime of the program.

In the other hand compile time constant is a value that is know to the compiler during the compilation phase. It’s value remain a constant during the program’s execution.

example:-

*public class Main {*

*public static void main(String[] args) {*

*final int MY\_CONST = 100; // compile time constant*

*final long myLong = 10 \* (int) Math.random(); // run-time constant*

*}*

*}*

1. When converting larger data types into a smaller data type such conversions are called Narrowing Primitive Conversion in Java. Java basically allows 2 types of narrowing primitive conversions, which are Implicit (Automatic) Narrowing Primitive Conversions and Explicit Narrowing Conversions (Casting).

Implicit Narrowing Primitive Conversion:

Automatically narrowing primitive conversion occur automatically by Java compiler when certain conditions are met. Implicit narrowing conversion occurs in assigning context only, also value being assigned in an expression should be within the range of target data type, and assigning value should be a compile time constant.

ex:-

public class Demo15 {

public static void main(String[] args) {

final int x = 23; // compile time constants

byte myByte = x;

System.out.println(myByte);

}

}

Explicit Narrowing Primitive Conversion:

In Casting or explicit narrowing primitive conversion, conversions are need to be explicitly specified by the programmer. When convert a value from one data type to another need to be done and which might potentially result in data loss, they must use casting to indicate their intention explicitly.