Programming-Fundamentals-Assignment-2

1. Statically typed – Checking the type of identifiers at the compiled time

Dynamically typed – Checking the type of the identifiers at the run time

Java is a compiled language. Most of the type checks are performed at compile time, making it a **statically-typed** language. However, in some cases, type checks are also done at runtime, which means Java exhibits characteristics of both **statically and dynamically typed** languages.

Strongly typed – These languages strongly consider the type of identifiers.

Loosely typed – These languages only loosely consider the type of identifiers.

From them Java is considered as a **strongly typed** programming language.

2. **Case Sensitive** – If a programming language consider two strings with same characters but different case as two different strings, they are case sensitive.

Eg: #Java

string1 = “Hello World”;

string2 = “hello world”;

in Java it is considered string1 is not equal to string2.

Case Insensitive – If a programming language consider two different strings with same characters as equal without considering the case, they are case insensitive programming languages.

Case Sensitive-Insensitive – Some languages behave as both ways in different scenarios. They are called case sensitive-insensitive.

Java is **case sensitive** language as two strings with same characters with different cases are considered as two different strings

Eg:

String greeting1 = "hello";

String greeting2 = "Hello";

System.out.println(greeting1.equals(greetimg2)); //Gives false as their cases are different.

3. Identity conversion – Converting the type of the identity without changing its value or representation.

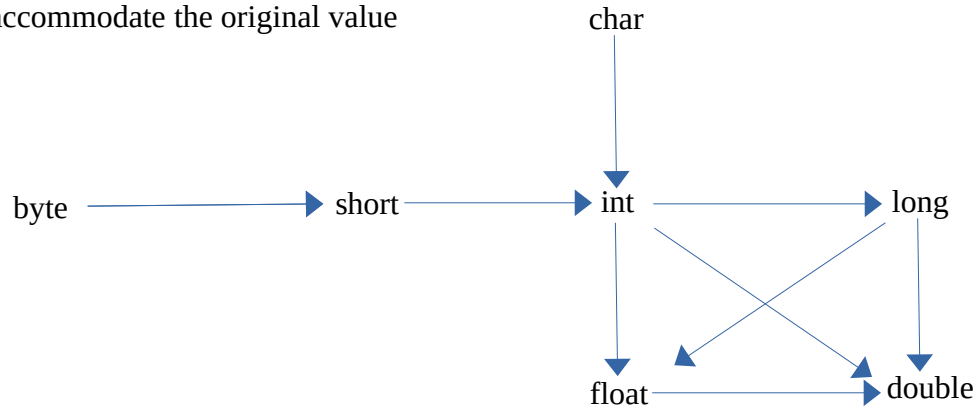
Ex:

byte myByte = 12;

int myInt = var1;

4. Primitive widening conversion -

Converting a value from one primitive data type to another data type that has a larger size or a wider range to accommodate the original value



Ex:

short myShort = 125;

int myInt = myShort;

byte myByte = 122;

short myShort = myByte;

5. **Compile time constant** – Constants whose values can be determined at the compiled level

Ex final double PI = 3.1432

Runtime constants – Constants whose values are only be determined at the run time.

Ex:

```
public static final int RANDOM_VALUE = getRandomValue();
public static int getRandomValue() {
    return (int) (Math.random() * 100);
}
```

6. **Implicit narrowing conversion** -

Converting a value from one primitive data type to another data type that has a smaller size or a less range by the compiler without any external force

Ex

int myInt = 120;

byte myByte = myInt

Explicit narrowing conversion

Converting a value from one primitive data type to another data type that has a smaller size or a less range by force

For implicit narrowing conversion, the value should be a **complied time constant** and the specific value should be **within the range of the converting data type**.

7. Here we cannot compare long and float considering no of bites since their storing format is totally different. Float data type is designed to store large numbers. That range is more than integer value range.

long data type limits = 2^{63} to 2^{63} -1

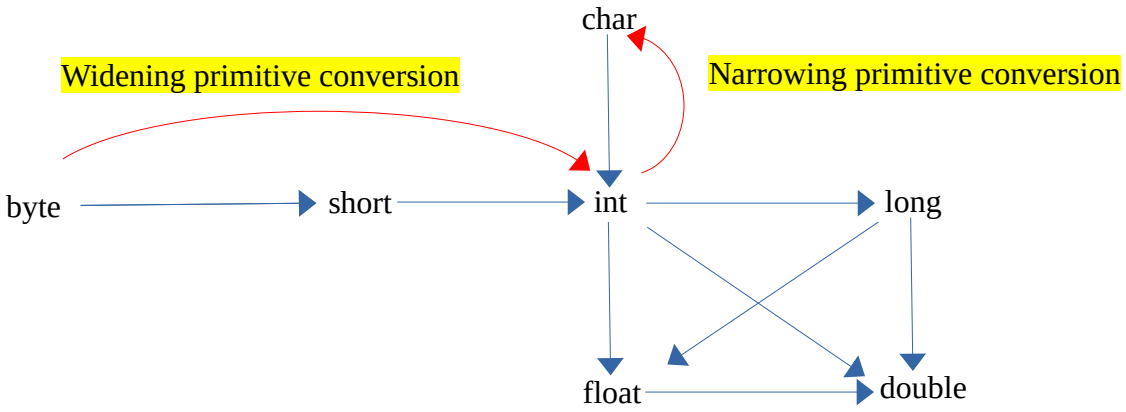
8. Java is a C based language. In C, the default data types are int and double which has been inherited into the Java.

Int and double data types are well balanced data types considering the storage and the common practice. When assigning int and double as default most of the values in practice can be used without explicitly narrowing.

9. Because these byte, short, int and char are the mostly used values in common practice and data loss can be avoided in most of the numbers used in common practice.

10. **Widening and Narrowing Primitive Conversion**

Here the conversions are formed in two steps. In the first step a **widening conversion** is happening and in the second step, a **narrowing primitive conversion** is happening.



Size of the short and char data types are similar and it is 16 bits. So after the conversion no data range has been changed in the value. As per the standard in a narrowing or widening conversion, data range should be changed but here data range is not changed. So that it is not considered as a widening narrowing converion.