Programming Fundamentals – Assignment 02

01.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 -> These type of languages performing the Type checking at the Compile Time.
Dynamically Typed Language -> These type of languages performing their Type checking at the Run Time.
Strongly Typed Language -> These type of languages consider about their type very strictly.
As an Example you can not assign a String value to a Integer type variable.
Loosely Typed Language -> This is the opposite of Strongly typed languages. In these type of languages you did not need to mentioned the type of your variables.

- Java is a both Statically and Dynamically typed Language. It is also in the category of 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 Programming Languages means they treated the lower case and upper case letters as distinct. For examples : C , C++ , Java , C# , Python

Case Insensitive Programming Languages means they treated the lower case and upper case letters as equivalent. For examples : ABAP , BASICs , SQL , Fortran

Some other programming languages the case sensitivity is varying. For a example PHP is a that kind of language. In this language variable names are case sensitive but function names are not case sensitive.

Java is a case-sensitive programming language.

//

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

Assigning two instance of same type is Identity Conversion.

It is always permitted for an expression to have the desired type to begin with, thus allowing the simply stated rule that every expression is subject to conversion, if only a trivial identity conversion.

It implies that it is permitted for a program to include redundant cast operators for the sake of clarity.

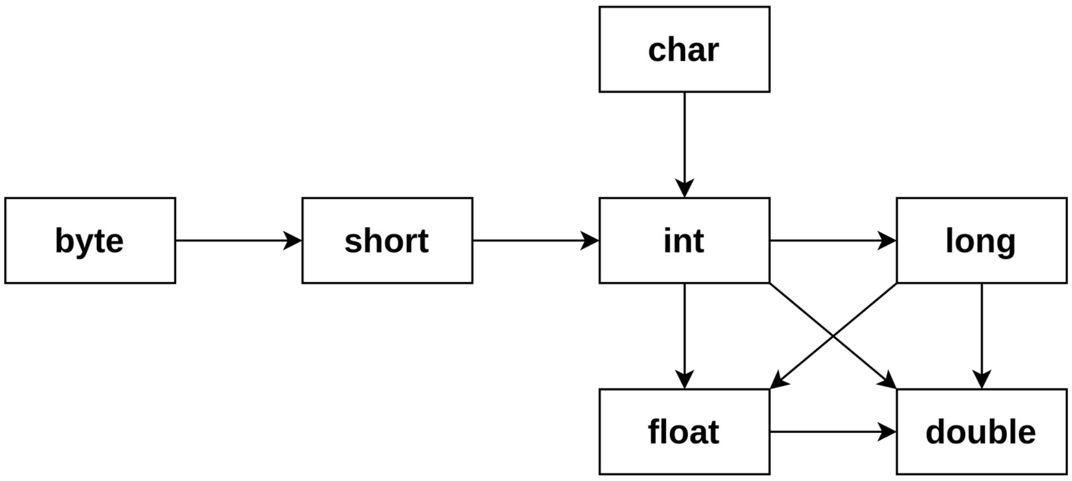
```
int i1;
int i2 = 100;
i1 = i2; // Identity Conversion
i1 = (Integer) i2; // Casting
```

//

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

19 specific conversions on primitive types are called Widening Primitive Conversion. This didn’t lose information about the overall magnitude of a numeric value. No cast required and will never result in a runtime exception.

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



//

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

Compile-time constants -Compile-time constant is computed at the time the code is compiled. A compile-time constant will have the same value each time the application runs.

Run-time constants -Run-time constant can only be computed while the application is running. A run-time constant may change each time the application runs.

```
final int MY_CONST1 = 10; // Compile - Time Constant
final int MY_CONST2 = 10 + (int) Math.random(); // Run - Time Constant
```

//

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 Conversion will occur only between byte, short, int and char data types. Explicit Narrowing Conversion called casting. We have to do that manually between other data types. Some conditions must be met to occur a implicit narrowing primitive conversion.

- This only occurs in a assignment context.
- The assigning value should be a compile-time constant.
- The assigning value should be in the range of reference data type.

//

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 range of the float is wider than the range of long data type. That happens mainly because of the data representing structure of the floats in the RAM. That is why we can assign long (64bits) into the float (32bits) without out of range exception. But you can lose the precision by a little, but the general magnitude of the value can be represent.

//

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?

Integer makes the balance between memory usage and range. Using 32 bits are generally sufficient and it provides a good compromise between memory efficiency and range of values that can be represented. Smaller data types could lead to frequent type conversions, and larger data types would consume more memory unnecessarily for many common use cases. Defaulting int, java aims to promote simplicity and efficiency while still accommodating a wide range of integer values for most applications. As same as this using double as floating point literals provides higher degree of precision compared to the float. It is suitable for handling a broad spectrum of numerical data. Also it allows java to align with common standard and practices in many other programming languages.

//

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

‘long’ is a 64-bit singed data type with a very large range comparing to byte,short,int and char. When converting from a larger data type to a smaller data type there is a risk of losing data or precision. By disallowing implicit narrowing conversion to ‘long’ java enforces explicit type casting in such cases.

//

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

Conversion of byte to char combines with both widening primitive conversion and the narrowing primitive conversion. That is called as Widening and Narrowing Primitive Conversion. In this case 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.

When we consider the conversion of short to char, Both types are 16-bit in java. But char is a unsigned type where short is a signed type. So in this conversion we only can consider the zero and positive values. In that case the maximum positive number that char can represent is very much higher than the maximum positive number that short can represent (32 767). So in the conversion here effectively widening the range of positive values of the short. So we can consider this conversion as a Implicit Widening Primitive Conversion, not as a widening and narrowing primitive conversion.

//