1. **Statically Typed Language**

In statically typed programming languages, type checking occurs at compile time. Java is a statically typed programming language.

**Dynamically Typed Language**

In dynamically typed programming languages, type checking take place at run time or execution time. Java is also a dynamically typed programming language.

**Strongly Typed Language**

Strongly typed languages, that variable have a well-defined type and that there are strict rules about the type of variable. Java is a strongly typed programming language because it demands the declaration of every variable with a data type.

**Loosely Typed Language**

Loosely typed languages don’t care about the type of variables. We do not have to specify the variable type in advance. Java is not a loosely typed language.

1. **Case Sensitive**

If a programming language is case sensitive, it means that it distinguishes between uppercase and lowercase letters. Case sensitive programming languages include C, C#, C++, Java, Python, Ruby and Swift.

**Case Insensitive**

If a programming language is case insensitive, it has ability to ignore the difference between upper- and lower-case version of a letter. Some examples of these programming languages include Ada, Fortran, SQL, and Pascal.

1. **Identity Conversion**

In Java, identity conversion is a type conversion that is the simplest and safest type of conversion. It occurs when a value is assigned to a variable of the same type without any explicit casting or conversion. In other words, when the sources value’s type matches exactly with the target variable’s type, the Java compiler performs an identity conversion.

Ex 1: Integer to int

int number = 42; // Declaration of an int variable

Integer integerObject = number; // Identity conversion from int to Integer

In this example, we have an int variable named number with value 42. The wrapper class Integer wraps the primitive int type. When we assign the int variable number to an Integer object integerObject, an identity conversion takes place because the two types of matches. The java compiler automatically converts the int primitive to the corresponding Integer object.

Ex 2: String to String

String name = "John"; // Declaration of a String variable

String newName = name; // Identity conversion from String to String

In this example, we have a String variable named name with the value John. When we assign the String variable name to another String variable newname, an identity conversion occurs because both variables have the same type (String). The java compiler recognizes this and performs the assignment without any need for explicit casting or conversion.