1.

Statically Typed Language: In this type, the data types of variables are explicitly defined at the compile time

Dynamically Typed Language: the data type of variables are determined at runtime.

Strongly Typed Language : Strongly care about the type. It means the compiler will check the compatibility of data types and prevent any operation between incompatible types

Loosely Typed Language : Here, it doesn’t care about the type and allowing for more flexibility in operation between different data types.

2.

Case Sensitive: In this type languages, uppercase and lowercase letters are considered that means if we declare a variable as “myInt” and if we try to access it by using “Myint” or something else it would not be consider as the same variable. Eg; java

Case Insensitive Languages : In this type, the language does not differentiate between upper case and lower case characters. “myInt” and “Myint” would be considered as same variable. Eg: Ada, Fortran , SQL and Pascal

Case Sensitive-Insensitive- This type language refers to combination of both approaches within the same programming language. Certain aspects are case sensitive while others are case insensitive.

Java-Case Sensitive, Identifiers like variable names,method names, class names are case sensitive in java. Myint and myInt would be considered as different variables in java.

3.
Identity Conversion is a specific type of conversion that applies only to reference types (objects) and not to primitive types. It occurs when a reference is assigned to a variable of the same type or when a method returns an object of the same type or when a method returns an object of the same type without any explicit type casting or data transformation.

```
Eg:     Color myColor = new Color(“red”);
        Color finalColor = myColor;

         Name myName = new Name(“A”);
         Name lastName=myName;
```

4. It allows to automatic and implicit conversion of a smaller data type to a larger data type without the risk of loosing information

Primitive widening Conversion

$\longrightarrow$

byte --------- > short -----------> int----------> long

It is essential to be aware of the data range and precision when performing widening converssion.

```
eg:  byte myByte=100;
     short myShort=mybyte;
      int myInt = myByte;
      long myLong =myByte;
```

5. compile time constant:-

Compile time constant are known to the compiler at compile time and their values cannot change during program execution. These constants are replaced with their values at compile time, which helps in optimizing the code. They are using the “final” keyword and can be primitive type or string literals. .

```
final int MY_INT_CONST = 65;
```

Run time constant:-

Those values are determine at run time and they may change during program execution. These constants are not replace with their values at compile time and are evaluated at run time.

```
byte myByte3=My_CONSt2;
```

6.
Implicit (Automatic) Narrowing conversion occur automatically and do not require any explicit casting syntax in the code. They are allowed when converting from a wider range data type to a narrow er range data type, potentially leading to a loss of information or precision. Explicit narrowing conversions also known as casting, requires the programmer to use the casting syntax explicitly. They are used to convert a value from a wider to a narrower data type when the compiler cannot perform the conversion automatically due to the potential loss of information or precision.

Conditions for implicit narrowing primitive conversion.

- An implicit narrowing conversion can occur when the value been converted can be represented within the appropriate bit range and when the value is a compile time constant.

7.

8.

9.

Implicit narrowing primitive conversions only take place among byte, char, int, short because types are closely related in their size and representation. These conversions are safe and do not cause loss of information. These for types have well defined range of values that can be easily converted without significant loss of data. For conversion between other numeric type like “long” or “float” developers must use explicit casting to indicate their intention and handle any potential loss of data consciously.

10.