DATA TYPES & VARIABLES In Java

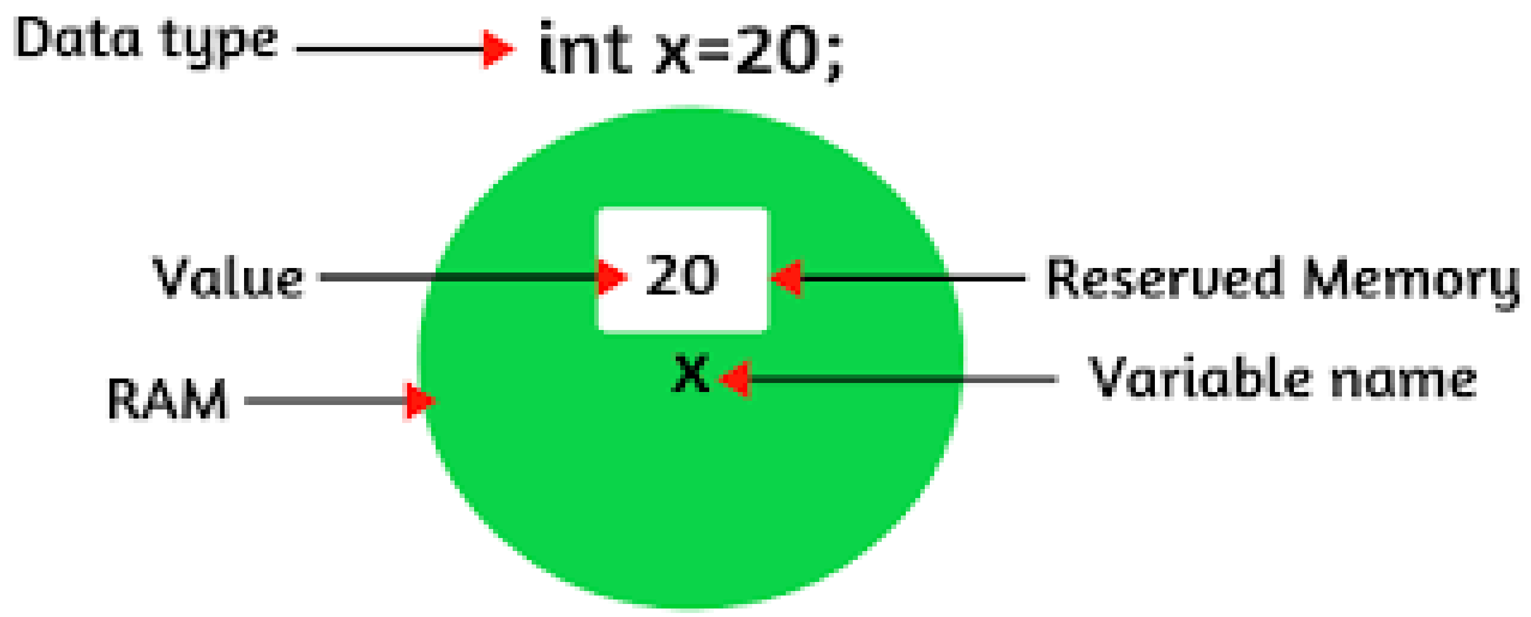
Code For Coders

Variables in Java

Java variable is a name given to a memory location. It is the basic unit of storage in a program.

- The value stored in a variable can be changed during program execution.
- Variables in Java are only a name given to a memory location. All the operations done on the variable affect that memory location.
- In Java, all variables must be declared before use.

Variables in Java

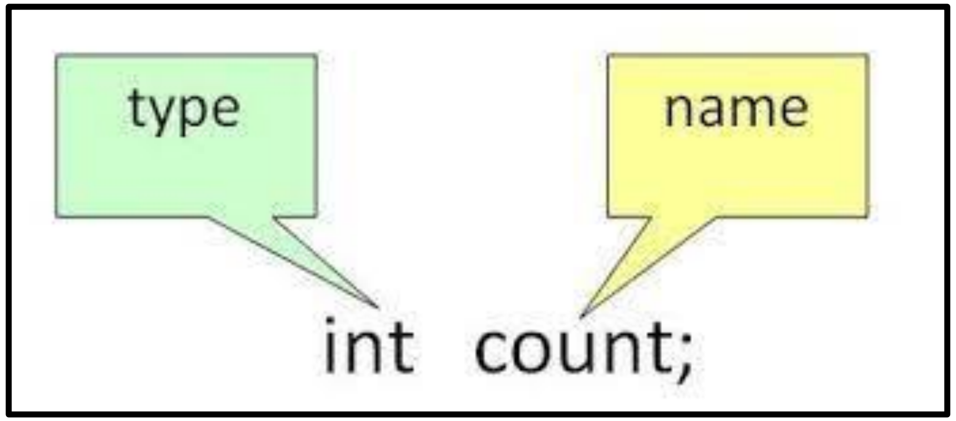


A Memory Representation of Variable

How to Declare Variables in Java?

- We can declare variables in Java as pictorially depicted below as a visual aid.
- From the image, it can be easily perceived that while declaring a variable, we need to take care of two things that are:

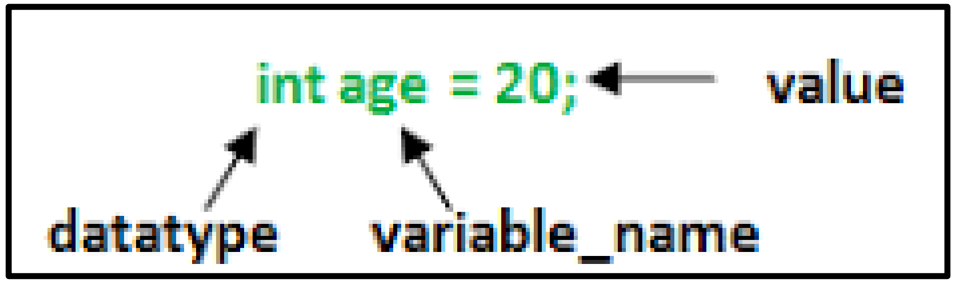
1. datatype: Type of data that can be stored in this variable.
2. data_name: Name was given to the variable.

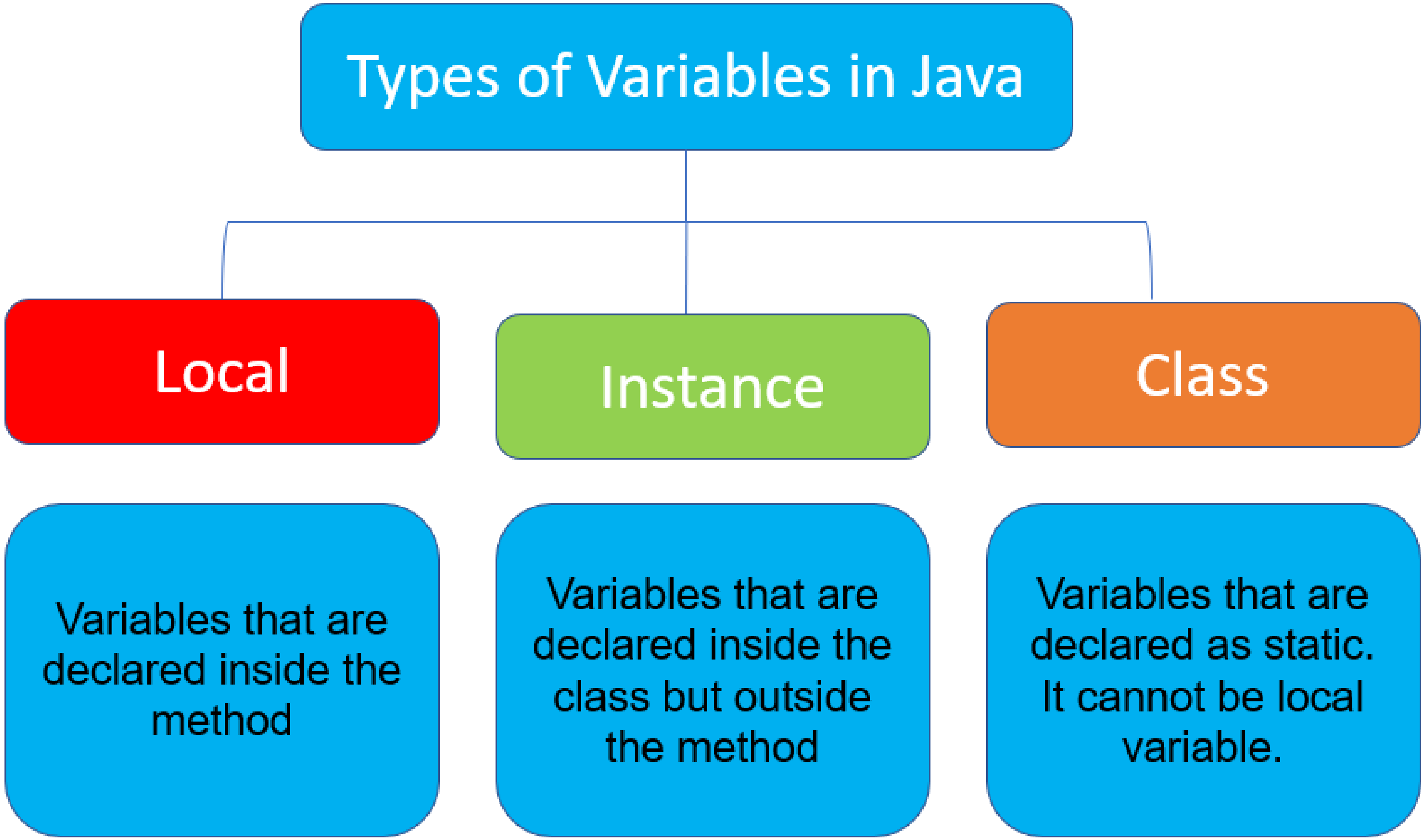


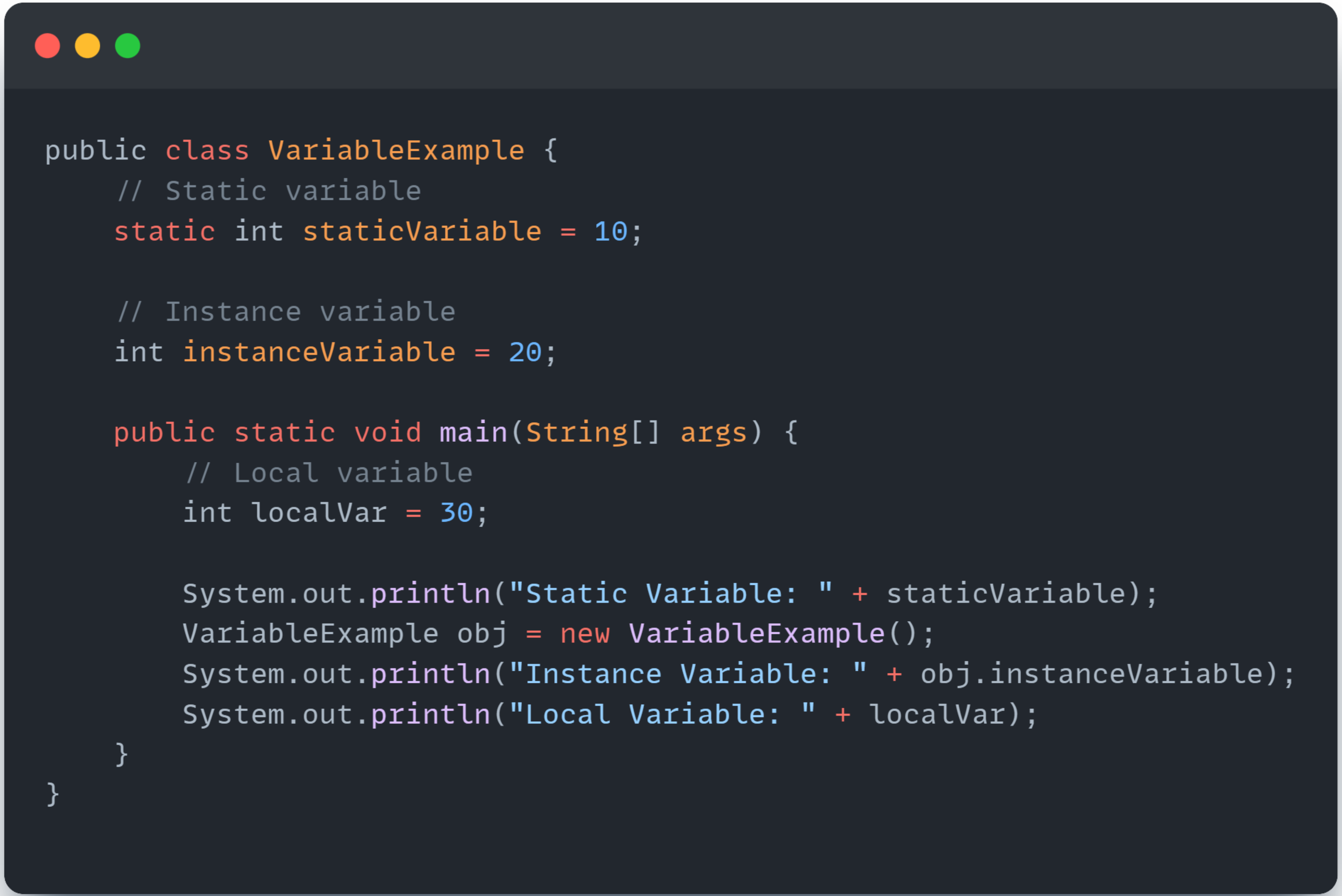
How to Initialize Variables in Java?

➢It can be perceived with the help of 3 components that are as follows:

1. **datatype**: Type of data that can be stored in this variable.
2. **variable_name**: Name given to the variable.
3. **value**: It is the initial value stored in the variable.




Types of Variables in Java
Local
Instance
Class
Variables that are declared inside the method
Variables that are declared inside the class but outside the method
Variables that are declared as static. It cannot be local variable.



```java
public class VariableExample {
    // Static variable
    static int staticVariable = 10;

    // Instance variable
    int instanceVariable = 20;

    public static void main(String[] args) {
        // Local variable
        int localVar = 30;

        System.out.println("Static Variable: " + staticVariable);
        VariableExample obj = new VariableExample();
        System.out.println("Instance Variable: " + obj.instanceVariable);
        System.out.println("Local Variable: " + localVar);
    }
}
```

Differences between the Instance variables and the Static variables:

- Each object will have its own copy of an instance variable, whereas we can only have one copy of a static variable per class, irrespective of how many objects we create. Thus, **static variables** are good for **memory management**.
- Changes made in an instance variable using one object will not be reflected in other objects as each object has its own copy of the instance variable. In the case of a static variable, changes will be reflected in other objects as static variables are common to all objects of a class.
- We can access instance variables through object references, and static variables can be accessed directly using the class name**.**
- Instance variables are created when an object is created with the use of the keyword ‘new’ and destroyed when the object is destroyed. Static variables are created when the program starts and destroyed when the program stops.

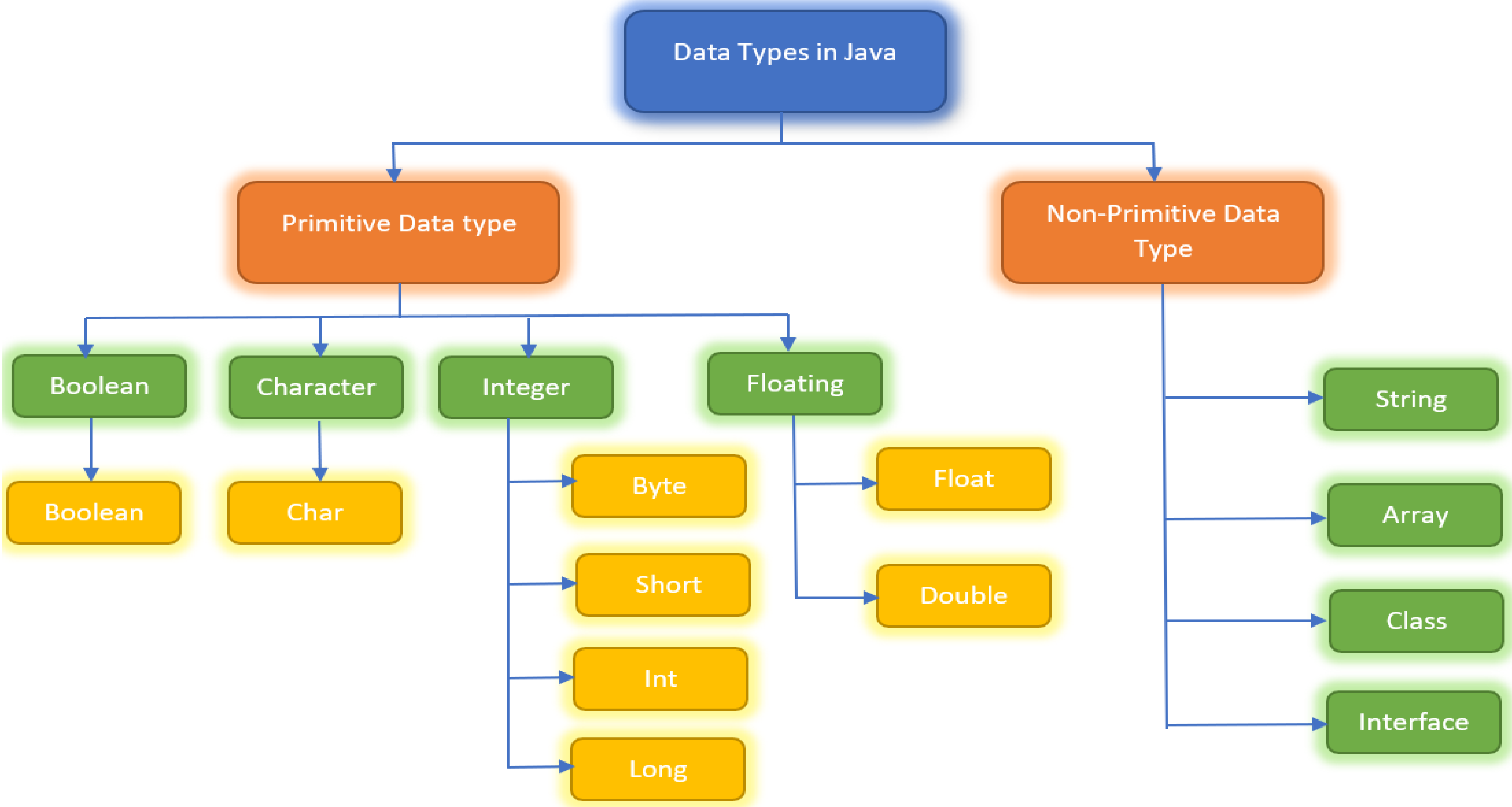
Data Types in Java?

1. Data types specify the different sizes and values that can be stored in the variable.
2. Java is statically typed and also a strongly typed language .
3. Java being statically typed means you have to declare the type of variables before its use.
4. Strongly typed means that Java strictly enforces the use of correct variable types and doesn't allow operations between incompatible types without explicit conversion. statically typed and also a strongly typed language

Categories of Data Types in Java

There are two types of data types in Java:

1. **Primitive data types**:
   - In Java language, primitive data types are the building blocks of data manipulation. These are the most basic data types available in Java language.
   - The primitive data types include boolean, char, byte, short, int, long, float and double.
   - There are 8 types of primitive data types.
2. **Non-primitive data types:**
   - The non-primitive data types include Classes, Interfaces, and Arrays.


Data Types in Java
Primitive Data type
Non-Primitive Data Type
Boolean
Character
Integer
Floating
Boolean
Char
Byte
Short
Int
Long
Float
Double
String
Array
Class
Interface

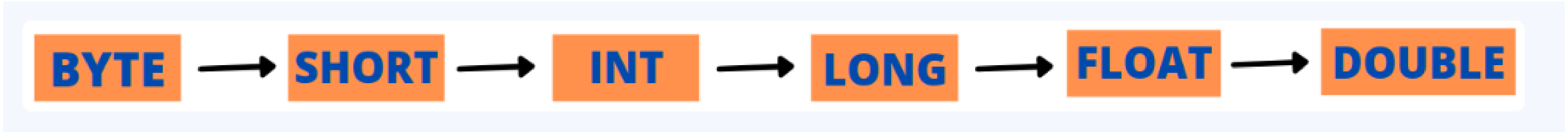
DATA TYPES	SIZE	DEFAULT	EXPLAINATION
boolean	1 bit	false	Stores true or false values
byte	1 byte/ 8bits	0	Stores whole numbers from -128 to 127
short	2 bytes/ 16bits	0	Stores whole numbers from -32,768 to 32,767
int	4 bytes/ 32bits	0	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes/ 64bits	0L	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes/ 32bits	0.0f	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes/ 64bits	0.0d	Stores fractional numbers. Sufficient for storing 15 decimal digits
char	2 bytes/ 16bits	'\u0000'	Stores a single character/letter or ASCII values

Type Conversion/Type Casting in Java

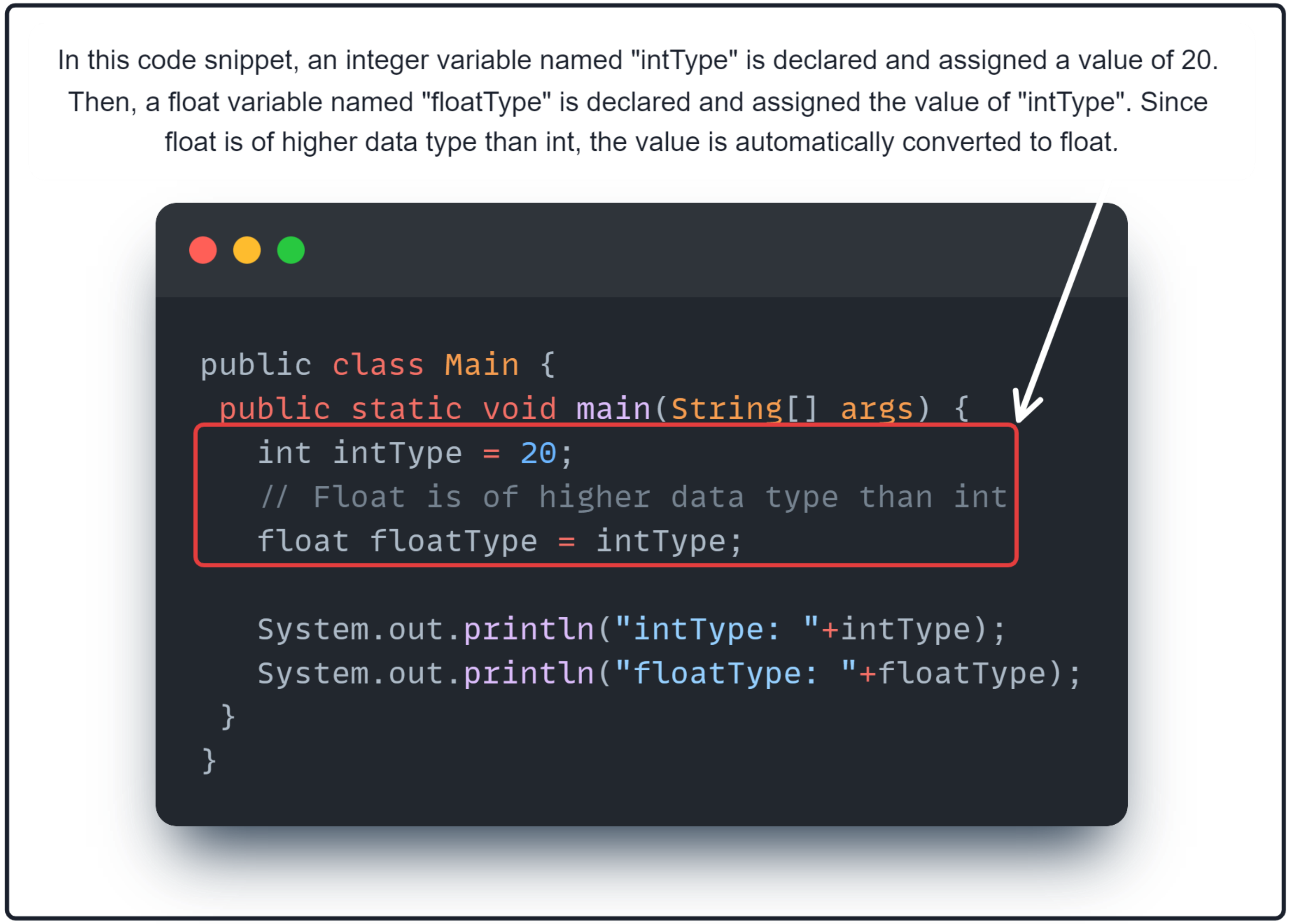
Type Conversion in Java

Type conversion is a process in which the data type is automatically converted into another data type. The compiler does this automatic conversion at compile time. The data type to which the conversion happens is called the destination data type, and the data type from which the conversion happens is called the source data type. If the source and destination data types are compatible, then automatic type conversion takes place.

For type conversion to take place, the destination data type must be larger than the source type. In short, the below flow chart has to be followed.


$$\begin{tikzcd} \text{BYTE} \arrow[r] & \text{SHORT} \arrow[r] & \text{INT} \arrow[r] & \text{LONG} \arrow[r] & \text{FLOAT} \arrow[r] & \text{DOUBLE} \end{tikzcd}$$

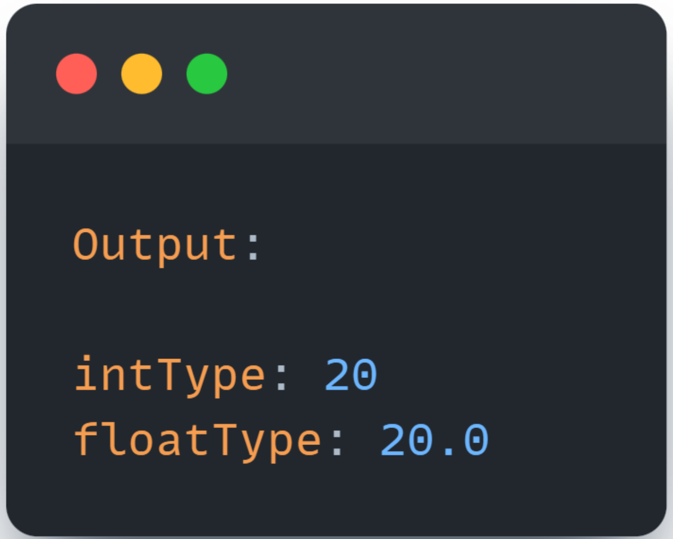
This type of type conversion is also called **Widening Type Conversion/ Implicit Conversion/ Casting Down.** In this case, as the lower data types with smaller sizes are converted into higher ones with a larger size, there is no chance of data loss. This makes it safe for usage.



In this code snippet, an integer variable named "intType" is declared and assigned a value of 20. Then, a float variable named "floatType" is declared and assigned the value of "intType". Since float is of higher data type than int, the value is automatically converted to float.

```
public class Main {
 public static void main(String[] args) {
   int intType = 20;
   // Float is of higher data type than int
   float floatType = intType;

   System.out.println("intType: "+intType);
   System.out.println("floatType: "+floatType);
 }
}
```

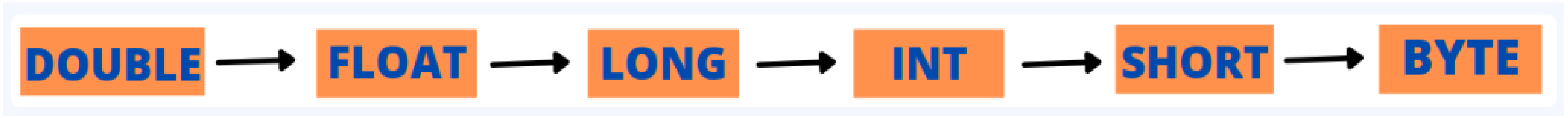


```
Output:

intType: 20
floatType: 20.0
```

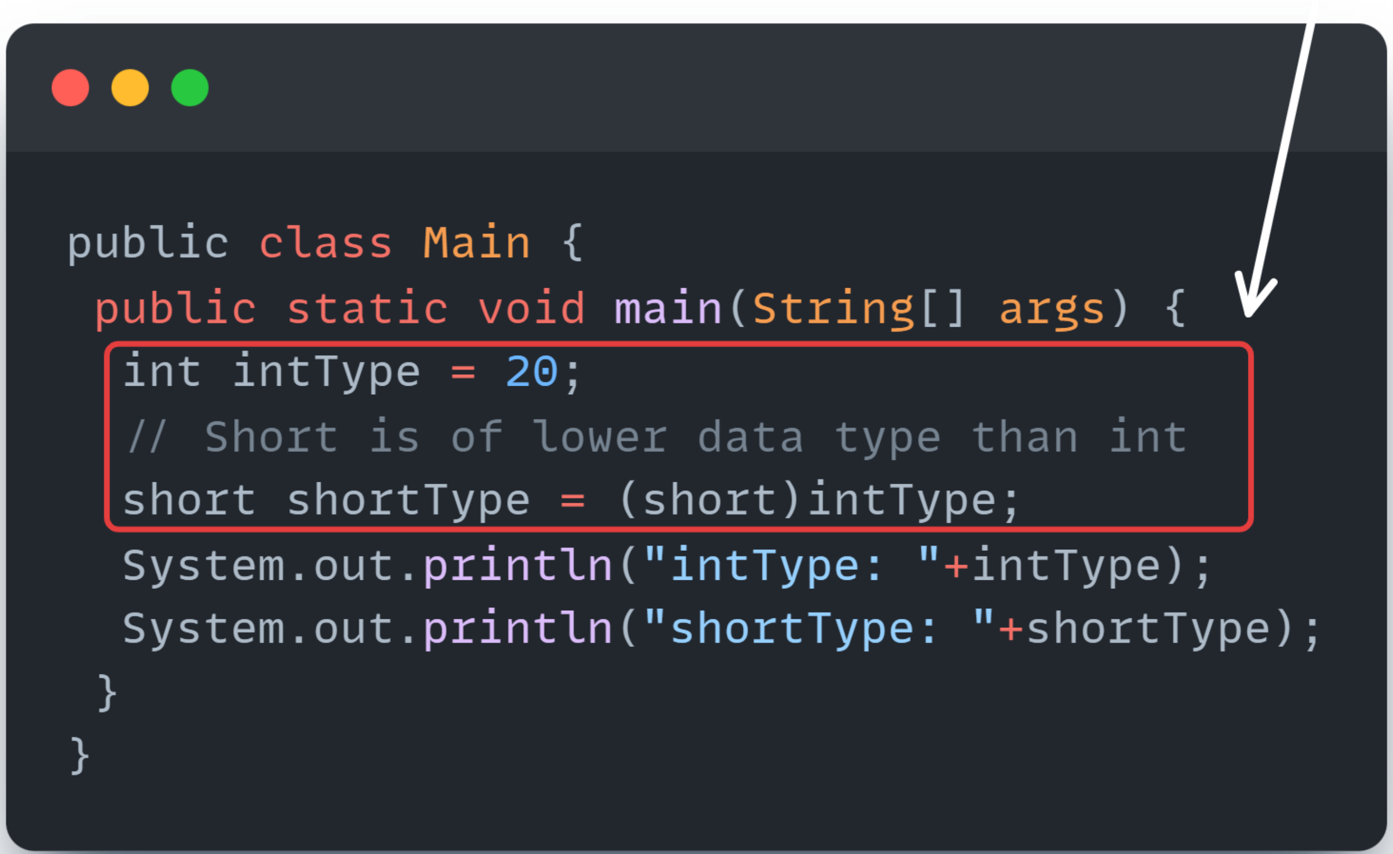
Type Casting in Java

Type casting is a process in which the programmer manually converts one data type into another data type. For this the casting operator (), the parenthesis is used. Unlike type conversion, the source data type must be larger than the destination type in type casting. The below flow chart has to be followed for successful type casting.

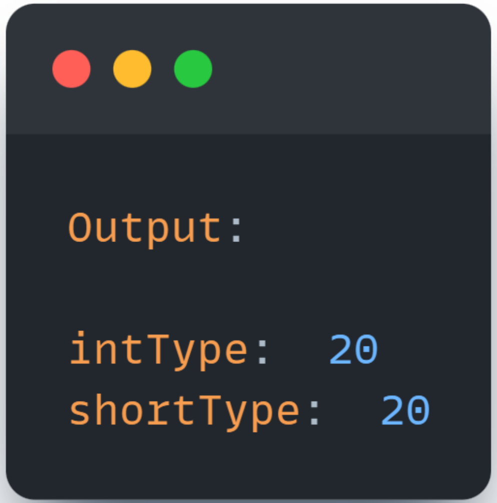


Type casting is also called **Narrowing Type Casting/ Explicit Conversion/ Casting Up**. In this case, as the higher data types with a larger size are converted into lower ones with a smaller size, there is a chance of data loss. This is the reason that this type of conversion does not happen automatically.

In this code, an integer variable named "intType" is assigned the value of 20. Then, the value of "intType" is explicitly casted to a short data type and assigned to the variable "shortType", which may result in a potential loss of precision.



```java
public class Main {
 public static void main(String[] args) {
  int intType = 20;
  // Short is of lower data type than int
  short shortType = (short)intType;
  System.out.println("intType: "+intType);
  System.out.println("shortType: "+shortType);
 }
}
```



```
Output:

intType:  20
shortType:  20
```

CODE FOR CODERS

THANKS FOR WATCHING

DON'T FORGET LIKE❤ COMMENT, SUBSCRIBE,

COODE FOR CODERS