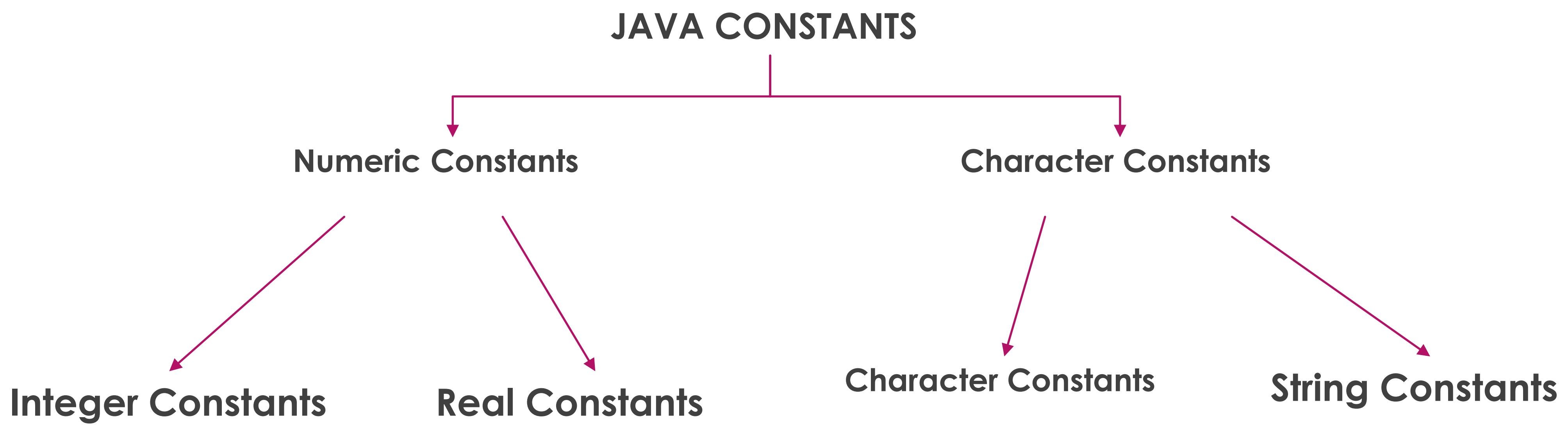
Constants, Variables, and Data Types

Contents:

Constants

- Constants refer to fixed values that do not change during the execution of a program.



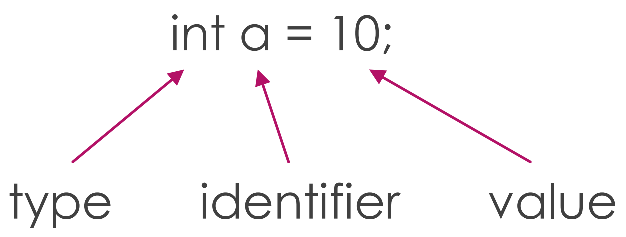
TYPES OF CONSTANTS

- **Integer Constants :** refer to sequence of digits.
- **Real Constants :** Numbers containing decimal part. Ex. 0.025, -2.5
- **Single Character Constants :** Single character enclosed in a quotation marks. Ex. ‘a’,’5’,’ ’,’;’

  Note : ‘5’ means character and 5 means number.
- **String Constants :** Sequence of characters enclosed in double quotation marks. Ex. “Hello Java”, “Tim”, “;!@#$”, “5+7”,”X”
- Note: “5+7” means String whereas 5+7 means 7.
- **Backslash Character Constants :** used in output methods. Ex. \b for backspace, \n for new line, etc.

Variables

- The variable is a storage location used to store data value.
- A variable may take different values at different times during the execution of a program.
- A variable name can be chosen by the programmer in a meaningful way so as to reflect what it represents. E.g. length, breadth, area.



```
int a = 10;
```

- Variable names may contain alphabets, digits, underscore (_) and dollar ($) character.
  - Variable names must not begin with a digit.
  - Uppercase and lowercase are distinct. Ex. Total and total are different.
  - It should not be a keyword.
  - White space is not allowed.
  - Variable names can be of any length.

VARIABLE DECLARATION

- Tells the compiler
  - Name of variable
  - Type of variable
  - Scope of variable
- Syntax for declaration is:

  type variable1,variable2,………, variableN

VARIABLE INITIALIZATION

- **Static Initialization:**

  variables can be initialized using constants

  ```
  e.g. char c = 'M';
   or, char c;
       c = 'M';
  ```

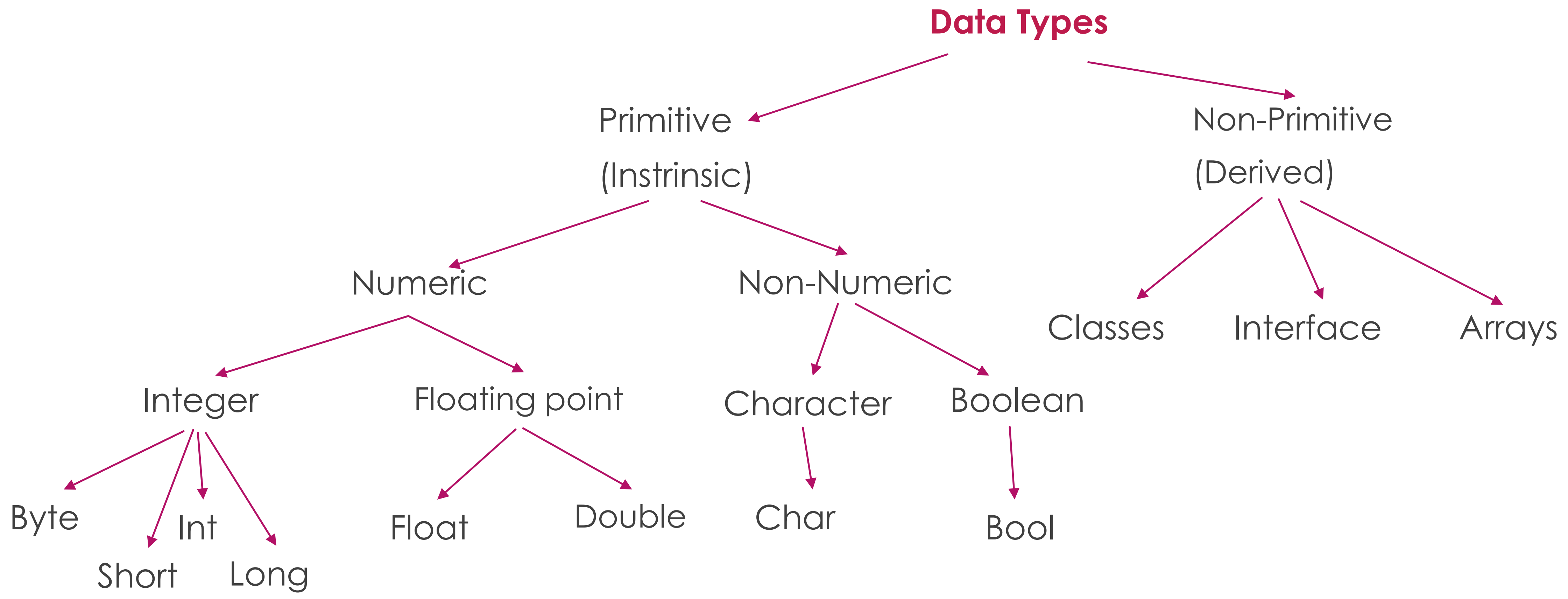
- **Dynamic Initialization:**

  Java allows variables to be initialized dynamically, using any expression valid at the time the variable is declared.

```
class Initialize
{
        public static void main(String args[])
        {
                // a and b are statically initialized
                double a = 3.0, b = 4.0;
                // c is dynamically initialized
                double c = Math.sqrt(a * a + b * b);
                System.out.println("Hypotenuse is " + c);
        }
}
```

Data Types

- Java is a strongly typed language.
- It means:
  - *Every variable has a type*
  - *Every expression has a type, and every type is strictly defined*
- Data types specify the size and type of value that is stored.
- All assignments, whether explicit or via parameter passing in method calls, are checked for type compatibility.
- Any type mismatches are errors.


Data Types
Primitive
(Instrinsic)
Non-Primitive
(Derived)
Numeric
Non-Numeric
Classes
Interface
Arrays
Integer
Floating point
Character
Boolean
Byte
Short
Int
Long
Float
Double
Char
Bool

Integers

- Java defines four integer types: byte, short, int, and long.
- All of these are signed, positive and negative values.
- Java does not support unsigned, positive-only integers.

Type	Width (Bytes)	Width (Bits)	Range
byte	1	8	-2^{8-1} to 2^{8-1}
short	2	16	-2^{16-1} to 2^{16-1}
Int	4	32	-2^{32-1} to 2^{32-1}
Long	8	64	-2^{64-1} to 2^{64-1}

Floating-Points

- Floating-point numbers, also known as **real numbers**, are used when evaluating expressions that require **fractional precision**.
- There are two kinds of floating-point types, **float** and **double**, which represent single- and double-precision numbers, respectively.
- By default floating point constants are double type in java.

Type	Width (Bytes)	Width (Bits)	Range
Float	4	32	3.4e-038 to 1.7e+0.38
Double	8	64	3.4e-038 to 1.7e+308

Float

- Float specifies a single-precision value that uses **32 bits** of storage.
- Single precision is faster on some processors and takes half as much space as double precision.
- Floating points are treated as double precision quantities. To force them to be in single precision, we must **append f or F** to the numbers.

For Example :

```
float f = 1.23f
```

Double

- Double provides high precision, and uses **64 bits** to store a value.
- Double precision is actually faster than single precision on some modern processors.
- All transcendental math functions, such as sin( ), cos( ), and sqrt( ), return double values.
- Double is useful when we need to maintain **accuracy over many iterative calculations**.

Characters

- Data type used to store characters is **char.**
- Unlike C/C++ (8 bits), Java char is a **16-bit** type.
- The range of a char is **0 to 65,536**.
- There are no negative chars.
- char variables behave like **integers** (as shown in the example).

```
class CharTest{
        public static void main(String args[])
        {
                char c1;
                c1 = ‘A’;
                System.out.println("c1 is currently " + c1);
                c1++;
                System.out.println("c1 is now " + c1);
        }
}
```

Output : c1 is currently A

c1 is now B

Boolean

- Boolean can have only one of two possible values, **true or false**.
- This is the return type of all relational operators.

  e.g. **a < b (either true or false)**

- Boolean is also the type required by the conditional expressions that govern the control statements such as if and for.

  e.g. **if (x == 5)**

Types of java Variables

- Instance Variable
- Class/ Static Variable
- Local Variable

- **Local variables** – Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.
- **Instance variables** – Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.
- **Class variables** – Class variables are variables declared within a class, outside any method, with the static keyword.

```
Class Demo
{
        int a=10;            //Instance Variable
        static int b=20; //Class/Static Variable
        void sumOf Numbers()
        {
                int c;          //Local Variable
                c=a+b;
                System.out.println("Sum is : "+c);
        }
}
```

Symbolic Constants

- The values of the constant can't be changed once its declared.
- Constants are required to:
  - Prevent modifiability
  - Understandability
- Constants are declared using the final keyword.
- Even though Java does not have a constant type, we can achieve the same effect by declaring and initializing variables that are static, public, and final.
- Example:
  - *final* int NUMBER_OF_HOURS_IN_A_DAY = 24;

Type Casting and Type Conversion

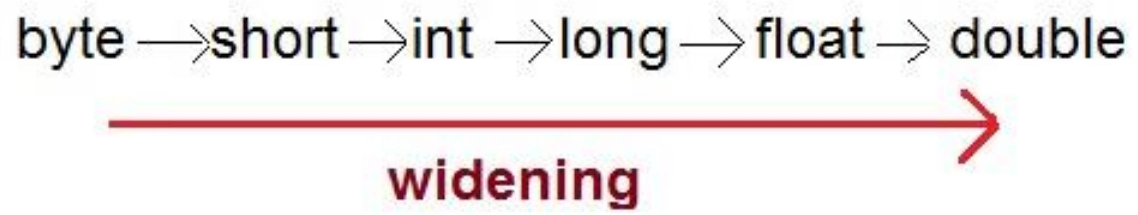
- **Type Conversion (Implicit) :** is that which automatically converts the one data type into another.
- we can store a large data type into the other.

  For ex. we can t store a float into int because a float is greater than int.
- **Type Casting (Explicit) :** User can convert the one higher data type into lower data type then it is called as the type casting.

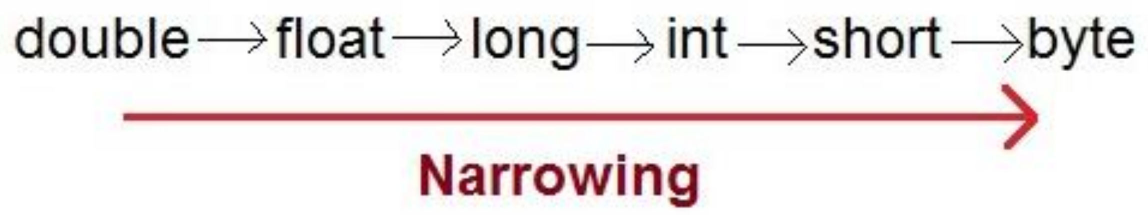
  For ex converting a float into int.

Type Conversion is performed by the compiler but a casting is done by the user.

- Widening Casting(Implicit)



- Narrowing Casting(Explicitly done)



Widening or Automatic type conversion

- Automatic Type casting take place when,
  - the two types are compatible
  - the target type is larger than the source type

```
public class Test
{
    public static void main(String[] args)
    {
      int i = 100;
      long l = i;          //no explicit type casting required
      float f = l;         //no explicit type casting required
      System.out.println("Int value "+i);
      System.out.println("Long value "+l);
      System.out.println("Float value "+f);
    }
}
```

Output :

```
Int value 100
Long value 100
Float value 100.0
```

Narrowing or Explicit type conversion

```
public class Test
{
   public static void main(String[] args)
   {
     double d = 100.04;
     long l = (long)d;  //explicit type casting required
     int i = (int)l;  //explicit type casting required

     System.out.println("Double value "+d);
     System.out.println("Long value "+l);
     System.out.println("Int value "+i);
    }
   }
```

Output :

```
Double value 100.04
Long value 100
Int value 100
```

Standard Default Values

Types of variable	*Default value*
Byte	0
Short	0
Int	0
Long	0
Float	0.0f
Double	0.0d
Char	null
Boolean	false