Java environment setup and java basics

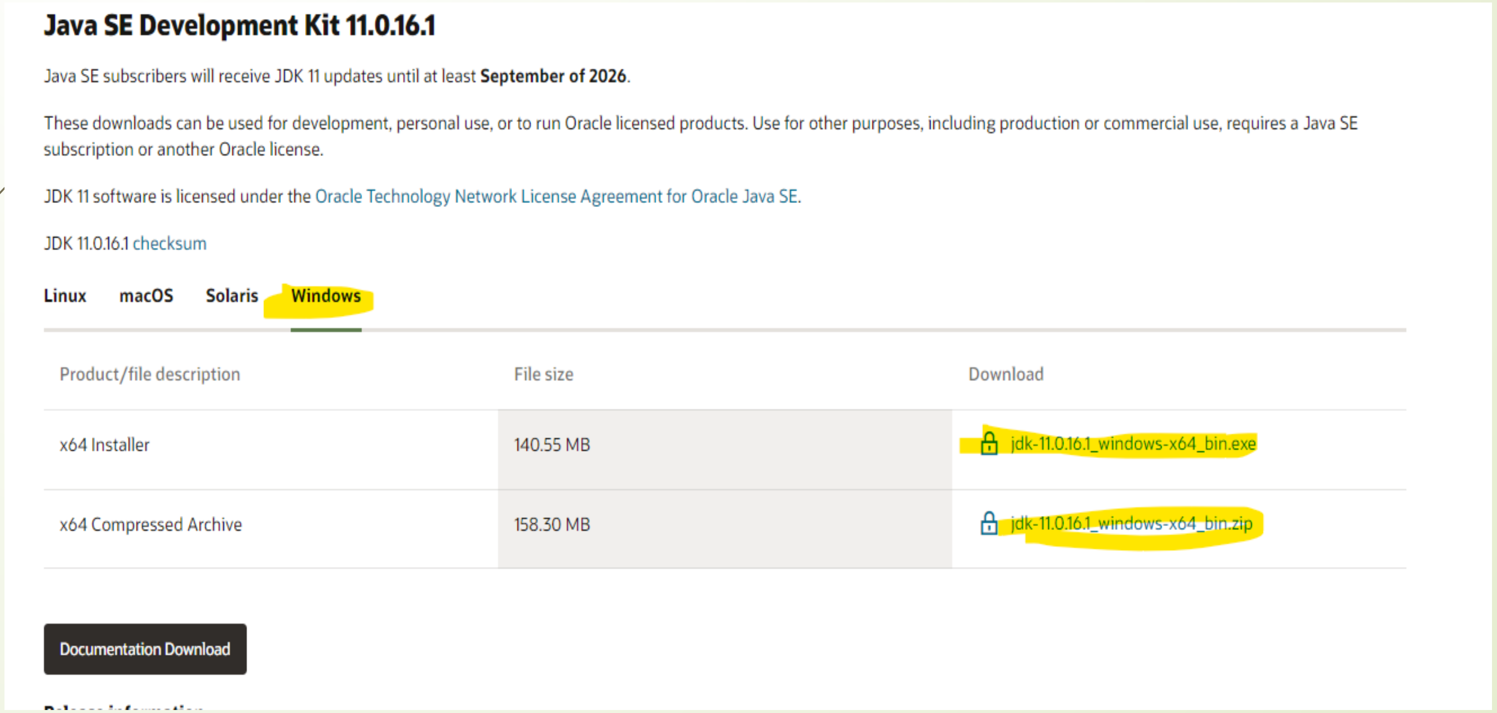
Java Features

- Simple & robust
- Secure
- Portable
- Platform independent, Hardware independent and Architecture neutral
- WORA (Write Once and Run Anywhere)
- Full OOP support
- Exclusive threading support
- Interpreted
- Automatic memory management (garbage collection)
- Functional programming support

Java installation steps

- Download JDK 11 from Oracle official site

  https://www.oracle.com/java/technologies/downloads/#java11



Java SE Development Kit 11.0.16.1

Java SE subscribers will receive JDK 11 updates until at least **September of 2026**.

These downloads can be used for development, personal use, or to run Oracle licensed products. Use for other purposes, including production or commercial use, requires a Java SE subscription or another Oracle license.

JDK 11 software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE.

JDK 11.0.16.1 checksum

Linux macOS Solaris Windows

Product/file description	File size	Download
x64 Installer	140.55 MB	jdk-11.0.16.1_windows-x64_bin.exe
x64 Compressed Archive	158.30 MB	jdk-11.0.16.1_windows-x64_bin.zip

Documentation Download

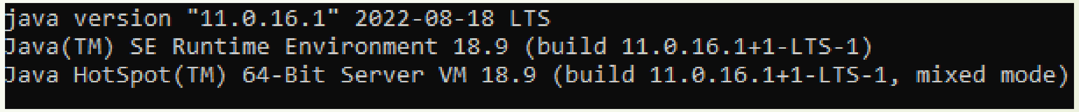
Java installation steps

- Install Java using installer if exe is downloaded or extract if zip is downloaded
- Setting up path on windows
  - Add jdk installation directory path in path variable

- Check java version post installation with below command

  *Command: **java –version***

  java is command/executable name, –version is option



```
java version "11.0.16.1" 2022-08-18 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.16.1+1-LTS-1)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.16.1+1-LTS-1, mixed mode)
```

- Download Java Docs

  Direct : https://docs.oracle.com/en/java/javase/11/docs/api/index.html

  Downloadable : https://www.oracle.com/java/technologies/javase-jdk11-doc-downloads.html

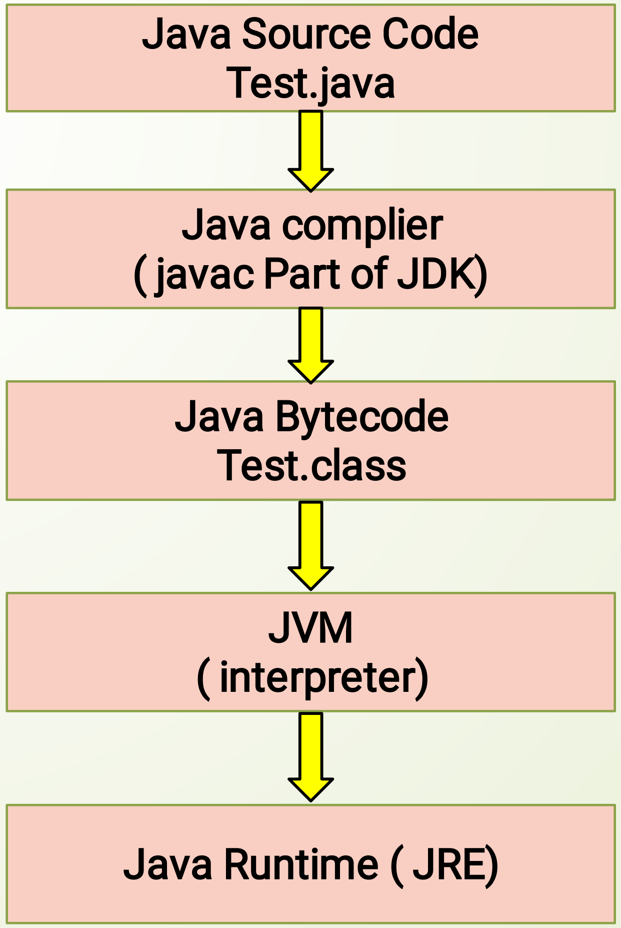
JRE, JVM and JDK

- JRE :
  - Java API Libraries for running the application + JVM (Java Virtual Machine)
    ex. rt.jar
  - Contains packaged class like java.lang, java.util
  - JRE has a major responsibility for creating an environment for the execution of code.
- JVM :
  - Loads classes using class loaders
  - Helps in executing the Java bytecode.
  - It has interpreter to interpret byte code
  - I comes with Just-in-Time (JIT) compiler for converting the Java source code into a low-level machine language.

JRE, JVM and JDK

- **JDK :**
  - The JDK enables developers to create Java programs that can be executed and run by the JRE and JVM
  - JRE + dev tools for developing, debugging, and monitoring java code. Ex javac, javap, javah, jar, keytool etc.

Java compilation and execution



Java keywords

abstract	Specifies that a class or method will be implemented later, in a subclass
assert	Assert describes a predicate placed in a java program to indicate that the developer thinks that the predicate is always true at that place.
boolean	A data type that can hold True and False values only
break	A control statement for breaking out of loops.
byte	A data type that can hold 8-bit data values
case	Used in switch statements to mark blocks of text
catch	Catches exceptions generated by try statements
char	A data type that can hold unsigned 16-bit Unicode characters
class	Declares a new class
continue	Sends control back outside a loop
default	Specifies the default block of code in a switch statement
do	Starts a do-while loop

Java keywords

double	A data type that can hold 64-bit floating-point numbers
else	Indicates alternative branches in an if statement
enum	A Java keyword is used to declare an enumerated type. Enumerations extend the base class.
extends	Indicates that a class is derived from another class or interface
final	Indicates that a variable holds a constant value or that a method will not be overridden
finally	Indicates a block of code in a try-catch structure that will always be executed
float	A data type that holds a 32-bit floating-point number
for	Used to start a for loop
if	Tests a true/false expression and branches accordingly
Implements	Specifies that a class implements an interface
import	References other classes
instanceof	Indicates whether an object is an instance of a specific class or implements an interface

Java keywords

int	A data type that can hold a 32-bit signed integer
interface	Declares an interface
long	A data type that holds a 64-bit integer
native	Specifies that a method is implemented with native (platform-specific) code
new	Creates new objects
null	This indicates that a reference does not refer to anything
package	Declares a Java package
private	An access specifier indicating that a method or variable may be accessed only in the class it's declared in
protected	An access specifier indicating that a method or variable may only be accessed in the class it's declared in and its sub-classes
public	An access specifier used for classes, interfaces, methods, and variables indicating that an item is accessible throughout the application/anywhere
return	Sends control and possibly a return value back from a called method
short	A data type that can hold a 16-bit integer

Java keywords

static	Indicates that a variable or method is a class method
strictfp point	A Java keyword is used to restrict the precision and rounding of floating-calculations to ensure portability.
super	Refers to a class's base class (used in a method or class constructor)
switch	A statement that executes code based on a test value
synchronized	Specifies critical sections or methods in multithreaded code
this	Refers to the current object in a method or constructor
throw	Creates an exception
throws	Indicates what exceptions may be thrown by a method
transient	Specifies that a variable is not part of an object's persistent state
try	Starts a block of code that will be tested for exceptions
void	Specifies that a method does not have a return value
volatile	This indicates that a variable may change asynchronously
while	Starts a while loop

Hello java program

```
public class HelloJava {
    public static void main(String[] args) {
        System.out.println("Hello Java");
    }
}
```

```
//Below takes value form command line arguments
public class HelloJava {
    public static void main(String[] args) {
        System.out.println("Hello " + args[0]);
    }
}
```

Java statements and block of statements

- **Statements**

  A statement is a java code terminated by a semi-colon which can execute.

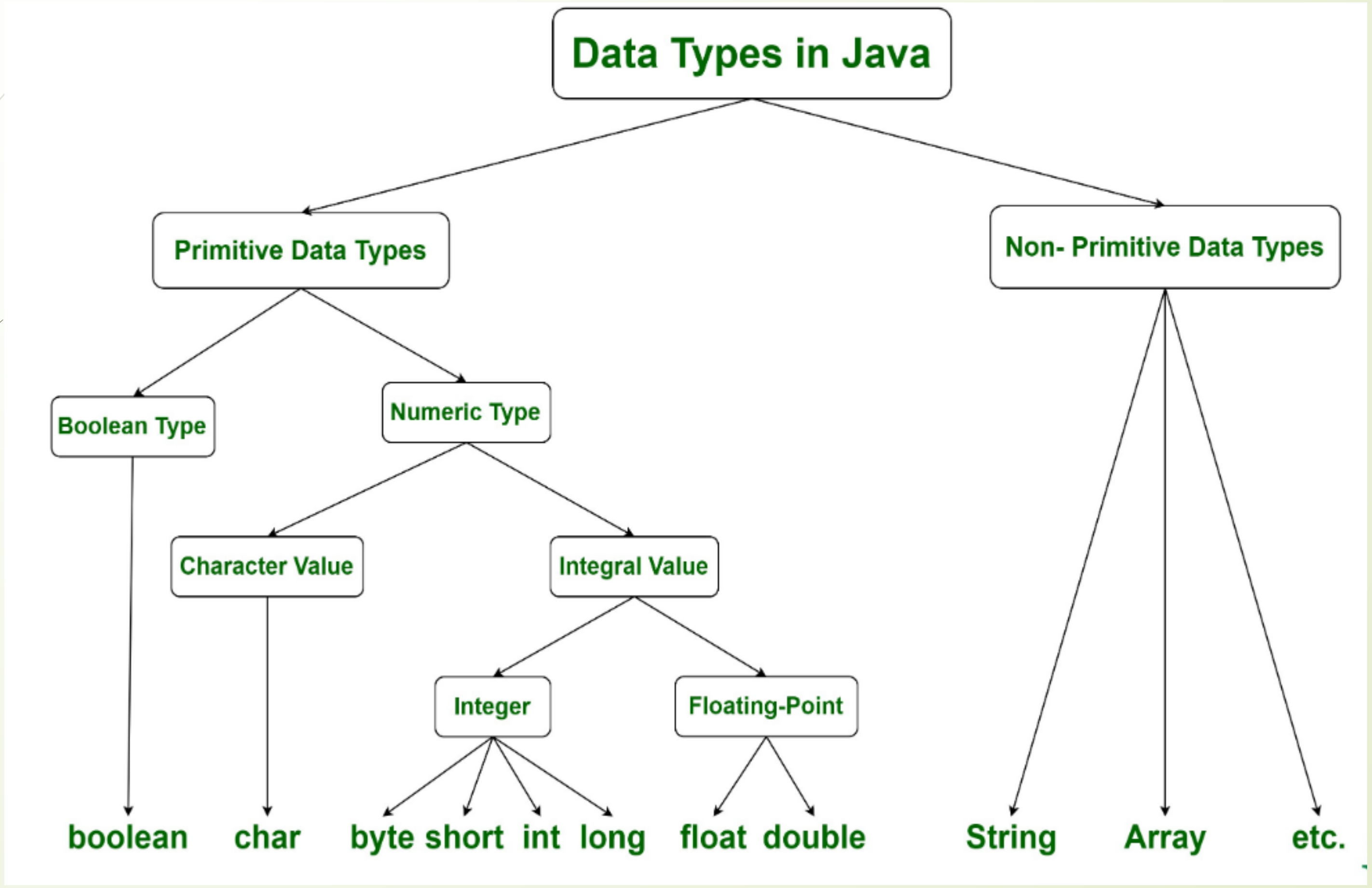
  Ex. System.out.println("hello");

- **Block of statements**
  - It contains more than one statements enclosed in curly brackets.
  - Blocks can be nested: block within block

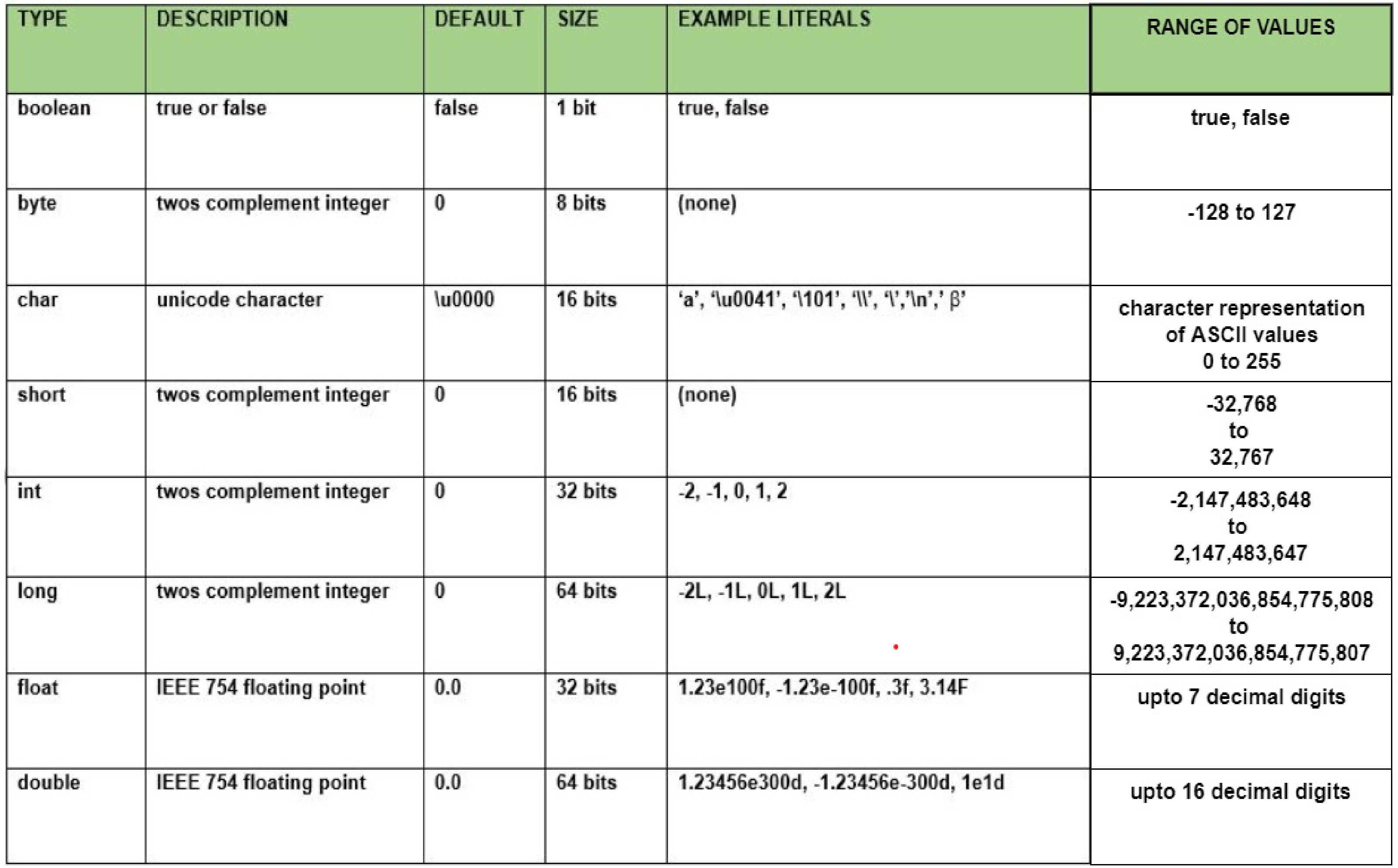
  Ex.

```
{
System.out.println(“Hello");
System.out.println(“Java");
 }
```

Data types in Java



Data types in Java



TYPE	DESCRIPTION	DEFAULT	SIZE	EXAMPLE LITERALS	RANGE OF VALUES
boolean	true or false	false	1 bit	true, false	true, false
byte	twos complement integer	0	8 bits	(none)	-128 to 127
char	unicode character	\u0000	16 bits	‘a’, ‘\u0041’, ‘\101’, ‘\\’, ‘\’,’\n’,’ β’	character representation of ASCII values 0 to 255
short	twos complement integer	0	16 bits	(none)	-32,768 to 32,767
int	twos complement integer	0	32 bits	-2, -1, 0, 1, 2	-2,147,483,648 to 2,147,483,647
long	twos complement integer	0	64 bits	-2L, -1L, 0L, 1L, 2L	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	IEEE 754 floating point	0.0	32 bits	1.23e100f, -1.23e-100f, .3f, 3.14F	upto 7 decimal digits
double	IEEE 754 floating point	0.0	64 bits	1.23456e300d, -1.23456e-300d, 1e1d	upto 16 decimal digits

2’s Compliment

- Java stores integer types in 2’s compliment format
- 2’s Compliment = 1’s Compliment + 1
- Ex. Decimal 40

```
40 to binary   ->     0010 1000
1’s Compliment ->  1101 0111
                 + 1  __________
            1101  1000
```

Size of byte is 8 and 1 digit is signed so range will be 2^7 to ((2^7)-1) = -128 to 127

Format specifiers for datatypes

Format Specifier	Conversion Applied
%%	Inserts a % sign
%x %X	Integer hexadecimal
%t %T	Time and Date
%s %S	String
%n	Inserts a newline character
%o	Octal integer
%f	Decimal floating-point
%e %E	Scientific notation
%g	Causes Formatter to use either %f or %e, whichever is shorter
%h %H	Hash code of the argument
%d	Decimal integer
%c	Character
%b %B	Boolean
%a %A	Floating-point hexadecimal

Variable declaration and initialization

- Variable is saved In memory which value varies withing the rage of values.
- Range of value depends upon the datatype of variable
- Syntax

  <datatype> variableName;

Variable declaration/definition	Variable initialization and assignment

int number;
double balance;
float percentage;
char c;
	int number = 100; // Init
	int number1 = 50; // Init
	number1 = number; //Assignment
	double balance = 100.50;
	char ch= ‘A’;
	float percentage = 10.50F;

Rules for Identifier names

- Allowed characters for identifiers are all alphanumeric characters([A-Z],[a-z],[0-9]), '$'(dollar sign) and '_' (underscore).
- Identifiers should not start with digits([0-9]).
- Java identifiers are case-sensitive.
- Reserved Words/Keywords can't be used as an identifier.

Valid Identifiers	Invalid identifies
number, $balance, empName, dept_name, PI, EMPTY, str, out etc.	#number, no@, 123number, final, null etc.

Naming convention

- **For classes and interfaces:**
  - Names should begin with a capital letter. And if there are multiple words in the class name then each word must also begin with a capital letter.
  - It follows UpperCamelCase notations.

  Ex. String, HelloJava, Scanner, GregorianCalendar, HashMap, Comparable, Comparator etc.
- **For packages:**
  - Also package names always start with lowercase characters And if there are multiple words in the package name, then you need to use uppercase for all words except for the starting word.

  Ex. util, lang, io, nio etc.
- **For data members**
  - Instances and other variables names must start with lowercase and if there are multiple words in the name, then you need to use Uppercase for starting letters for the words except for the starting word.
  - It follows as lowerCamelCase.

  Ex. empName, index, calculatePerimeter, balcance, assetsValue etc.
- **For constants**
  - Constants/finals shold have all letters capital/ uppercase.

  Ex. PI, EMPTY, HEX, APP_NAME etc

Few more Java rules

- Variables must be initialized before use. Un-initialized data members and local variables gives error in compilation
- A .java file can have more than one non public classes but only one public per file
- If there is a public class in a file, the name of the file must match the name of the public class.
- Ex. a class declared as

```
public class Test {
//some code goes here
}
```

must be in a source code file named **Test.java.**

- Ex : int number;

```
      number++; // Un-initialized  variable usage is error
```

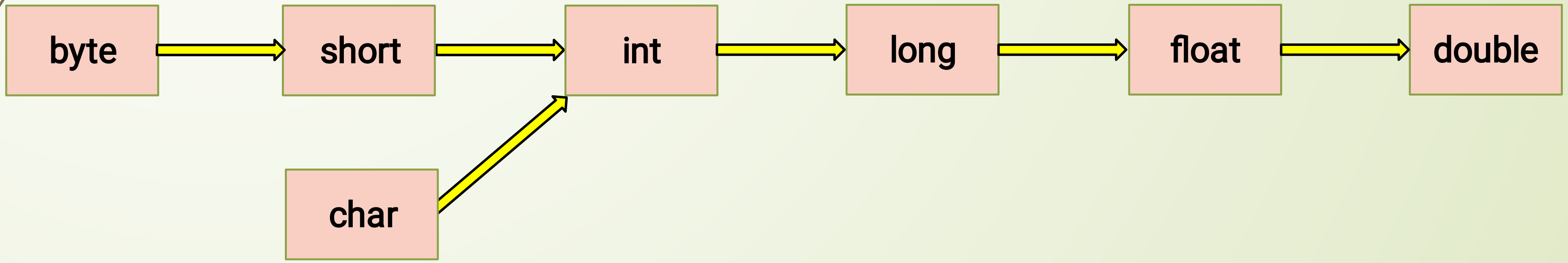
Programs demo

- Print integer, float, double and char using System.out.printf methods
- Program to add 2 numbers using command line arguments
- Program to add 2 numbers by getting inputs from user
- Print size of data types
- Demonstrate local variable, class/static variable and instance variables

Widening and narrowing for primitive data types

- **Widening (Automatic promotion)**– Converting a lower datatype to a higher datatype is known as widening.
  - Ex. int to long
- **Narrowing (type-casting)** – Converting a higher datatype to a lower datatype is known as narrowing.

  Ex. float to int



$$\begin{tikzcd} byte \arrow[r] & short \arrow[r] & int \arrow[r] & long \arrow[r] & float \arrow[r] & double \\ & char \arrow[ur] & & & & \end{tikzcd}$$

Programs Demo

- Print ASCII values of char
- Test widening and narrowing of primitive data types
- Binary Literals

Operators

- Arithmetic Operators
- Unary Operators
- Assignment Operator
- Relational Operators
- Logical Operators
- Ternary Operator
- Bitwise Operators
- Shift Operators
- instance of Operator

Arithmetic Operators

- **Arithmetic Operators:** Used to perform simple arithmetic operations on primitive data types.

  *** : Multiplication**

  / : Division

  % : Modulo

  + : Addition

  – : Subtraction

Unary Operators

- **Unary Operators:** Unary operators need only one operand. They are used to increment, decrement or negate a value.
  - – : **Unary minus**, used for negating the values.
  - + : **Unary plus** indicates the positive value
  - ++, -- : **Increments and Decrement operator**. They cab be prefix and postfix
  - ! : **Logical not operator**, used for inverting a boolean value.

Assignment Operator

'=' : Assignment operator is used to assigning a value to any variable. It has a right to left associativity
Ex. i = 10; i = i +10;

Compound Statement/Shorthand. For example, instead of i = i+10, we can write i += 10;

+=, for adding left operand with right operand and then assigning it to the variable on the left.

-=, for subtracting right operand from left operand and then assigning it to the variable on the left.

***=**, for multiplying left operand with right operand and then assigning it to the variable on the left.

/=, for dividing left operand by right operand and then assigning it to the variable on the left.

%=, for assigning modulo of left operand by right operand and then assigning it to the variable on the left.

Ex. value += 10;

Relational Operators

Relational Operators: These operators are used to check for relations like equality, greater than, and less than. They return boolean results after the comparison

- **==, Equal to** returns true if the LHS is equal to the RHS.
- **!=, Not Equal to** returns true if the LHS is not equal to the RHS.
- **<, less than:** returns true if the LHS is less than the RHS.
- **<=, less than or equal to** returns true if the LHS is less than or equal to the RHS.
- **>, Greater than:** returns true if the LHS is greater than the RHS.
- **>=, Greater than or equal to** returns true if the LHS is greater than or equal to the RHS.

Logical Operators and Ternary

Logical Operators: These operators are used to perform “logical AND” and “logical OR” operations,
&&, Logical AND: returns true when both conditions are true.
||, Logical OR: returns true if at least one condition is true.
!, Logical NOT: returns true when a condition is false and vice-versa

Ternary operator: Ternary operator is a shorthand version of the if-else statement.
Syntax
condition ? if true : if false

Bitwise, Shift and instance of Operators

Bitwise Operators: These operators are used to perform the manipulation of individual bits of a number.

&, Bitwise AND operator: returns bit by bit AND of input values.
|, Bitwise OR operator: returns bit by bit OR of input values.
^, Bitwise XOR operator: returns bit-by-bit XOR of input values.
~, Bitwise Complement Operator: This is a unary operator which returns the one's complement representation of the input value.

Shift Operators: These operators are used to shift the bits of a number left or right, thereby multiplying or dividing the number by two, respectively.

<<, Left shift operator: shifts the bits of the number to the left and fills 0 on voids left as a result. Similar effect as multiplying the number with some power of two.
>>, Signed Right shift operator: shifts the bits of the number to the right and fills 0 on voids left as a result. The leftmost bit depends on the sign of the initial number. Similar effect as dividing the number with some power of two.
>>>, Unsigned Right shift operator: shifts the bits of the number to the right and fills 0 on voids left as a result. The leftmost bit is set to 0.

instance of Operators

instanceof operator: The instance of the operator is used for type checking. It can be used to test if an object is an instance of a class, a subclass, or an interface.