**Java Basics: Revision Notes**

**Table of Contents**

1. [Introduction to Output in Java](https://companion.scaler.com/meeting/class/858012/notes?_gl=1*1aitnjq*_gcl_gb*R0NMLjE3NDE3MTE0NjQuQ2xrS0NBandtN3EtQmhCbEVra0F4aEFCZU5kUTB0NmFPckdKNlJ2bEQtSWVVLVRaZGo0TmdPTXE5enFGWElNUEMyVFRhMDZoUmFSbWtheE5iMHowcjRHOVRCQlRULUh4cEVTbkZCMUM0eDMzZGQ0enA5Qm5HZ0p5eGcuc0dDakNJLTZ4OGdDRUx5XzA2RUQuQlpJRUNNU2EwWmtaRUx5XzA2RUQ.*_gcl_au*MTIyMjk0OTUwOS4xNzM4MzM3MDIxLjE3ODE5NzQ1NTguMTc0NDI0Mzc1Mi4xNzQ0MjQzNzUx*FPAU*MTIyMjk0OTUwOS4xNzM4MzM3MDIx*_ga*MTEwOTQzNzc1OC4xNjkxODM5NDAy*_ga_53S71ZZG1X*MTc0NDU1Nzk4My4yNDguMS4xNzQ0NTU5MTczLjAuMC4zNzM5MzIwODc.#introduction-to-output-in-java)
2. [Printing Strings and Numbers](https://companion.scaler.com/meeting/class/858012/notes?_gl=1*1aitnjq*_gcl_gb*R0NMLjE3NDE3MTE0NjQuQ2xrS0NBandtN3EtQmhCbEVra0F4aEFCZU5kUTB0NmFPckdKNlJ2bEQtSWVVLVRaZGo0TmdPTXE5enFGWElNUEMyVFRhMDZoUmFSbWtheE5iMHowcjRHOVRCQlRULUh4cEVTbkZCMUM0eDMzZGQ0enA5Qm5HZ0p5eGcuc0dDakNJLTZ4OGdDRUx5XzA2RUQuQlpJRUNNU2EwWmtaRUx5XzA2RUQ.*_gcl_au*MTIyMjk0OTUwOS4xNzM4MzM3MDIxLjE3ODE5NzQ1NTguMTc0NDI0Mzc1Mi4xNzQ0MjQzNzUx*FPAU*MTIyMjk0OTUwOS4xNzM4MzM3MDIx*_ga*MTEwOTQzNzc1OC4xNjkxODM5NDAy*_ga_53S71ZZG1X*MTc0NDU1Nzk4My4yNDguMS4xNzQ0NTU5MTczLjAuMC4zNzM5MzIwODc.#printing-strings-and-numbers)
3. [String Concatenation](https://companion.scaler.com/meeting/class/858012/notes?_gl=1*1aitnjq*_gcl_gb*R0NMLjE3NDE3MTE0NjQuQ2xrS0NBandtN3EtQmhCbEVra0F4aEFCZU5kUTB0NmFPckdKNlJ2bEQtSWVVLVRaZGo0TmdPTXE5enFGWElNUEMyVFRhMDZoUmFSbWtheE5iMHowcjRHOVRCQlRULUh4cEVTbkZCMUM0eDMzZGQ0enA5Qm5HZ0p5eGcuc0dDakNJLTZ4OGdDRUx5XzA2RUQuQlpJRUNNU2EwWmtaRUx5XzA2RUQ.*_gcl_au*MTIyMjk0OTUwOS4xNzM4MzM3MDIxLjE3ODE5NzQ1NTguMTc0NDI0Mzc1Mi4xNzQ0MjQzNzUx*FPAU*MTIyMjk0OTUwOS4xNzM4MzM3MDIx*_ga*MTEwOTQzNzc1OC4xNjkxODM5NDAy*_ga_53S71ZZG1X*MTc0NDU1Nzk4My4yNDguMS4xNzQ0NTU5MTczLjAuMC4zNzM5MzIwODc.#string-concatenation)
4. [Input Handling in Java](https://companion.scaler.com/meeting/class/858012/notes?_gl=1*1aitnjq*_gcl_gb*R0NMLjE3NDE3MTE0NjQuQ2xrS0NBandtN3EtQmhCbEVra0F4aEFCZU5kUTB0NmFPckdKNlJ2bEQtSWVVLVRaZGo0TmdPTXE5enFGWElNUEMyVFRhMDZoUmFSbWtheE5iMHowcjRHOVRCQlRULUh4cEVTbkZCMUM0eDMzZGQ0enA5Qm5HZ0p5eGcuc0dDakNJLTZ4OGdDRUx5XzA2RUQuQlpJRUNNU2EwWmtaRUx5XzA2RUQ.*_gcl_au*MTIyMjk0OTUwOS4xNzM4MzM3MDIxLjE3ODE5NzQ1NTguMTc0NDI0Mzc1Mi4xNzQ0MjQzNzUx*FPAU*MTIyMjk0OTUwOS4xNzM4MzM3MDIx*_ga*MTEwOTQzNzc1OC4xNjkxODM5NDAy*_ga_53S71ZZG1X*MTc0NDU1Nzk4My4yNDguMS4xNzQ0NTU5MTczLjAuMC4zNzM5MzIwODc.#input-handling-in-java)
5. [Data Types: An Overview](https://companion.scaler.com/meeting/class/858012/notes?_gl=1*1aitnjq*_gcl_gb*R0NMLjE3NDE3MTE0NjQuQ2xrS0NBandtN3EtQmhCbEVra0F4aEFCZU5kUTB0NmFPckdKNlJ2bEQtSWVVLVRaZGo0TmdPTXE5enFGWElNUEMyVFRhMDZoUmFSbWtheE5iMHowcjRHOVRCQlRULUh4cEVTbkZCMUM0eDMzZGQ0enA5Qm5HZ0p5eGcuc0dDakNJLTZ4OGdDRUx5XzA2RUQuQlpJRUNNU2EwWmtaRUx5XzA2RUQ.*_gcl_au*MTIyMjk0OTUwOS4xNzM4MzM3MDIxLjE3ODE5NzQ1NTguMTc0NDI0Mzc1Mi4xNzQ0MjQzNzUx*FPAU*MTIyMjk0OTUwOS4xNzM4MzM3MDIx*_ga*MTEwOTQzNzc1OC4xNjkxODM5NDAy*_ga_53S71ZZG1X*MTc0NDU1Nzk4My4yNDguMS4xNzQ0NTU5MTczLjAuMC4zNzM5MzIwODc.#data-types-an-overview)
6. [Typecasting in Java](https://companion.scaler.com/meeting/class/858012/notes?_gl=1*1aitnjq*_gcl_gb*R0NMLjE3NDE3MTE0NjQuQ2xrS0NBandtN3EtQmhCbEVra0F4aEFCZU5kUTB0NmFPckdKNlJ2bEQtSWVVLVRaZGo0TmdPTXE5enFGWElNUEMyVFRhMDZoUmFSbWtheE5iMHowcjRHOVRCQlRULUh4cEVTbkZCMUM0eDMzZGQ0enA5Qm5HZ0p5eGcuc0dDakNJLTZ4OGdDRUx5XzA2RUQuQlpJRUNNU2EwWmtaRUx5XzA2RUQ.*_gcl_au*MTIyMjk0OTUwOS4xNzM4MzM3MDIxLjE3ODE5NzQ1NTguMTc0NDI0Mzc1Mi4xNzQ0MjQzNzUx*FPAU*MTIyMjk0OTUwOS4xNzM4MzM3MDIx*_ga*MTEwOTQzNzc1OC4xNjkxODM5NDAy*_ga_53S71ZZG1X*MTc0NDU1Nzk4My4yNDguMS4xNzQ0NTU5MTczLjAuMC4zNzM5MzIwODc.#typecasting-in-java)
7. [Best Practices for Course and Problem-Solving](https://companion.scaler.com/meeting/class/858012/notes?_gl=1*1aitnjq*_gcl_gb*R0NMLjE3NDE3MTE0NjQuQ2xrS0NBandtN3EtQmhCbEVra0F4aEFCZU5kUTB0NmFPckdKNlJ2bEQtSWVVLVRaZGo0TmdPTXE5enFGWElNUEMyVFRhMDZoUmFSbWtheE5iMHowcjRHOVRCQlRULUh4cEVTbkZCMUM0eDMzZGQ0enA5Qm5HZ0p5eGcuc0dDakNJLTZ4OGdDRUx5XzA2RUQuQlpJRUNNU2EwWmtaRUx5XzA2RUQ.*_gcl_au*MTIyMjk0OTUwOS4xNzM4MzM3MDIxLjE3ODE5NzQ1NTguMTc0NDI0Mzc1Mi4xNzQ0MjQzNzUx*FPAU*MTIyMjk0OTUwOS4xNzM4MzM3MDIx*_ga*MTEwOTQzNzc1OC4xNjkxODM5NDAy*_ga_53S71ZZG1X*MTc0NDU1Nzk4My4yNDguMS4xNzQ0NTU5MTczLjAuMC4zNzM5MzIwODc.#best-practices-for-course-and-problem-solving)

**Introduction to Output in Java**

In Java, the execution of a program starts from the main function. Within the main function, you can print to the console using the System.out.print or System.out.println methods.

public class Main {

public static void main(String[] args) {

System.out.print("Hello, World!"); // prints without a new line

System.out.println("Hello, World!"); // prints with a new line

}

}

* System.out.print: Prints the text in the parentheses on the same line.
* System.out.println: Prints the text in the parentheses and then moves to a new line .

**Printing Strings and Numbers**

**Printing Strings**

To print a string, you place the string inside double quotes within the print or println methods.

System.out.println("Hello, World!"); // Prints: Hello, World!

**Printing Numbers**

Numbers can be printed directly without quotes.

System.out.println(12345); // Prints: 12345

**Combined Example**

When switching between types, it's crucial to understand how they interact.

System.out.print("My age is " + 25); // Prints: My age is 25

* If a string is concatenated with a number, the number becomes part of the string called *String Concatenation* .

**String Concatenation**

Concatenation is combining multiple strings into one by using the + operator.

**Basic Concatenation**

System.out.println("Hello" + " World!"); // Prints: Hello World!

**Concatenation of Strings and Numbers**

The type of the first operand determines the operation:

System.out.println("My age is " + 25 + 10); // Prints: My age is 2510

System.out.println(25 + 10 + " is my age"); // Prints: 35 is my age

* If the first operand is a string, the entire expression is treated as a string .

**Input Handling in Java**

To take input from the user, Java uses the Scanner class.

**Usage of Scanner**

1. Import the Scanner class: import java.util.Scanner;
2. Create a Scanner object: Scanner sc = new Scanner(System.in);
3. Use methods like nextInt(), nextFloat(), nextLine() to read different types of input.

**Example**

Scanner sc = new Scanner(System.in);

// Reading an integer

int num = sc.nextInt();

System.out.println("The number is: " + num);

Multiple input types in a single program:

Scanner sc = new Scanner(System.in);

int intValue = sc.nextInt();

long longValue = sc.nextLong();

double doubleValue = sc.nextDouble();

String stringValue = sc.next(); // for a single word

System.out.println(intValue + ", " + longValue + ", " + doubleValue + ", " + stringValue);

``】

For whole lines, `nextLine()` can be used:

```java

String fullInput = sc.nextLine(); // Reads an entire line

System.out.println(fullInput);

`` .

## Data Types: An Overview

Java supports several data types for storing numbers, characters, and boolean values. Key data types include:

### Integer Data Types

1. \*\*byte\*\*: 1 byte, stores whole numbers from -128 to 127.

2. \*\*short\*\*: 2 bytes, stores whole numbers from -32,768 to 32,767.

3. \*\*int\*\*: 4 bytes, stores whole numbers from -2^31 to 2^31-1.

4. \*\*long\*\*: 8 bytes, stores whole numbers from -2^63 to 2^63-1.

### Floating Point Data Types

1. \*\*float\*\*: 4 bytes, stores fractional numbers up to 7 decimal digits.

2. \*\*double\*\*: 8 bytes, stores fractional numbers up to 16 decimal digits .

### Other Data Types

1. \*\*char\*\*: 2 bytes, stores a single 16-bit Unicode character.

2. \*\*boolean\*\*: 1 bit, stores `true` or `false`.

## Typecasting in Java

Typecasting is converting one data type into another. Types are casted to avoid data loss and to ensure the correct type of data handling.

### Implicit Typecasting (Widening)

```java

int i = 100;

long l = i; // Automatically done, no explicit cast needed

**Explicit Typecasting (Narrowing)**

When converting from a larger to a smaller type, the cast must be explicit to avoid data loss.

double d = 100.04;

int i = (int) d; // d is explicitly casted to int, floating part lost

System.out.println(i); // Prints: 100

Special care must be taken when narrowing to avoid unexpected behavior.

**Best Practices for Course and Problem-Solving**

1. **Attend Classes**: Regular attendance ensures you stay up to date with the material .
2. **Daily Practice**: Work on problems daily or the very next day to retain new concepts better.
3. **Assignments and Homework**: Focus on assignment problems first, then challenge yourself with additional problems .
4. **Engage in Public Forums**: Ask questions publicly to help yourself and others .

Remember to keep pace with assignments and use discussion forums proactively to clarify doubts. Consistency and practice are key to mastering Java programming.

Integer To Long Conversion:

It’s a implicit Type casting where an integer is converted to a long without needing explicit conversion.

Long to Integer Conversion:

Requires explicit typecasting since data might be lost if the long value exceeds the integer range.

Float to Integer conversion:

Converting a float to an integer truncates the functional part causing some information loss.

String Concatenation:

Combining multiple strings into one using the plus (+) operator.

Print VS Println Vs Printf:

Print prints text on the same line while println moves to the next line after printing and printf prints the formatted output.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function** | **Adds New Line?** | **Supports Formatting** | **Typical Use** | **Example** |
| print | ❌ No | ❌ No | Simple output on the same line | System.out.print("Hello"); System.out.print("World");  Output: HelloWorld |
| println | ✅ Yes | ❌ No | Output with line breaks | System.out.println("Hello"); System.out.println("World");  Output: Hello World |
| printf | ❌ No | ✅ Yes | Formatted output | double pi = 3.14159; System.out.printf("Pi = %.2f", pi);  Output: Pi = 3.14 |

Byte Data Type:

A data type that uses 1 byte of memory and can store values from -128 to 127.

Short Data Type:

Uses 2 bytes of memory and can store values from -32768 to 32767.

Integer Data Type:

Uses 4 bytes of memory and can store values from -2147483648 to 2147483647.

Long Data Type:

Uses 8 bytes of memory and can store very large integer values.

Character Data Type:

A primitive data type that can store a single 16-bit Unicode character.

Boolean Data Type:

Stores only two possible values: true and false.

Default Data Type:

Whole numbers are by default int, and fractional numbers are by default double.

Typecasting:

Converting one data type to another.

Implicit Typecasting:

Automatic conversion of smaller data type to larger data type by the compiler.

Explicit Typecasting:

Manual conversion of one data type to another using cast operator.

Garbage Value:

Incorrect value stored due to typecasting beyond valid range.

Scanner class:

Class used to take input from user in java.

String Concatenation:

Combining multiple strings into one using the ‘+’ operator.

Integer to Long Typecasting:

Automatically converting an integer to a long type.

Long to Integer Typecasting:

Converting a long to an integer manually; may result in loss of information.

Main Method:

Starting point of execution in a java program.

Primitive Data Types:

Predefine data types in java like int, long, float, double etc.

Non-Primitive Data Types:

User-defined data types in java like arrays, string and classes.

System.out.print:

Method to print output without a newline in java.

Java program start with main method

Public static void main (){

System.out.print(“----");

}

=================================================================================

public class Main {

public static void main (String[] args){

System.out.print(10);

}

}

-To print number no need double quote

-Java is a case sensitive language

public class Main {

public static void main (String[] args){

System.out.print(“5 \* 10”);

}

}

Output: 5\*10 🡪 as its in double quote it is consider as a string not arithmetic operation

public class Main {

public static void main (String[] args){

System.out.print(5\*10);

System.out.print(10/5);

System.out.print(5+10);

System.out.print(10-5);

}

}

Output: 502155 🡪 the output is printing in same line for all operations

public class Main {

public static void main (String[] args){

System.out.println(5\*10);

System.out. println(10/5);

System.out. println(5+10);

System.out. println(10-5);

}

}

Output:

50

2

15

5

🡪 the output is printing in next line for all operations as we are using println instead of print

All class name and package names are start with capital letters

Single line comments:

// This is comment

Multi line comments

/\*

This

Is

Multi

Line

Comments

\*/

Ctrl+/ : commenting and uncommenting

System.out.print(“Awesome” +”Shubham”) 🡪 concatenating two string

System.out.print(“My age is ” + 30) 🡪 output: My age is 30 🡪 String + int is concatenation

System.out.print(7+1+”156”) 🡪 output: 8156 🡪 **Operations happened left to right**.

System.out.print(”156” +7+1) 🡪 output: 15671 🡪 left to right, ”156” +7+1 > “1567”+1 > 15671

**DATA TYPES:**

1. Primitive
2. Numeric
3. Integer

* Byte
* Short
* Int
* long

1. Floating point

* Double
* Float

1. Non-Numeric

* Char
* Boolean

1. Non-Primitive **(Made by primitive data types)**

* String
* Arrays
* User define classes

**BYTE:**

1 byte in memory

1 byte= 8 bits

Range: -2n-1 to 2n-1-1 where n = bits

-28-1 to 28-1-1

-27 to 27-1

-128 to 127

**SHORT:**

2 bytes in memory

16 bits

-32768 to 32767

**INT:**

4 bytes in memory

32 bits

-2147483648 to 2147483647

**LONG:**

8 bytes in memory

64 bits

------------------------------------------------------

**210 ≈ 103**

32 bits range is

-232-1 to 232-1-1

-231 to 231-1

231 🡪 2 \* 2 30 🡪 2 \* (210)3 🡪2 \* (103)3 🡪 2 \* 109 ≈ 109

So, -231 to 231-1 will be approx. -109 to 10 9

------------------------------------------------------

**FLOAT:**

Store decimal number up to 6 to 7 points.

**DOUBLE:**

Store decimal number up to 15 points.

In java every integer considers as int and every floating point consider as double by default.

long d = 123456789l;

float f=1234.34453f;

To define long use “l” after int number and for float use “f” after floating point number

String s= “This is String”

**TYPE CASTING:**

Converting one data type to another

Int i=1000;

Long l=i; 🡪 implicit typecasting

Print I 🡪 output: 1000

Long l=1000l;

Int i=l;

Print i 🡪 throw error possible lossy conversion

Long l=1000l;

Int i=(int)l; 🡪 forcing to convert , explicit typecasting

Print i 🡪 output: 1000

Long l=10000000000000l;

Int i=(int)l; 🡪 forcing to convert , explicit typecasting

Print i 🡪 output: some garbage value as its exceeding/overflow range of int and we forcing to convert

DEFAULT VALUES OF DATATYPE IN JAVA:

**🧮 Primitive Data Types**

| **Data Type** | **Default Value** |
| --- | --- |
| byte | 0 |
| short | 0 |
| int | 0 |
| long | 0L |
| float | 0.0f |
| double | 0.0d |
| char | '\u0000' (null character) |
| boolean | false |

**📦 Reference Types**

| **Data Type** | **Default Value** |
| --- | --- |
| Any Object | null |
| Arrays | null (unless initialized) |
| String | null |

**✅ Java Number Literals Summary:**

| **Format** | **Prefix** | **Base** | **Example** |
| --- | --- | --- | --- |
| Decimal | none | 10 | 123 |
| Octal | 0 | 8 | 012 → 10 |
| Hexadecimal | 0x or 0X | 16 | 0xA → 10 |
| Binary (since Java 7) | 0b or 0B | 2 | 0b1010 → 10 |

**🔁 What is Automatic Type Promotion?**

When operands of different types are used in an expression, **Java automatically promotes the smaller data type to a larger one** to avoid loss of information.

**🔼 Promotion Rules (for primitives):**

1. byte, short, and char → promoted to int when used in expressions.
2. If one operand is long, the result is promoted to long.
3. If one is float, the result is promoted to float.
4. If one is double, the result is promoted to double.

**🧮 Example:**

byte a = 10;

byte b = 20;

// byte c = a + b; // ❌ Compile error: result is int

int c = a + b; // ✅ Correct: promoted to int

Why? Because a + b is promoted to int, even though both are byte.

**🧠 Also Happens in:**

* **Method overloading:** when you pass a smaller type, and Java promotes it to match a

method parameter.

void show(int x) {

System.out.println("int version");

}

show('A'); // char is promoted to int → prints "int version"

**TAKING INPUT FROM USER:**

import java.util.\*;

Scanner sc=new Scanner(System.in);

Int i=sc.nextInt();

nextInt()

nextLong()

nextFloat()

nextDouble()

next() 🡪 It takes one word at a time

nextLine() 🡪 to take entire line as input at a time