

* LECTURE 4:

- if you are creating a class - first letter should be capital.
- in Java, where your program starts is the main function.

→ `public static void main (String[] args) { }`

- `javac Main.java` → compiles the program.

`public static void main (String[] args) { }`

- ① public means that this class can be accessed from anywhere.

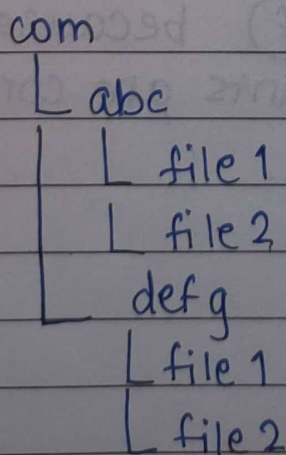
- ② main function is the entry point of the JAVA program.

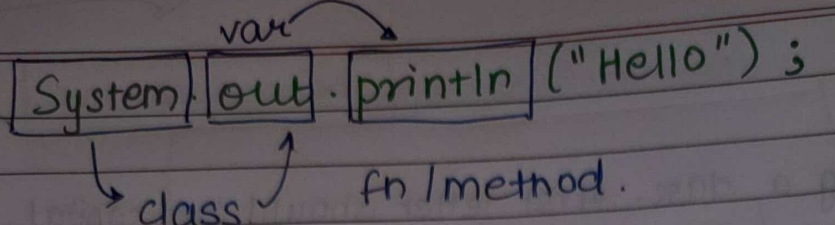
- ③ static → we want to run the main function without creating Object of class.

- ④ void → returns nothing.

- ⑤ whatever arguments you are passing at the command line, are saved in the variable args.

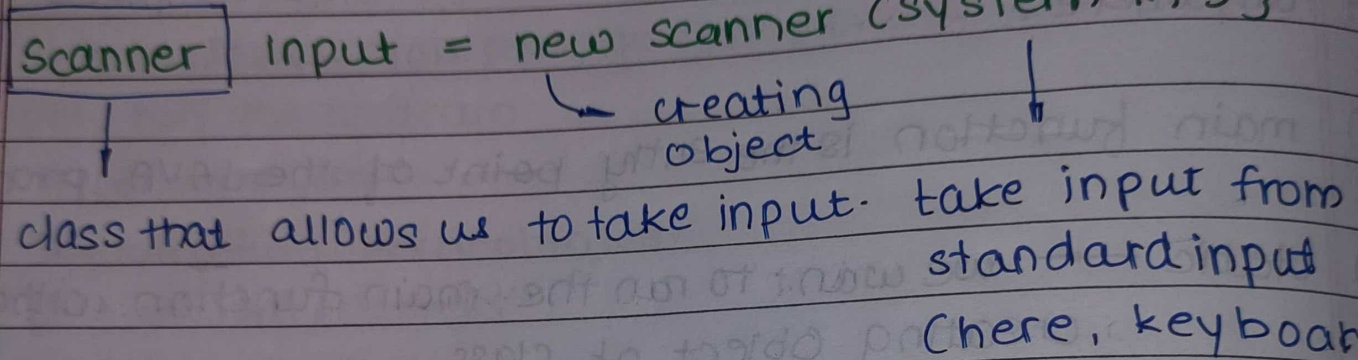
package com.abc OR package com.defg.



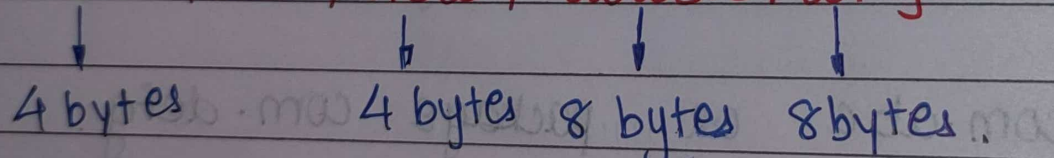
- `System.out.println("Hello");` ; This means print the output on the standard output stream (Here, terminal)


`println` ——— adds a new line

`print` ——— does not add a new line

- `Scanner input = new Scanner(System.in);`


* **Primitives.** —→ any datatype you can't break any further.

`int`, `char`, `float`, `double`, `long`, `boolean`.


for `float` —→ we add `(f)` because by default decimal points are considered as `double`.

* Literals and Identifiers,

int a = 10;

↓
↓
identifier literal

Literals:- Java literals are syntactic representations of boolean, character, numerical or string data.

Identifiers:- Identifiers are the names of variables, methods, classes, packages and interfaces.

int a = 234_000_000;

↳ the value of a will be 234000000, underscore will be ignored.

⇒

• nextInt()

↳ takes integer input

• next()

↳ takes one word input

• nextLine()

↳ takes the entire line as input.

⇒

floating point → rounds it off → thus it's not accurate.

• Type Casting and Type Conversion

⇒ Widening or Automatic Type Conversion

→ Two datatypes are automatically converted.

→ This takes place in two conditions:-

⊙ We assign the value of smaller datatype to bigger datatype

⊙ Two datatypes must be compatible.

byte → short → int → long → float → double

⇒ Narrowing or Explicit Type Conversion

→ This happens when we want to assign a value of larger data type to a smaller data type.

double → float → long → int → short → byte

eg: - `int num = (int) (67.56f);`

↳ Output = 67

• Automatic Type Promotion In Expressions

→ while evaluating expressions, the intermediate value may exceed the range of operands & hence the expression value will be promoted.

→ Some conditions of type promotion are:-

1. Java automatically promotes each byte, short, char to int when evaluating an expression.

2. Long, float or double the whole expression is promoted to long, float or double.

eg:- After solving expressions:-

$$\begin{array}{ccccccc} (f * b) & + & (i / c) & - & (d * s); & & \\ \downarrow & & \downarrow & & \downarrow & & \\ \text{float} & + & \text{int} & - & \text{double} & = & \text{double} \end{array}$$

converted to biggest one

• Automatic Type Promotion in Expressions.

while evaluating expressions, the intermediate value may exceed the range of operands & hence the expression value will be promoted.

Some rules:-

- ① Java automatically promotes each byte, short, char to int while evaluating an expression.
- ② If any one of the operand is long, float or double, whole expression is promoted to long, float or double.

eg:-

$$\begin{array}{ccccccc} (f * b) & + & (i / c) & - & (d * s); & = & \text{double} \\ \downarrow & & \downarrow & & \downarrow & & \\ \text{float} & + & \text{int} & - & \text{double} & & \end{array}$$

* Explicit type casting in expression.

If we want to store large value into small data type

eg:- byte b = 50

b = (byte) (b * 2) ——— type casting into byte.