* objects in classes are accessed by dot operators.
* Running the code on CMD - javac filename.java
  java filename.java

→ Additional Notes from Module 01 -

* java dosn't use explicit pointers and operator overloading, it takes care of memory management and provides automatic gaubage collection (collects unused objects).
* java is portable → platform-independent.
* java code $\xrightarrow{HLL}$ compiler → bytecode $\xrightarrow[JVM]{}$ interpreter $\xrightarrow{LLL}$ output
* java is robust → used in bothe consumer and mission-critical applications.
* Java was developed in 1991.
* java is Architecture Neutral → "write once, run anywhere, anytime, forever."
* we can start and operate the computer without the cache memory.
* we cannot run the java program without main() method.

Module 02 - Variables and Datatypes -

* java is statically typed lang. → faster.
* basic datatypes - int, float, string...
* variable name should not being with number and must not contain any special symbol other than _, $ and must not be a keyword; can be camelCase and for constants, all caps → MAX_LIMIT = 100.

↓

→ we use final keyword, eg: final int MAX_AGE = 75;

* datatypes - ① primitive (boolean, byte, short, int, long, char, double...)

② non-primitive (arrays, stacks, strings, etc...)

* byte → 8 bit; short → 16 bit; int → 32 bit; long → 64 bit;; big int → 128 bit; float, double → 32 bits; char → 16 bit (for ranges of datatypes (DT) google them).

* any user-defined class-type can be a non-primitive DT.

* int age = 25 ; → semicolon.
  DT variable assignment literal

* variables of non-primitive DT are created using the new operator, and accessed using dot operator.

* in java, non-primitives are always passed by references. → (secure)

* in java, non-primitives are stored in heap memory (for dynamic memory allocation) and primitives are stored in stack memory (used for function calls).

* non-assigned non-primitives get a default value.

↳ Swapping two values -

* method / logic for the approach -

① temp variable declaration ① (conventional)
② addition / subtraction logic
③ multiplication / division logic
④ single line (using in-built function) → currently not available in recent version of Java.
⑤ using XOR operation ②.

X	Y	X ⊕ Y
1	1	→ 0
0	1	→ 1
1	0	→ 1
0	0	→ 0

→ XOR Truth Table

* wrapper class → non-primitive class for primitive DT.

↓

need → to need all as objects and used in collections & generics.

* Autoboxing ⇒ int $x_1 = 10$;

& Integer $x_2 = x_1$; // Auto-boxing

Autounboxing ⇒ int $x_3 = x_2$; // Auto-unboxing

* widening / implicit conversion -

byte → short → int → ~~float~~ long → float → double

char.

↳ eg: int x = 100;
long y = x;
float z = y;

o/p → 100
100
100.0

* narrowing / explicity convesion -

↳ eg: double d = 65.4;
int i = (int) d;
char c = (char) i;
S.o.p(i);
S.o.p(c);

o/p → 65
A
↳ Ascii value of 65.

* Autoboxing / Auto-unboxing doesn't work for small values due to caching.

* by default, the floating number is treated as double DT.

⇒ Additional Notes from Module 02 -

* there are 3 types of variables - ① local variables
② instance variables
③ static variables

① → scope is within a block/method; initialization is mandatory.

② → initialization is done by constructor; uses access specifiers; scope is within a class.

③ → declared using static keyword.