13/09/21

First JAVA Program - Input/ output, Debugging and Datatype

File name : Demo.java

class name: Demo

↳ It is good practice to use initial character as capital (you can use small too)

public → This ~~function~~ keyword means, it is used to access the class from anywhere

function → Collection of code, that we can use again and again. Function are also known as methods.

void → This void keyword specifies that a method should not have a return value.

String([]) **args** → means an array of sequence of character ("string") that are passed to main function.

↓
array

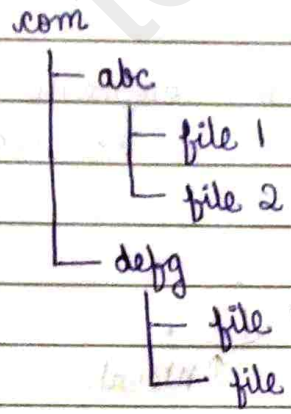
- after compiling class file is always saved in current location where you are in.
- if you want to change the location, use -d (destination) while compiling and specify the path.

java -d <path> Demo.java

__/__/____

- class name and file name should be same, but if we don't want to make class name as file name then it should not be private.

- Package com.abc OR Package com.defg.



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

  System → class; out → var; println → fn/method

- Scanner sc = new Scanner(System.in);

  Scanner → class that allow to take input; new → creating object; System.in → take input from standard input (here, keyboard)

Primitive → means datatype that cannot be broken further

```
int     rollno  = 64;          → 4 byte
char    letter  = 'x';         →
float   marks   = 98.67f;      → 4 bytes
double  largeDecimal = 451321086.4113;  → 8 byte
long    largeIntegers = 1234130125L;    → 8 byte
boolean check   = True;
```

__/__/____

- String is written in double quotes, while char are in single quotes.

- Integer ⟶ wrapper class ⟶ provide additional functionalities
  ↓
  convert primitive datatype to object

- Comment → the lines that we comment are ignored by JAVA and will not be executed

```
int a = 10;
    ↑ identifier
          ↑ literal
```

Type casting and Type conversion:

- **Widening or Automatic Type conversion**
  → the datatype are automatically converted
  → this happen when we assign value of smaller datatype to bigger datatype and two datatype must be compatible.

byte → short → int → long → float → double.

Eg
```
int i = 100;        100
long l = i;         100
float f = l;        100.0
```

- **Narrowing or Explicit Conversion**

→ This happens when we want to assign a value of larger data type to smaller datatype. We perform explicit type casting or narrowing.

double → float → long → int → short → byte

Eg
```
double d = 100.04 ;     100.04
long l = (long)(d) ;    100
int i = (int)(l)        100
```

- **Automatic Type Promotion in Expression**

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

→ some condition of type promotion are :-

1. JAVA automatically promote 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 expression :

```
(f * b) + (i / c) - (d * s);
```

we get ↓

float + int - double = double.

(convert to biggest one)

- **Explicit type casting in Expression**

If we want to store larger value to small datatype.

```
byte b = 50;
b = (byte) (b * 2);   ← type casting int to byte.
```