3/8/21

First Java Program - Input/Output, Debugging & Datatypes

File name: **Demo.java** Class Name: **Demo**

Its good practice to use initial character as capital (you can use small also)

public → this keyword means, it is used so that we can access the class from anywhere.

functions → Collection of code, that we can use again & again. Functions are also known as methods.

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

String[] args → means an array of sequence of characters ("strings") that are passed to the 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** (destinato option while compiling and specify the path.

javac -d <path> Demo.java

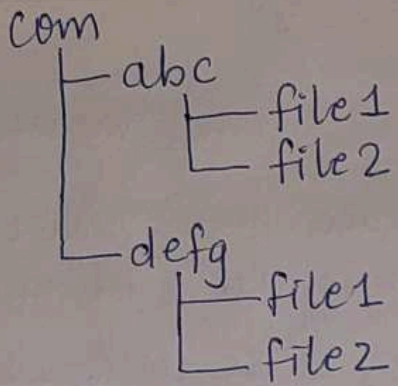
- **echo $PATH** → every command looks for this location before executing.

[environment variables]

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

for eg → **class Divide**

- package com.abc OR package com.defg



- `System.out.println("Hello");` → this means print the output on Standard output stream (here, terminal)
  - `System` → class
  - `out` → var
  - `println` → fn/method

println → adds new line
print → does not add new line.

- `Scanner input = new Scanner(System.in);`
  - `Scanner` → class that allows us to take input
  - `new` → creating object
  - `System.in` → take input from standard input (here, keyboard)

Primitive → means any data type that cannot be broke further.
integer, character etc. are primitive datatype.

⇒
```
int rollno = 64;  → 4 bytes
char letter = 'r';
float marks = 98.67f;  → 4 bytes
double largeDecimalNumbers = 456789.12345;  → 8 bytes
long largeIntegers = 12345678l0L;  → 8 bytes
boolean check = true;
```

- String is written in double quotes whereas while specifying char we write it in single quotes.
- All decimal values that we use are by default of double datatype, therefore if we want to store in float we have to use "f", same for int & long.

```
float marks = 7.2f
(by default) double largeDecimalNumbers = 456789101.12345

int rollno = 64; (by default)
long LargeInteger = 1234567891011L;
```

- Integer → Wrapper class → provides additional functionalities
  - converts primitive datatype to object.
- Comment → the lines that we comment are ignored by Java and will not be executed.
  Comment in Java → //

→ `int a = 10` → literal

a → identifier

- Literals : Java literals are syntactic representations of boolean, character, numeric or string data.
  here, 10 is an integer literal.
- Identifiers : Identifiers are the names of variables, methods, classes, packages & interfaces.

- **int a = 234_000_000;**
  ↳ the value of a will be 234000000, underscore will be ignored.
- 564.12345678 $\xrightarrow{\text{rounds off}}$ 564.12345
  If we give float very big, than it rounds off the value which gives floating point error.

⟹ Type Casting & Type Conversion:

• Widening or Automatic Type Conversion:

→ Two datatypes are automatically converted.
→ This happens when we assign value of smaller datatype to bigger datatype & 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 we want to assign a value of larger data type to a smaller data type 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 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, ~~whole~~ float or double.

eg: After solving expression :

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

we get → float + int − double = double

converted to biggest one

• Explicit type casting in expressions :

→ If we want to store large value into small data type

eg:
```
byte b = 50;
b = (byte)(b * 2);
```
→ type casting int to byte.

• If-else syntax in Java

```
if (condition) {
  // block of code
} else {
  // block of code
}
```

• For loop syntax

```
for (statement1 ; statement2 ; statement3) {
  // code block
}
```