First Java Program - Input/Output, Debugging & Datatypes

File name : Main.java Class Name : Main

- It's good practice to use initial character as capital

1st program of Java.

```
public class Main {
    public static void main (String[] args) {
        System.out.println("Hello World!");
    }
}
```

Brief :-

- public - it used so that this can be access from anywhere
- class - a class is a named group of properties & function
- function :- It is a block of code, can be used again & again
  function are called methods
- public class name & file name should be same.
- ~~recom~~ recommended to use Camel Case (Capitalise The First Letter Of Each Word)

- main :- main function is a entry point of Java program it is a reserved function.

- static :- it used as to run the program without creating an object of public class.

- void :- return type of function
  i.e. it do not have any return value.

- String [] args - known as comand line arguments
  - means an collection of arrays of sequence of character ("strings") that are passed to the main function.

  array.

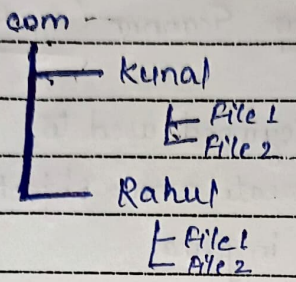
- After compiling , . class file is always saved in current location where the source file is located

- To change the location of byte code file (.class file) use following command while compiling
  javac -d <path> <filename>.

- Packages :- A package in java is like a folder that helps to organize yours java classes

  - To group related classes together
  - To avoid name conflicts
  - To keep code clean and organized
  - To create privacy or access control by access modifier.

Package com.kunal & package com.Rahul.



- echo $path (in Linux / Mac) & $evn:path (windows)
  - [Environment Variables] when we run javac or java or any command it look for this location before executing

- System.out.println ("Hello World"); → It means print the output on standard output
  - System :- built in class in java.lang package.
  - out :- a static field (variable) in system class
    it is a reference. its an object of printstream. stream: (i.e. here terminal)
  - System.out :- is the standard output stream.
  - println :- A method of the printstream class

  - println :- moves cursor to next line
  - print :- do not move cursor to next line

Date
Page

- **Input :-**

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

- `Scanner` — class that allows to take input.
- `new` — keyword used to create a new object in java
- `System.in` — take input from standard input stream (here keyboard)

- **Primitive :-** means any data types that connot be broken into simpler data types.

eg.:- integer, character, etc.

```
int rollno = 64;          → 4 byte
char letter = 'r';
float marks = 98.82f;     → 4 byte
double largeDecimalNumber = 456789.12345 → 8byte
long largeIntegers = 12345698101L;  8 byte
boolean check = true;
```

- string written in double quote
  char written in single quote

- A decimal value are by default of double datatype therefore if we want to store in float we use f

- All numbers/integers are by defaut of int datatype therefore for long we have to use l.

- **Integers :-** It is a wrapper class in Java.
  - Wrapper class is used to wrap primitive datatypes into an object.
  - provides additional functionalities.

- **Comment :-** The lines that are commented are ignored by java and will not be executed

- `int a = 10` ← literals
  ↑ identifiers

literals :- Java literals are syntactic representation of boolean, character, number or string data.
Here, 10 is an integral literal.

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

Int Input :- nextInt() is function used to take int as input.

Syantax :-
```
Scanner input = new Scanner (System.in);
int rollno = input.nextInt();
```

* `int a = 234_000_000;`
  ↳ value of a will be 234000000.
  underscore will be ignored

- **float Input :-** nextFloat() is a function used to ~~make~~ take float as a int.

Syntax :-

```
Scanner input = new Scanner(System.in);
float marks = input.nextFloat();
```

- 564.12345678 ⟶ round of to 564.12345

If we give float very big, than it rounds off to the value which gives floating point error.

- **String Input :-** Two ways to take string input.

1. Using next() Method :- It will take one word input till a space occurs.

Syntax :-

```
String s1 = input.next();
```

Input :- Hey Abhi
Output :- Hey

2. Using nextline() Method :- It will take ~~one~~ line as input. (including space)

Syntax :-

```
String s2 = input.nextline();
```

Input :- Hey Abhi
Output :- Hey Abhi

Type Conversion +

- **Automatic Type Conversion :-**

→ Two datatypes are automatically converted.

→ This happens when we assign value of smaller datatypes to bigger datatypes & two datatypes must be compatible.

byte → short → int → long → float → double

eg :- int i = 100 ; ⟶ 100
long l = i ; ⟶ 100
float f = l ; ⟶ 100.0

- **Narrow or Explicit Conversion :-**

→ This happens we want to assign a value of large data types 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

Date
Page

- Automatic type promotion in Expressions :-

→ while evaluting expression, the intermediati value may exceed the range of operands & hence the expression value will be promoted

→ Condition 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.

- 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 int to byte.

- **if-else :-** used to execute code conditionally
  if a condition is true execute a code block
  else execute anothe code block

Syntax :-

```
if (condition) {
    code block;
}
else {
    code block;
}
```

- **if-else if-else :-**

Syntax :-

```
if (condition) {
    code block;
}
else if (condition) {
    code block;
}
else ~~(condition)~~ {
    code block;
}
```

Loop :- A loop is a control structure that allows to repeate a block of code multiple times as long as ~~condit~~ given a condition is true.

1.) for loop :- used when no. of iderations are known.

Syantax :-

```
for (int i (initialization); i<=5 (condition),
                                i++ (update))
  {
    //(code block)
  }
```

⇒ While loop :- used when no. of Iterations is unknown but condition is provided.

Syantax :-

```
while (condition) {
  // code
}
```

⇒ Do-while loop :- Used when loop is required to run atleast once

syantax :-

```
do {
  // code
} while (condition);
```