

Basic codes in java:  
Hello world, Command line argument.  
Taking user input, creating functions

Unit-1

## ① Hello World in java

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

- In java, all codes must be inside a class.  
So main fun is created inside demo class.
- Declaration of main fun is \*\*

public static void main (String args[])

- 1) public → it can be accessed outside class.
- 2) static → main is static, so that it can be called w/o creating any obj of class.

main fun is called as follows.

demo.main()

When OS runs the pgm, it calls the main()

- for non-static fun, obj needs to be created  
eg `demo obj;` // create a ~~demo~~ obj of demo  
`obj.main();` & call the `main()`.

- 3) void  $\rightarrow$  return type of main must be void.
- 4) String args[]  $\rightarrow$  This an array of strings.
  - main fun  $\nmid$  must have this parameter.
  - Its purpose is to take command line arguments from user.

• println fun in java.

Its use is :-

`System.out.println("Hello");`

- 1) System :- is an inbuilt class.
  - It is inside `java.lang` package.
  - It is imported automatically.
  - It ~~is~~ provides fun to perform input/output.
- 2) Out :- ~~out~~ is an obj inside ~~the~~ System class.
  - It is an obj of `PrintStream` class.
  - It is a static member, so it can be accessed by using System class name.

3) println :- it is a fun to print  
It also prints new line character.

### Different ways to print :

```
String s1 = "Hello";  
"      s2 = "World";  
int a = 10, b = 20;
```

• println → System.out.println ( . )

• println ("Hello World");

println (s1 + s2);

println (s1 + "World");

println (s1 + a); // Hello10

it is converted to string.

To print multiple int, use mul times

println(a);

println(b);

println(c);

// println(a+b+c); will print the sum.

• println (); // prints a new line

• Use print() to print w/o new line

eg System.out.print("Hello");  
" " " ("World"); // Hello World.



## How to run a pgm in java

Step 1:- Compile using `javac filename.java`

Step 2:- run using `java classname`

eg open terminal

Save code to desktop

go to desktop: `cd desktop`

`javac hello.java`  
filename.

`java demo`  
class name.

---

## Command line Arguments

- CLA is a way to take input from user when the pgm is run.

eg i/p can be provided when running

> `java demo Rahul 10`  
cmd to run pgm      input → (CLA)

- 2 i/p r provided - these r called CLA.

- These i/p (CLA) r available to the user in the array passed inside main fun.

eg class demo {  
 public static void main (String args[]) {  
 // CLA r stored in this array  
 System.out.println (args[0]); // Rahul  
 " " " " (args[1]); // 10  
 }  
}

- This pgm is compiled & run.  
 > javac hello.java.  
 > java demo Rahul 10  
 These r passed to the args [] inside main fun.  
 Then the o/p is printed.

### Taking i/p from user

- Scanner class is used to take i/p from the user. It is inside the java.util package.  
 It needs to be imported using "import java.util.Scanner".



Note \*\*

• Special precaution needs to be taken when taking input in the order int, then String

• In this case, the `nextLine()` fun reads the stray '\n' in the buffer.

• To remove this '\n', ~~the~~ `nextLine()` fun needs to be called 2 times.

eg `int a;`

`String s;`

// When int is read by a string, use `nextLine()` fun 2 times.

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

`a = scobj.nextInt();`

`scobj.nextLine();`

`s = scobj.nextLine();` // to remove '\n' after from the buffer.

`s = scobj.nextLine();`

`scobj.nextLine();`



## Creating fun in java

- Any fun can be created/defined b4 or after main fun in java.
- eg add()
- This must be static, bcoz it is called inside the main()
- Main() itself is static, so it can call other static fun. Main() can't call other non static fun.

eg WAP to create add fun:...

Class demo {

main() → 

```
public static void main (String args[]) {  
    int a = 10; b = 20;  
    add(a, b);  
}
```

add() → 

```
public static void add (int x, int y) {  
    Sopln(a+b);  
}
```

must be static

}



eg WAP to print sq of a no. :-

```
class demo {  
    public static void main(String args[]) {  
        int a = 10;  
        sq(a);  
    }  
    public static void sq(int x) {  
        System.out.println(x * x);  
    }  
}
```

Call by value &  
Call by reference

Q\*\* Are fun called by value or  
" " ref in java?

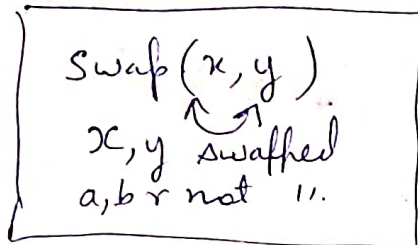
Ans, 1) When primitive data types (int, char, float etc)  
r passed, then fun is called by value.

2) When obj, arrays, strings r passed to fun  
then fun is called by ref.

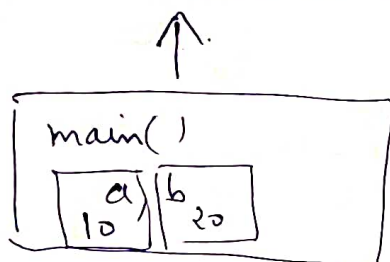
3) bcoz obj, arrays, strings r implemented  
variables r reference variables.

eg 1) A fun to swap 2 int var won't work.  
bcz int r passed by value.

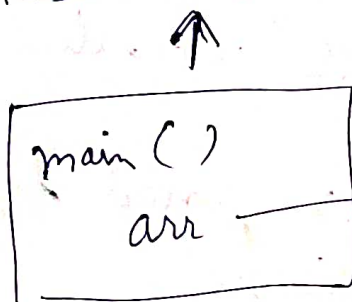
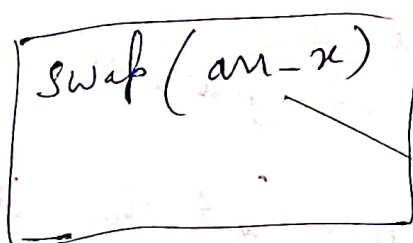
2) but a fun to swap 2 elements of an array.  
will work bcz arr r passed by ref.



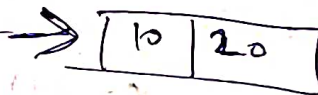
// A copy of ~~a, b~~ <sup>a, b</sup> is passed.  
to x, y in swap  
• x & y swapped.  
a, & b r not swapped.



### 1- Swap of int



both point to same arr.



### 2- Swap of arr el

// when arr is passed to a fun, both arr & arr-x  
point to same array.

// So arr el swapped in swap fun will also be  
swapped for arr el in main fun.

ex. Pgm to demonstrate that arr is passed by ref & int is passed by value.

- Create swapint to swap 2 int.
- " swap arr to " el of an arr.

C D { PS VM (..) }

means ↑

Class demo {  
public static void main (String args[]) {

int a = 10, b = 20;

int ar[] = { 10, 20 };

swapint(a, b); // fun to swap int

swap ar(ar); // " " " ar.

System.out.println(a); // 10. \*\*\* a, b are not swapped.

System.out.println(b); // 20. swapint fun doesn't work.

System.out.println(ar[0]); // 20. \*\*\* ar elements are swapped.

System.out.println(ar[1]); // 10. swap ar fun works  
bcz it is called by ref.

}

public static void swapint(int x, int y) {

int t = x;

x = y;

y = t;

}



```

public static void swapar(int ar[]) {
    int t = ar[0];
    ar[0] = ar[1];
    ar[ar[1] = art];
    // swap ar[0] & ar[1];
}

```

} // demo class ends.

Q) How to swap 2 int var?

A swap w/o using fun.

eg C D { P S V M ( ) }

```
int a = 10; b = 20;
```

```
int t = a;
```

```
a = b;
```

```
b = t;
```

// swap a, b

w/o using fun.

```
System.out.println(a); // 20
```

```
System.out.println(b); // 10
```

} values swapped.

}

Q2) ~~Explain the following terms:~~ ~~1. Primary~~ ~~2. Secondary~~ ~~3. Tertiary~~ ~~4. Quaternary~~ ~~5. Quinary~~ ~~6. Sextary~~ ~~7. Septary~~ ~~8. Octary~~ ~~9. Nonary~~ ~~10. Decary~~ ~~11. Undecary~~ ~~12. Dodecary~~ ~~13. Tridecary~~ ~~14. Quodecary~~ ~~15. Quindecary~~ ~~16. Sextodecary~~ ~~17. Septodecary~~ ~~18. Octodecary~~ ~~19. Nonodecary~~ ~~20. Decodecary~~ ~~21. Undecodecary~~ ~~22. Dodecodecary~~ ~~23. Tridecodecary~~ ~~24. Quodecodecary~~ ~~25. Quindecary~~ ~~26. Sextodecary~~ ~~27. Septodecary~~ ~~28. Octodecary~~ ~~29. Nonodecary~~ ~~30. Decodecary~~ ~~31. Undecodecary~~ ~~32. Dodecodecary~~ ~~33. Tridecodecary~~ ~~34. Quodecodecary~~ ~~35. Quindecary~~ ~~36. Sextodecary~~ ~~37. Septodecary~~ ~~38. Octodecary~~ ~~39. Nonodecary~~ ~~40. Decodecary~~ ~~41. Undecodecary~~ ~~42. Dodecodecary~~ ~~43. Tridecodecary~~ ~~44. Quodecodecary~~ ~~45. Quindecary~~ ~~46. Sextodecary~~ ~~47. Septodecary~~ ~~48. Octodecary~~ ~~49. Nonodecary~~ ~~50. Decodecary~~ ~~51. Undecodecary~~ ~~52. Dodecodecary~~ ~~53. Tridecodecary~~ ~~54. Quodecodecary~~ ~~55. Quindecary~~ ~~56. Sextodecary~~ ~~57. Septodecary~~ ~~58. Octodecary~~ ~~59. Nonodecary~~ ~~60. Decodecary~~ ~~61. Undecodecary~~ ~~62. Dodecodecary~~ ~~63. Tridecodecary~~ ~~64. Quodecodecary~~ ~~65. Quindecary~~ ~~66. Sextodecary~~ ~~67. Septodecary~~ ~~68. Octodecary~~ ~~69. Nonodecary~~ ~~70. Decodecary~~ ~~71. Undecodecary~~ ~~72. Dodecodecary~~ ~~73. Tridecodecary~~ ~~74. Quodecodecary~~ ~~75. Quindecary~~ ~~76. Sextodecary~~ ~~77. Septodecary~~ ~~78. Octodecary~~ ~~79. Nonodecary~~ ~~80. Decodecary~~ ~~81. Undecodecary~~ ~~82. Dodecodecary~~ ~~83. Tridecodecary~~ ~~84. Quodecodecary~~ ~~85. Quindecary~~ ~~86. Sextodecary~~ ~~87. Septodecary~~ ~~88. Octodecary~~ ~~89. Nonodecary~~ ~~90. Decodecary~~ ~~91. Undecodecary~~ ~~92. Dodecodecary~~ ~~93. Tridecodecary~~ ~~94. Quodecodecary~~ ~~95. Quindecary~~ ~~96. Sextodecary~~ ~~97. Septodecary~~ ~~98. Octodecary~~ ~~99. Nonodecary~~ ~~100. Decodecary~~ ~~101. Undecodecary~~ ~~102. Dodecodecary~~ ~~103. Tridecodecary~~ ~~104. Quodecodecary~~ ~~105. Quindecary~~ ~~106. Sextodecary~~ ~~107. Septodecary~~ ~~108. Octodecary~~ ~~109. Nonodecary~~ ~~110. Decodecary~~ ~~111. Undecodecary~~ ~~112. Dodecodecary~~ ~~113. Tridecodecary~~ ~~114. Quodecodecary~~ ~~115. Quindecary~~ ~~116. Sextodecary~~ ~~117. Septodecary~~ ~~118. Octodecary~~ ~~119. Nonodecary~~ ~~120. Decodecary~~ ~~121. Undecodecary~~ ~~122. Dodecodecary~~ ~~123. Tridecodecary~~ ~~124. Quodecodecary~~ ~~125. Quindecary~~ ~~126. Sextodecary~~ ~~127. Septodecary~~ ~~128. Octodecary~~ ~~129. Nonodecary~~ ~~130. Decodecary~~ ~~131. Undecodecary~~ ~~132. Dodecodecary~~ ~~133. Tridecodecary~~ ~~134. Quodecodecary~~ ~~135. Quindecary~~ ~~136. Sextodecary~~ ~~137. Septodecary~~ ~~138. Octodecary~~ ~~139. Nonodecary~~ ~~140. Decodecary~~ ~~141. Undecodecary~~ ~~142. Dodecodecary~~ ~~143. Tridecodecary~~ ~~144. Quodecodecary~~ ~~145. Quindecary~~ ~~146. Sextodecary~~ ~~147. Septodecary~~ ~~148. Octodecary~~ ~~149. Nonodecary~~ ~~150. Decodecary~~ ~~151. Undecodecary~~ ~~152. Dodecodecary~~ ~~153. Tridecodecary~~ ~~154. Quodecodecary~~ ~~155. Quindecary~~ ~~156. Sextodecary~~ ~~157. Septodecary~~ ~~158. Octodecary~~ ~~159. Nonodecary~~ ~~160. Decodecary~~ ~~161. Undecodecary~~ ~~162. Dodecodecary~~ ~~163. Tridecodecary~~ ~~164. Quodecodecary~~ ~~165. Quindecary~~ ~~166. Sextodecary~~ ~~167. Septodecary~~ ~~168. Octodecary~~ ~~169. Nonodecary~~ ~~170. Decodecary~~ ~~171. Undecodecary~~ ~~172. Dodecodecary~~ ~~173. Tridecodecary~~ ~~174. Quodecodecary~~ ~~175. Quindecary~~ ~~176. Sextodecary~~ ~~177. Septodecary~~ ~~178. Octodecary~~ ~~179. Nonodecary~~ ~~180. Decodecary~~ ~~181. Undecodecary~~ ~~182. Dodecodecary~~ ~~183. Tridecodecary~~ ~~184. Quodecodecary~~ ~~185. Quindecary~~ ~~186. Sextodecary~~ ~~187. Septodecary~~ ~~188. Octodecary~~ ~~189. Nonodecary~~ ~~190. Decodecary~~ ~~191. Undecodecary~~ ~~192. Dodecodecary~~ ~~193. Tridecodecary~~ ~~194. Quodecodecary~~ ~~195. Quindecary~~ ~~196. Sextodecary~~ ~~197. Septodecary~~ ~~198. Octodecary~~ ~~199. Nonodecary~~ ~~200. Decodecary~~ ~~201. Undecodecary~~ ~~202. Dodecodecary~~ ~~203. Tridecodecary~~ ~~204. Quodecodecary~~ ~~205. Quindecary~~ ~~206. Sextodecary~~ ~~207. Septodecary~~ ~~208. Octodecary~~ ~~209. Nonodecary~~ ~~210. Decodecary~~ ~~211. Undecodecary~~ ~~212. Dodecodecary~~ ~~213. Tridecodecary~~ ~~214. Quodecodecary~~ ~~215. Quindecary~~ ~~216. Sextodecary~~ ~~217. Septodecary~~ ~~218. Octodecary~~ ~~219.~~

Q2) " " " "  
A int in java?

Q3) " " " " in jaila?

Q4) " " " " representation of a No. in java?

Ans-1 - No. Char r paired by value they won't be swapped.

Ans-2. Yes. No change is need to be made to var. So fun can be used.

3. No. It is passed by value. So it won't be incremented.

incremented.

4. Yes. No change needs to be made - so fun can be used.