GOOD MORNING EVERYONE

We will start @ 10:10 am

String :

↳ group of characters

↳ non-primitive datatype

↳ syntax :- String str = "----";

double quotes.

capital

→ String in Java is immutable

DSA _ Bootcamp

↳ User input of string

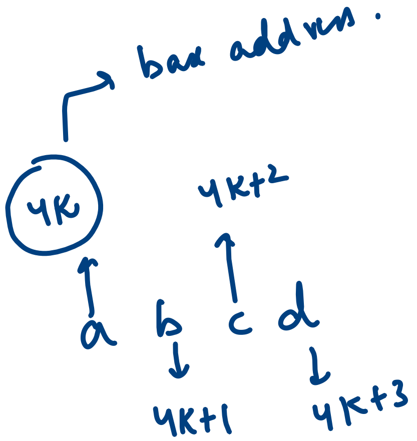
(a) String str = scn.next(); → DSA

(b) String str = scn.nextLine(); → DSA Bootcamp.

↳ String str = "abcde fghi jkl" → contn fashion

$\overset{0}{a}\,\overset{1}{b}\,\overset{2}{c}\,\underset{3}{d}\,\underset{4}{e}\ \underset{5}{\ }\ \overset{6}{f}\,\overset{7}{g}\,\overset{8}{h}\,\overset{9}{i}\ \underset{10}{\ }\ \overset{11}{j}\,\overset{12}{k}\,\overset{13}{l}$

4K



↳ Indexing in string (starts with zero).

↳ Length of string ⇒ no. of characters in a string

↳ length()

str. length(); ⇒ 14 (index+1)

(0 - 13) (0 -(n-1))

index ⇒ how far from the base address.

b ⇒ 4K+1 − 4K = $\boxed{1}$

a ⇒ 4K − 4K = 0

d = 4K+3 − 4K = 3

↳ charAt()

↳ str. charAt(6) ; → $\boxed{f}$

↳ In order to compare data of two strings → we use equals fn.

not == ✗ in Java.

Ex:- String str1 = "abc";

4k

String str2 = "def";

4k+10

str1. equals (str2); → O/P is a boolean value.

→ true

→ false.

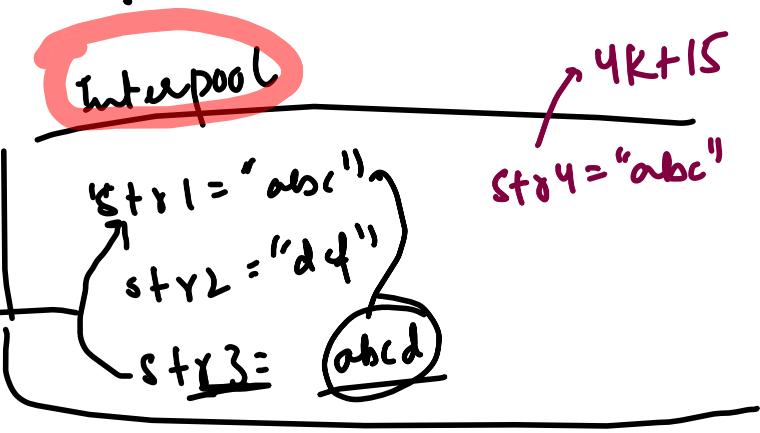
↳ if data is same → true

↳ otherwise O/P → false.

↳ == → in string compare address.

4K+12

String str3 = "abc"; ✗



points to that memory location if that string is already present.

str1 =

String str4 = new String("abc");

↳ forcing to create a new object (String) substring →

↓

cont^n part of a string

str = " abcdefghij " ;
0 1 2 3 4 5 6 7 8 9

① Syntax

str. substring (starting index, ending index +1); → str. substring (3, 9)
↳ defghi

last index is excluded

②

str. substring (starting index); → str. substring (4)

↳ to last

↳ efghij

String str= " abcdefghijkl";
0 1 2 3 4 5 6 7 8 9 10 11

str. substring (5, 13) // error.
↳ 5, 12 →

first periority is on ending index

str. substring (5,12)
↳ 5,11

str. substring (5, 5); → (5, 4) → no character in this region → O|P empty

str. substring (12,12); → (12, 11) → ___ x ______ → empty.
end

str. substring (14,14) → (14,13)
↳ both are invalid index → error

str. substring (6) → ghijkl

str. substring (11) → l

str. substring (12) → empty (bcoz it's the length of the string → it's a valid)

str. substring (15) → error → out of bound.

① string → " $\overleftarrow{\text{aba}}$ "

↳ check whether the given string is a palindrome or not?

↳ true   ↳ false.

$$\left.\begin{array}{l} \rightarrow \text{aba} \\ \rightarrow \text{aba} \end{array}\right) \rightarrow \text{palindrome}$$

ex → $\overset{\longleftarrow}{\overset{\longrightarrow}{\text{abcd}}}$ abcd
dcba X

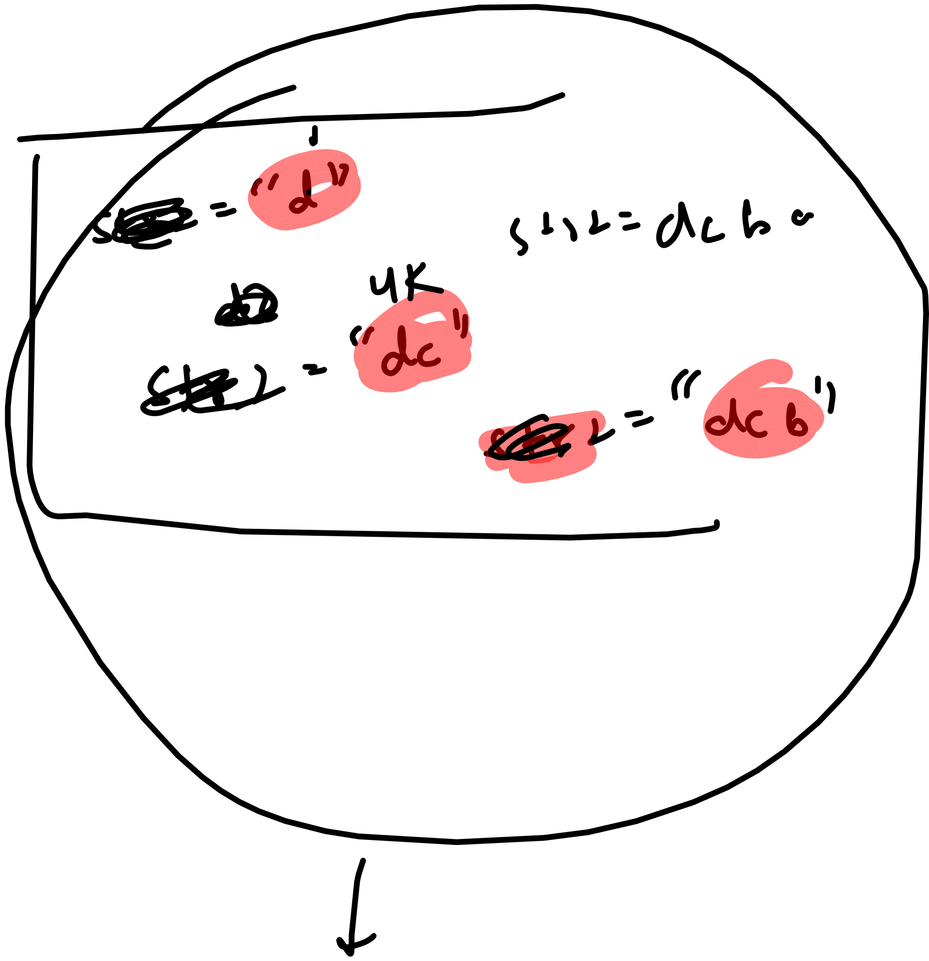
→

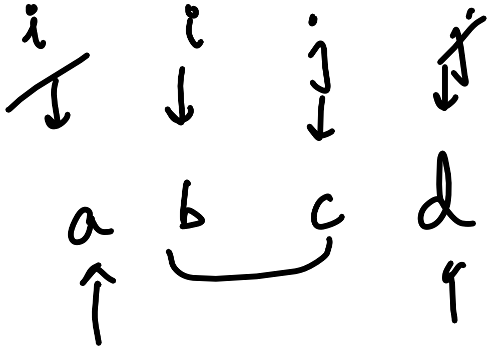
str= "abcd"

str2= 'dc'

str2+ = str.charAt(i);

↳ Everytime java creating a new String in interpool.





```
while (i<j) {
  if(str.charAt(i) != str.charAt(j))
        print(false)
        break;
   }
    i++;
    j--;
```

Functions in Java

$${}^{n}C_{r} = \frac{n!}{r!\,(n-r)!} \rightarrow$$

for(int i=1; i<=n; i++)

$n! = 1 \times 2 \times 3 \times ---- n \rightarrow ans1$

for(int i=1; i<=r; i++)

$r! = 1 \times 2 \times 3 \times --- r \rightarrow ans\ 2$

$(n-r)! = 1 \times 2 \times 3 \times --- (n-r) \rightarrow ans\ 3$

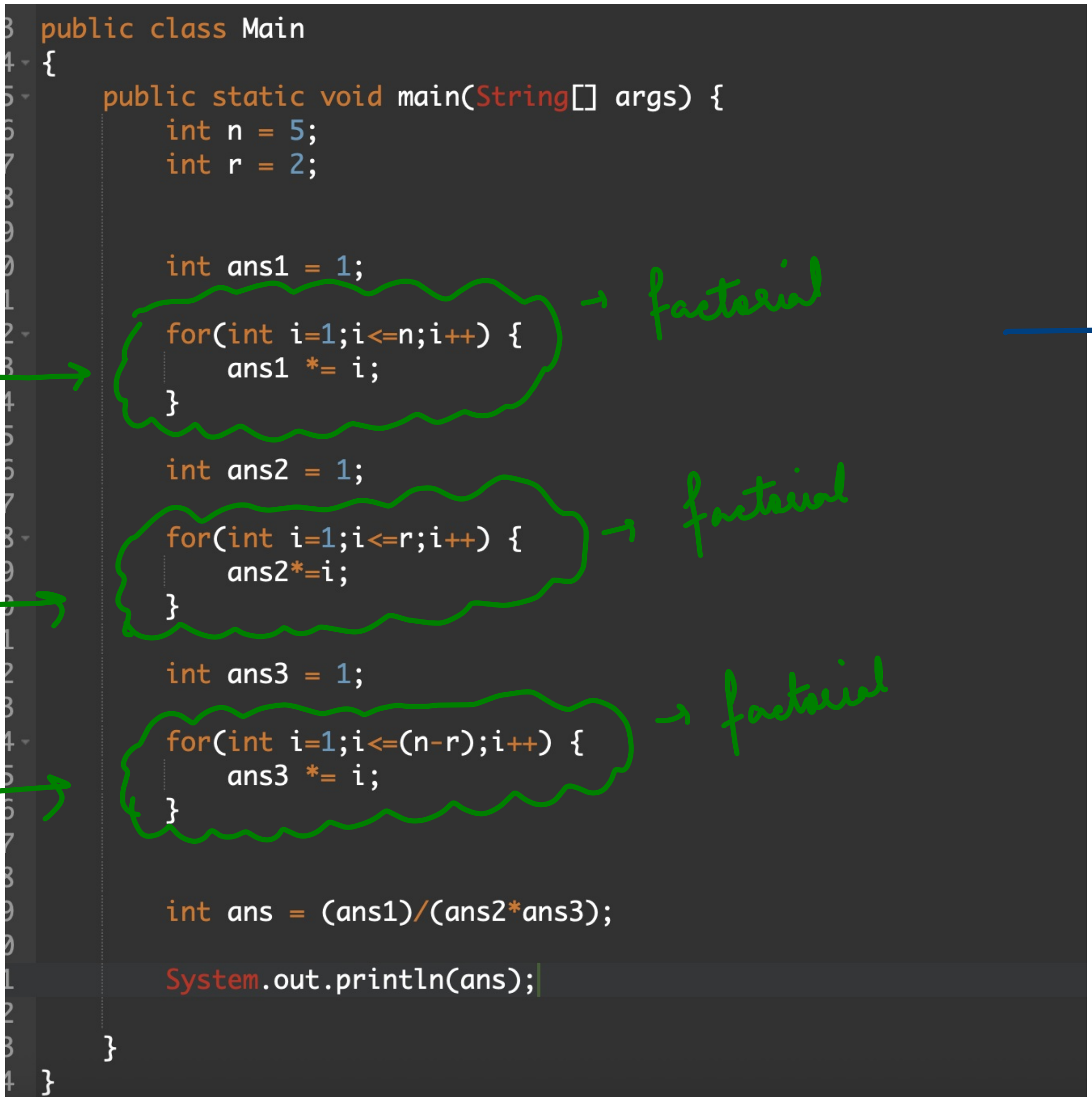
↳ for(int i=1; i<=(n-r); i++)

$$\boxed{ans = (ans1) / (ans2 \times ans3)}$$

$$\frac{5!}{2! \times 3!} \rightarrow \frac{5 \times \overset{2}{\cancel{4}} \times 3!}{\cancel{2} \times \cancel{3!}} = 10$$

DRY → Don't Repeat Yourself.

Bad Coder code.



```
public class Main
{
    public static void main(String[] args) {
        int n = 5;
        int r = 2;


        int ans1 = 1;
        for(int i=1;i<=n;i++) {
            ans1 *= i;
        }

        int ans2 = 1;
        for(int i=1;i<=r;i++) {
            ans2*=i;
        }

        int ans3 = 1;
        for(int i=1;i<=(n-r);i++) {
            ans3 *= i;
        }


        int ans = (ans1)/(ans2*ans3);

        System.out.println(ans);

    }
}
```

Issue

↳ Duplicate code which voilate DRY principle.

↳ Functions.

↳ a block of code used to perform a specific task.

Function :

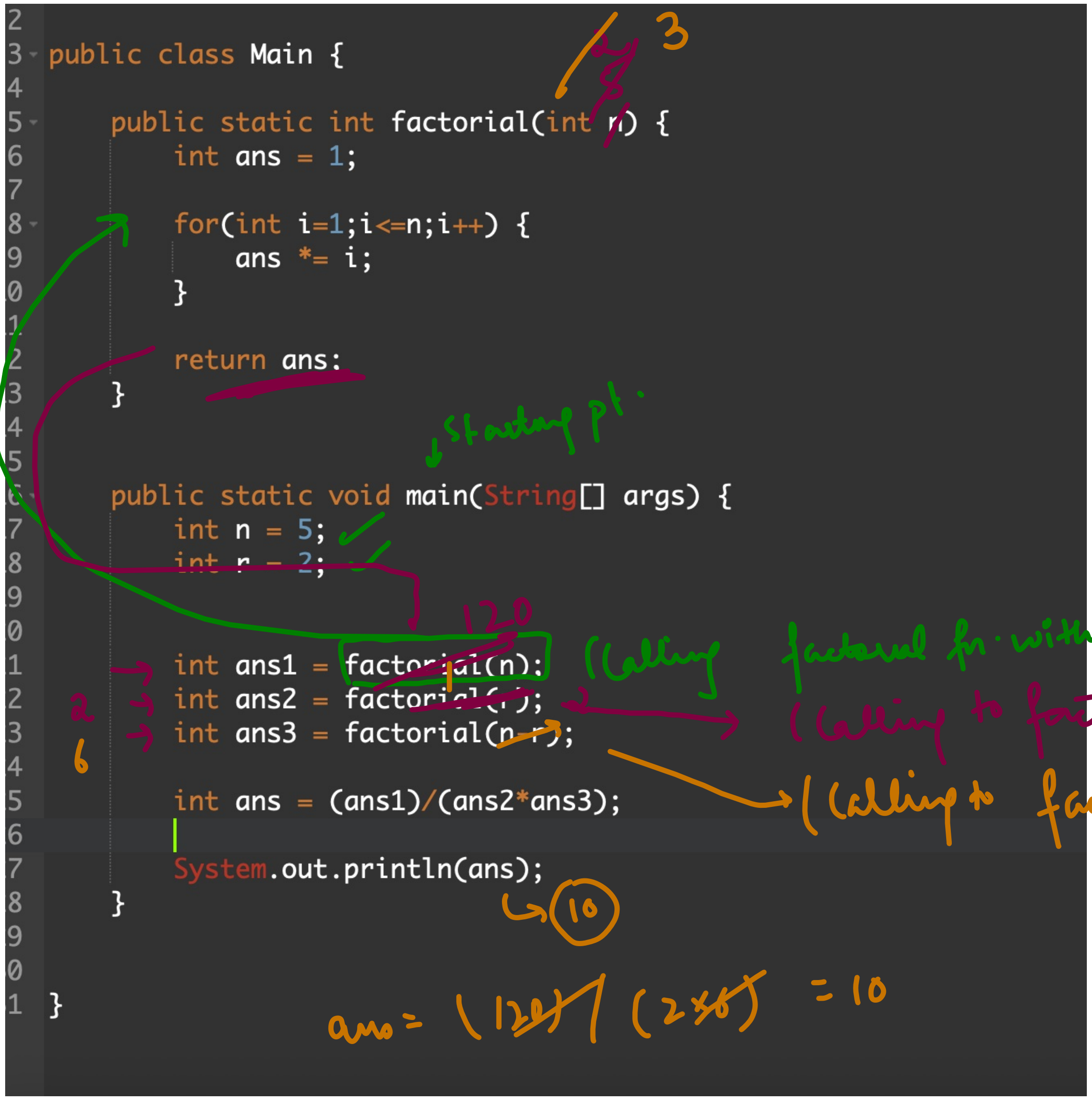
Syntax:-

what type of value fn is returning

0 to n

```
public static return type fn_name ( arguments ) {
    ____
    ____
    ____
    ____
    ____
    return ____ ;
}
```

↳ name of the fn



```java
public class Main {

    public static int factorial(int n) {
        int ans = 1;

        for(int i=1;i<=n;i++) {
            ans *= i;
        }

        return ans;
    }


    public static void main(String[] args) {
        int n = 5;
        int r = 2;


        int ans1 = factorial(n);
        int ans2 = factorial(r);
        int ans3 = factorial(n-r);

        int ans = (ans1)/(ans2*ans3);

        System.out.println(ans);
    }


}
```

3

starting pt.

120

2

6

(calling factorial fn. with value n)

(calling to factorial fn. with value r)

(calling to factorial fn. with value (n-r));

10

ans = (120)/(2×6) = 10

int ans=1
for(int i=1; i<=5; i++) {   }

ans=120

int ans =1
for(int i=1; i<=2; i++)
ans=2

int ans = 1
for(int i=1; i<=3; i++)
ans = 6

→ Return type

- ↳ int → integer
- ↳ boolean
- ↳ void → no return