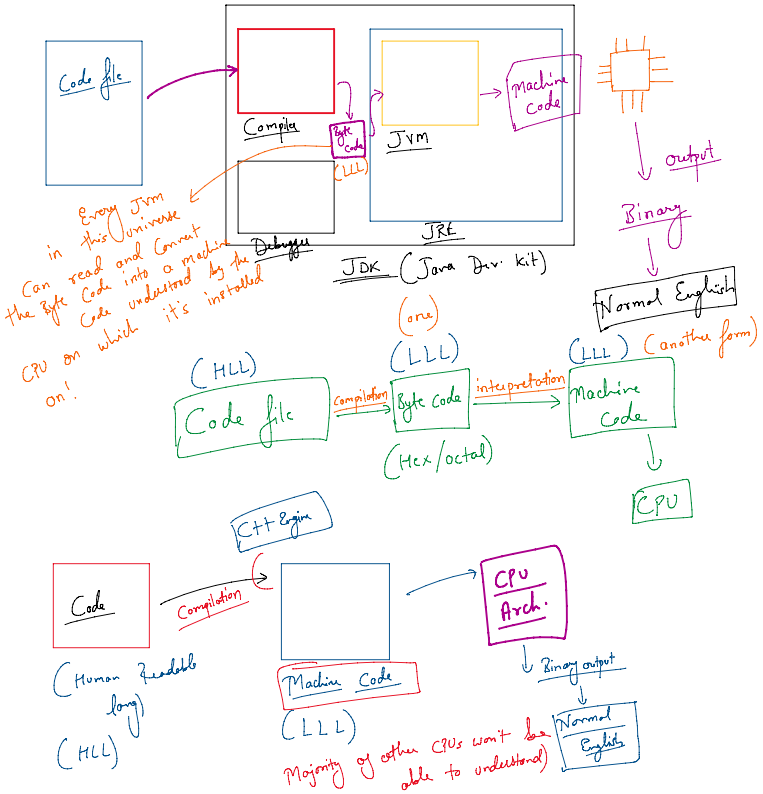
Core Java (Basics)

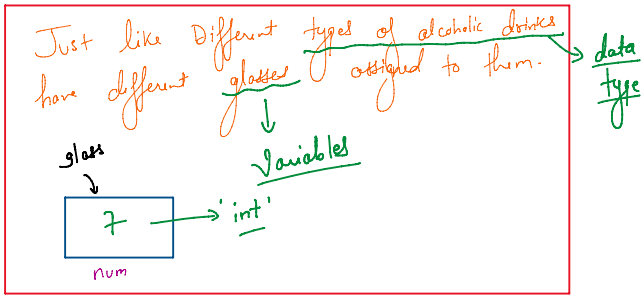
04 December 2023 18:28

★ How Java Code Runs ? ★



★ Variables & Data Types ★

Contains



★ Creating a Variable ★

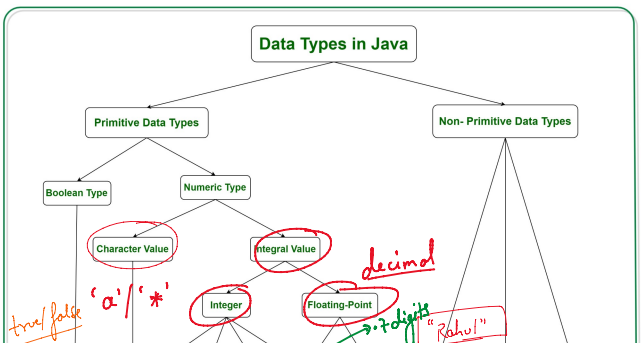
Declaring a Variable : type <Var_name>

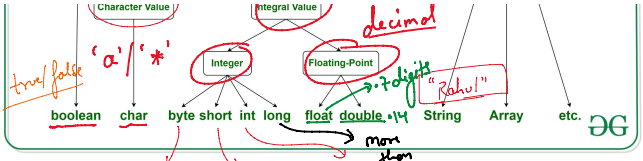
```
int num;
```

★ Putting Data into a Variable ★

(Initializing a Variable)

```
num = 7;
```





byte id

1

2

.

.

10

float num = 7.2f;

double num = 7.257;

long num = 25---L;

String name = "Rahul";

Capital

char ch = '/';

char ch = ' '; (Space is also a character)

L.H.S RHS

int num = 7;

String name = "Rahul";

★ Output Method ★

System.out.print ("Number:" + num);

Number: 7 Name: Rahul

System.out.print ("Name:" + name);

★ Operators ★

(quotient) (Remainder)

① Arithmetic : +, -, /, %, *, ++, --

double a = 5;

double b = 2;

(2.5)

double quotient = 5/2;

double rem = 5 % 2

(1)

int num = -2;

2 — quotient

2 ⟌ 5

-4

1 — Rem

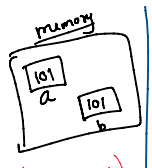
++ (Increment)

pre-increment

a = 100;

b = ++a

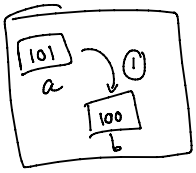
(Left most)



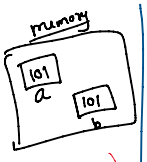
post-increment

a = 100;

b = a++;



Pre-increment

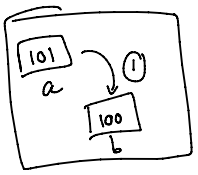


a = 100;

b = ++a; (Left most)

output : a → 101
b → 101

Post-increment



a = 100;

b = a++;

output : a → 101
b → 100

② Conditional : <, >, >=, <=, !=, == (Exactly Equal to?)

③ Assignment : =, +=, -=, *=, /=, %=

a += b

a = a + b

④ Logical : AND, NOT, OR

&&, !, ||

a	b	→
T	T	T

```
double marks = 75;
if ( marks > 55 && marks < 80)
  { Sout(" Grade B");
  {
```

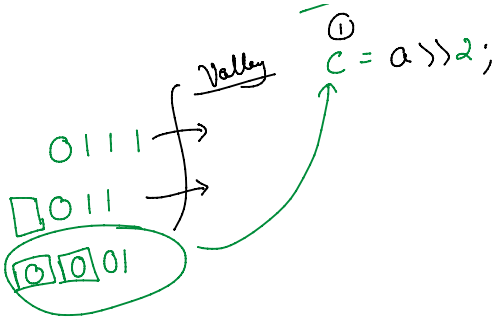
⑤ Bitwise : &, |, <<, >>

```
int a = 7;
int b = 3;
int c = a & b;
```

```
  0 1 1 1
& 0 0 1 1
---------
  0 0 1 1      Decimal → 3
---------
  (1)
```

```
c = a >> 2;
```

Valley

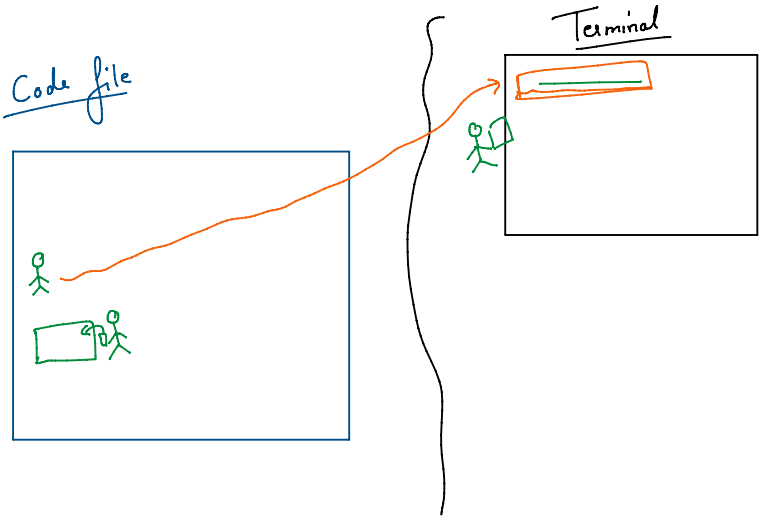


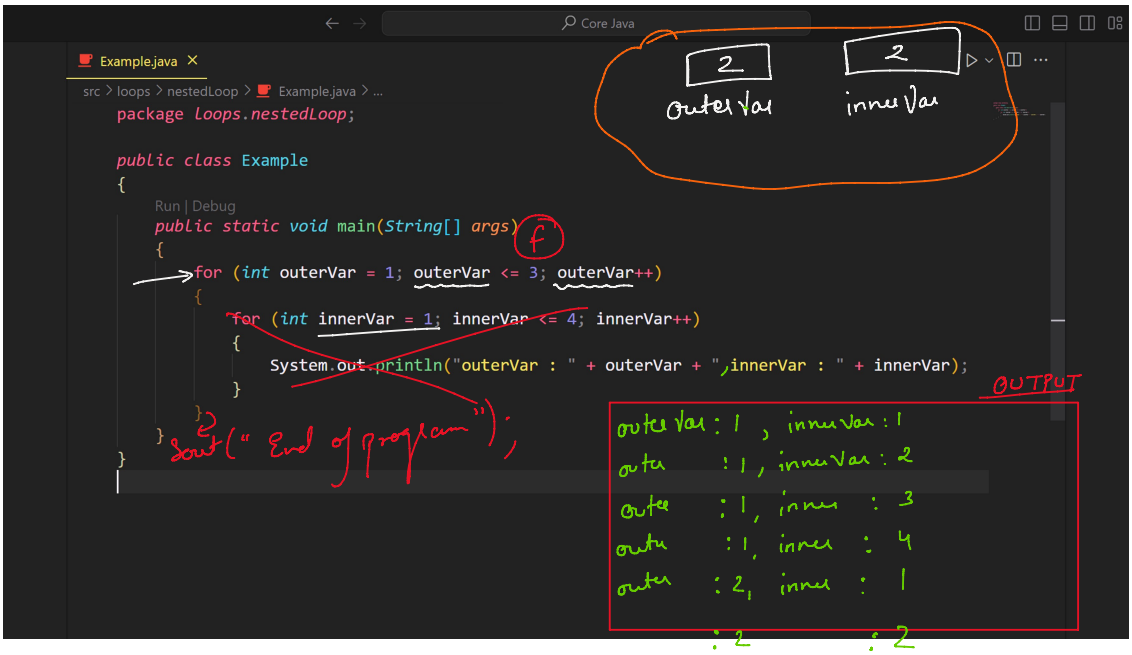
File Name - Class Name

Class Name - Pascal Convention

Variable/function/object Names - Camel Case

Every folder Containing Java files is called 'A Package'





: 2 : 2

2 8

2 4

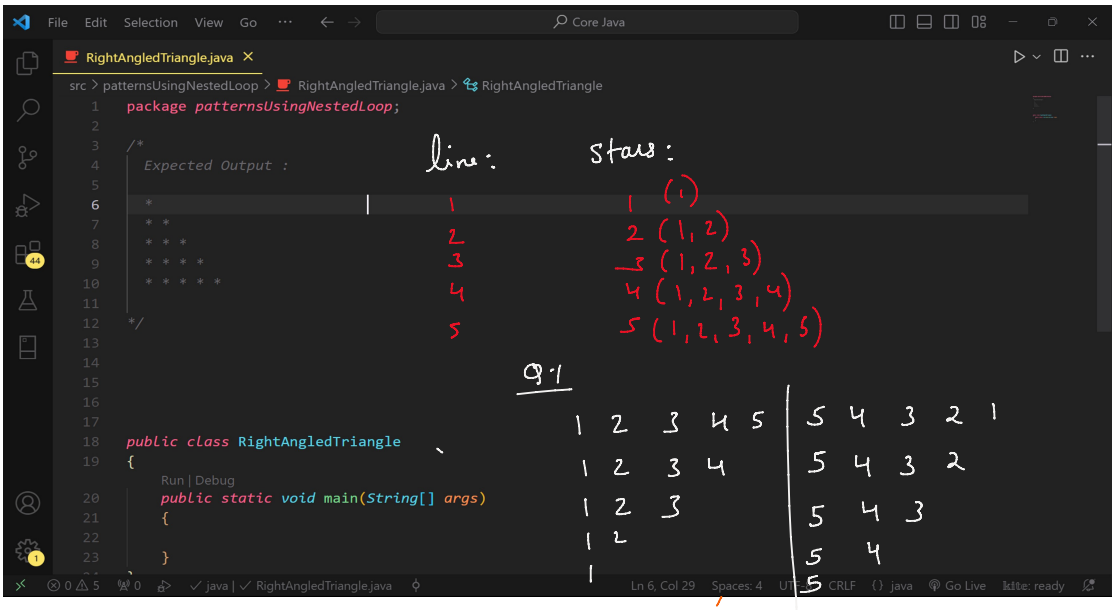
3 1

3 2

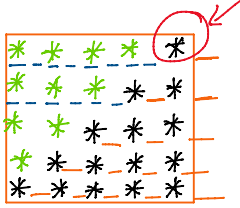
3 3

3 4

End of program



① Complete the Square



Space

8

6

4

2

0

Stars Line

1 — 1

2 — 2

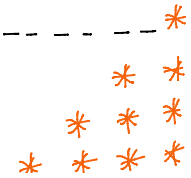
3 — 3

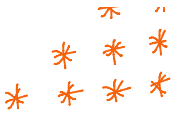
4 — 4

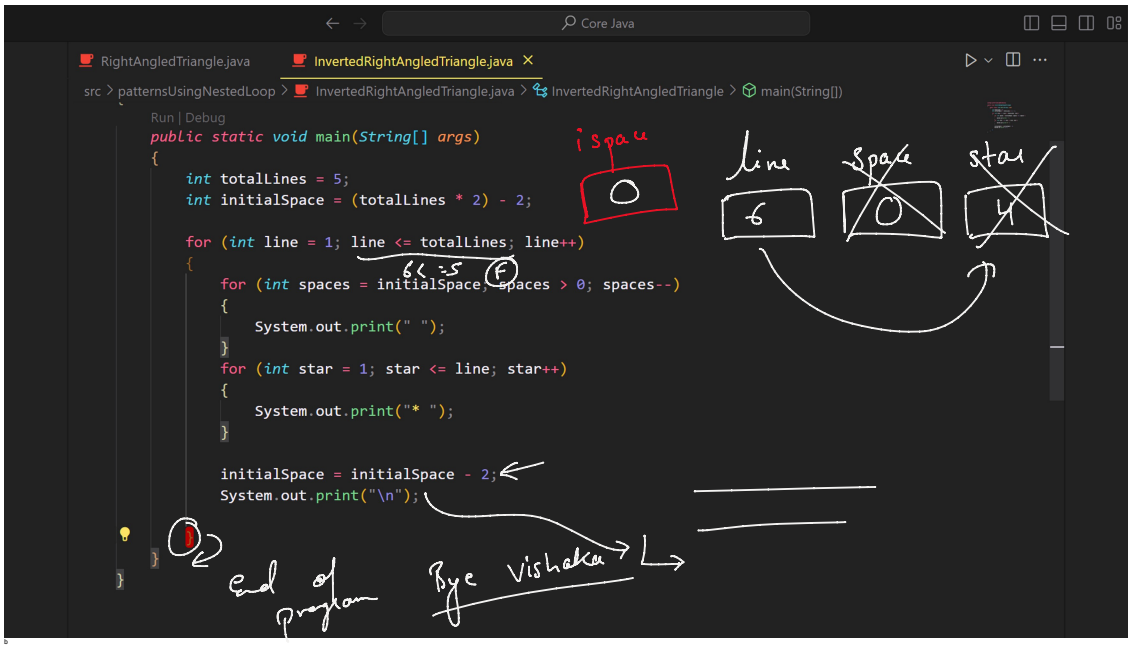
5 — 5

total No of lines - 4 ((*2) -2)

initial_Space: - 6







Increasing order of spaces allotted to data types

byte < char < short < int < long < float < double

[ASCII]

(Widening) type Conversion

(Narrowing) " — " Casting

```
byte b = 127;
int a = 3;
```

X `b = a` Casting

✓ `b = (byte) a;`

trim the data of a to fit in the space provided by 'byte' data type

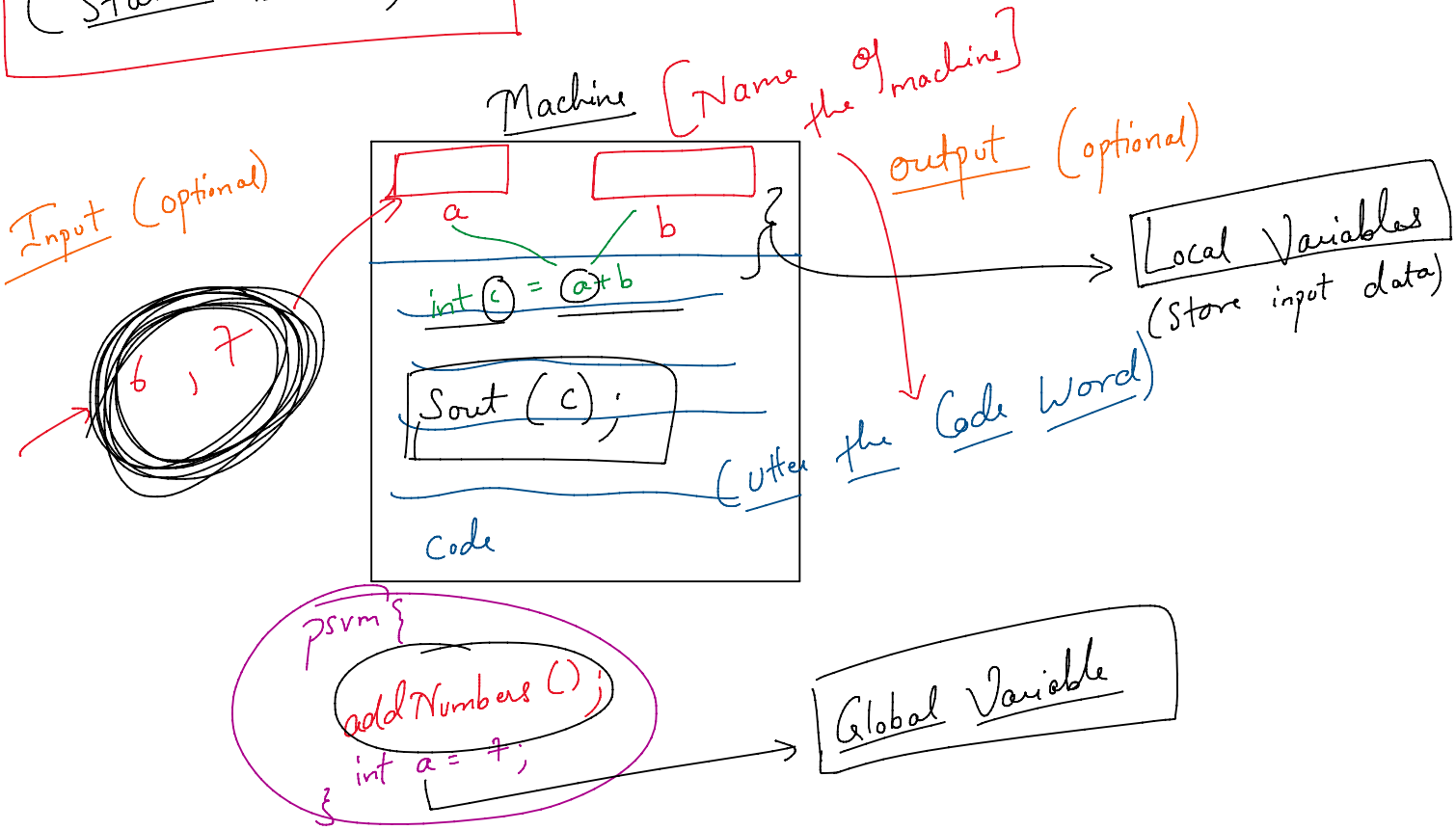
```
int a = 7;
```

```
int a = 7;
float f;
f = a;
```

Type Conversion

☆ Functions ☆

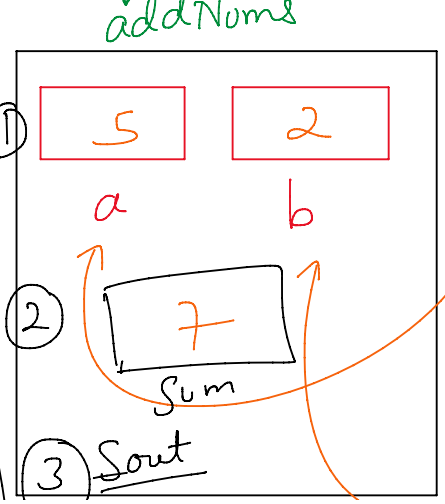
(Static Methods)



Creating a Static Method

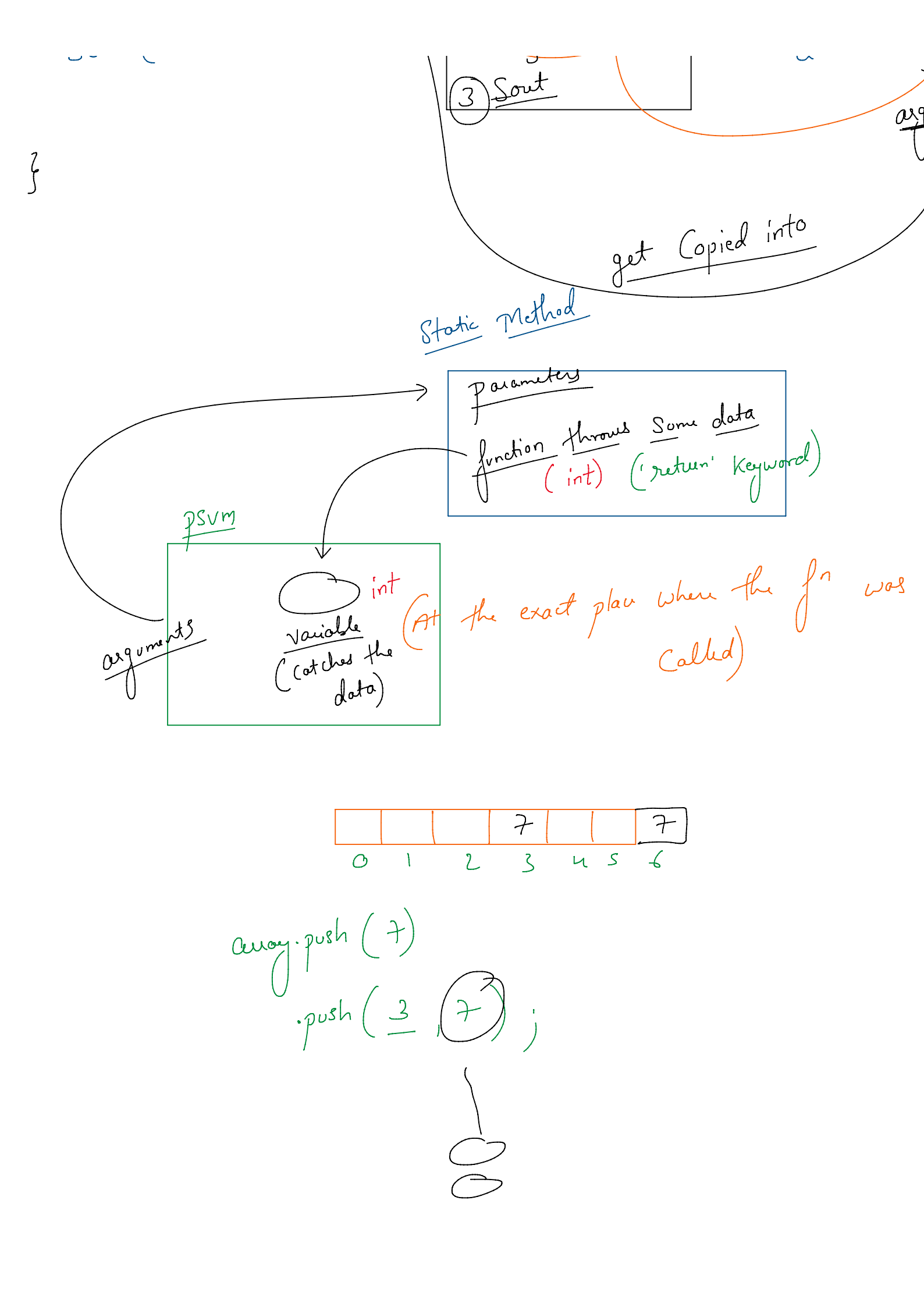
```
public static void addNums (int a, int b)
{
  int Sum = a+b;
  Sout ("Addition is :" + Sum);
}
```

parameters



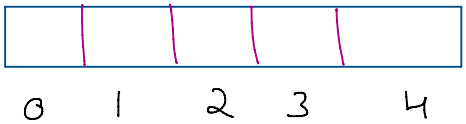

```
psvm
{
  addNums (
  addNums (
```


7, 6);
5, 2);


3 Sout
}
get Copied into
Static Method
Parameters
function throws Some data
(int) ('return' Keyword)
psvm
int
variable
(catches the data)
arguments
(At the exact place where the fn was called)
7
7
0 1 2 3 4 5 6
array.push (7)
.push (3, 7);

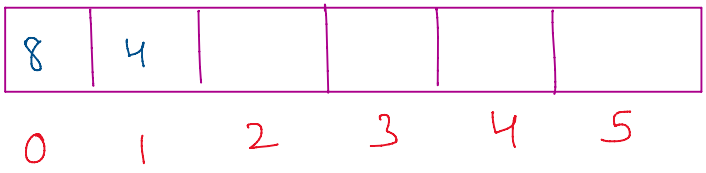

uments

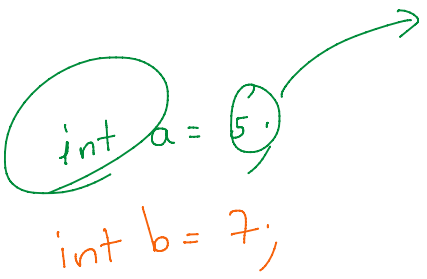
☆ Array ☆ (Same data-type)



☆ Arrays ☆

(1) Elements are stored in Consecutive memory
(2) Each element Can be Considered as a





Memory

label	Add.
a	3001 H
	3002
	3003
	3004
	3005

1 bit = Smallest unit

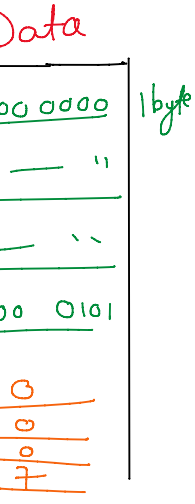
8 bits = 2 Nibbles = 1 byte

1024 bytes = 1 Kb

1024 Kb = 1 mb

1 - 1 Gb

location
Variable.

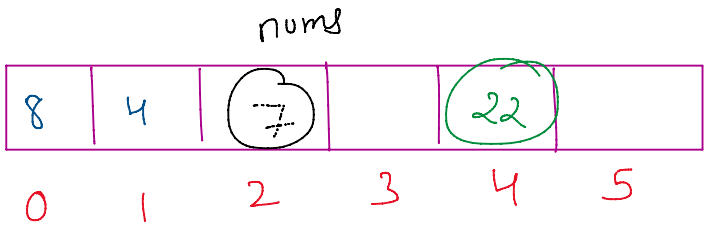


1024 Kb = 1 mb

1024 mb = 1 Gb

1024 Gb = 1 Tb

1024 Tb = 1 Pb



nums[2] = 7;

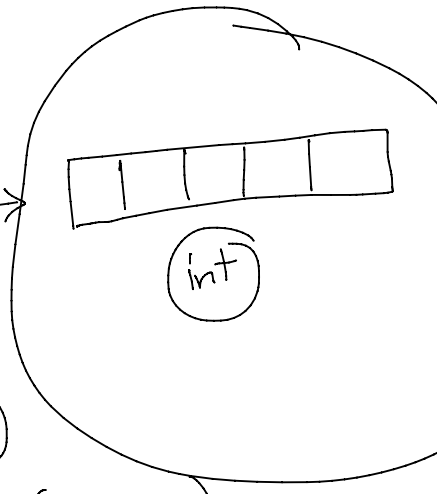
nums[4] = 22;

[Arrays have fixed size]

★ Creating an Empty Array ★

No. of elements that array

```
int [ ] nums = new int [5];
int [ ] nums
int nums[ ] =
int nums [ ] =
```

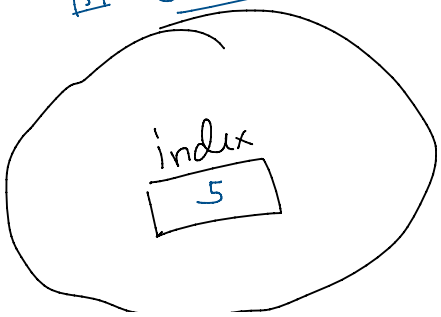


nums [0] = ____;

nums [1] = ____; l → 5 (n)

nums [2] = ____; i → 0 to 4 (0, n-1)

name of 1st element



SC

5

```
for( int index = 0; index < nums.length;
{
    Sout (" Element Ki Value enter ka
    nums [index] = SC.nextInt();
```

background → (nums[4] = 22)

refer to elements

shall hold.

index ++)

o:");

background → (nums[4] = 22)

{ }

: 22

for ( )

{

Sout (nums[index]);

}

Declaration + Initialization of an Array :

int nums [ ] = {5, 7, 2, 3, 8, 1, 3}; (length 7)

PRIME NUMBER PROGRAM USING ARRAYS :

[Divisible only by 1 or themselves]

5	21	7	10	2

OR

[Not Divisible by any Number rather than 1 or itself]

i.e ||

[If a Number (n) gets divided

1 — ×

2 — 1, 2 ✓

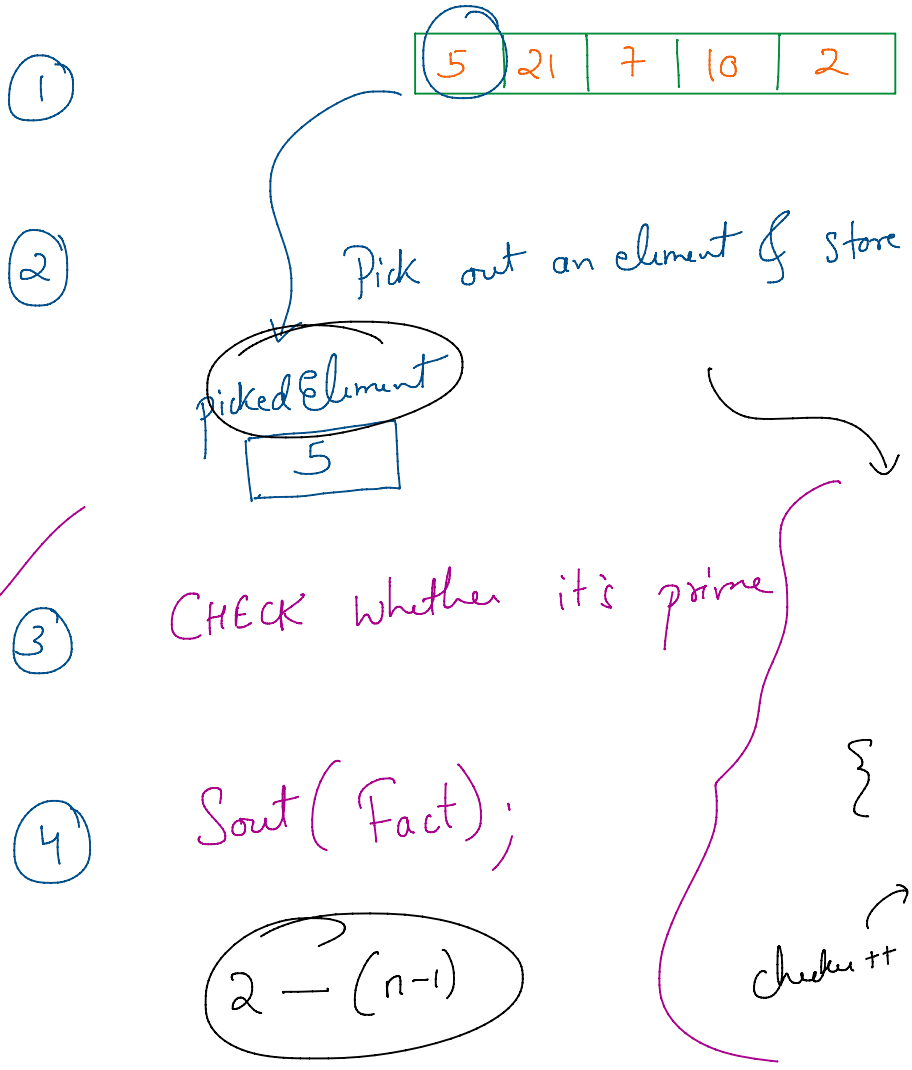
3 → 1, 3 ✓

4 → 1, 2, 4 ×

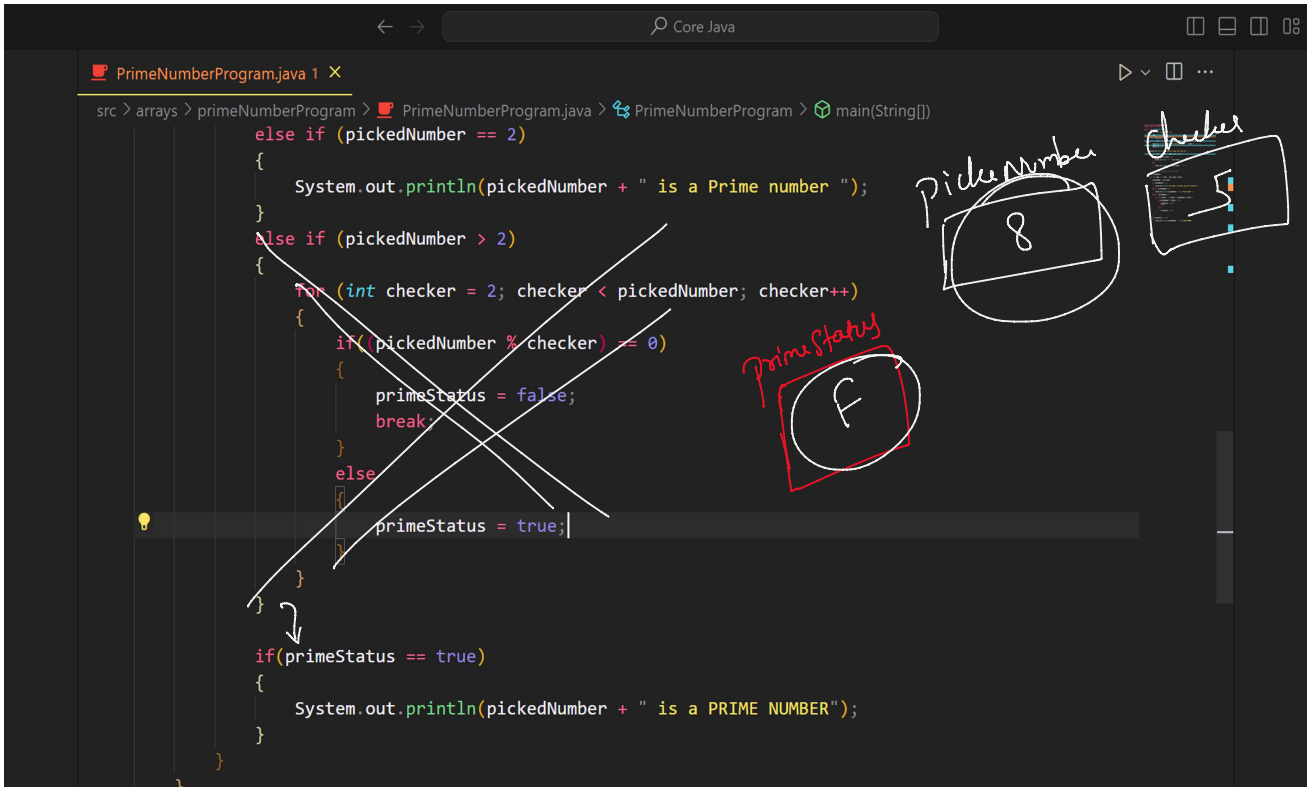
5 → 1, 5 ✓

[If a Number (n) gets divided by any number in the range 2 — (n-1), then it's NOT A PRIME NUMBER]

2 — 10




or Composite
A1)
n 0:
PRIME
execution
loop



1

for (1 — 100

pickedNumber

Checking

The prime nos in the range

7, ,

Else

2 —— PRIME

3 ——

—— NOT A

2

(min s elements)

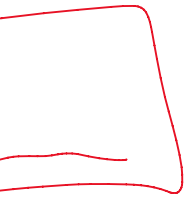
2	5	7	8	6

(Sum of all the Elements

Array)

28

of 1 to 100 a



ts of

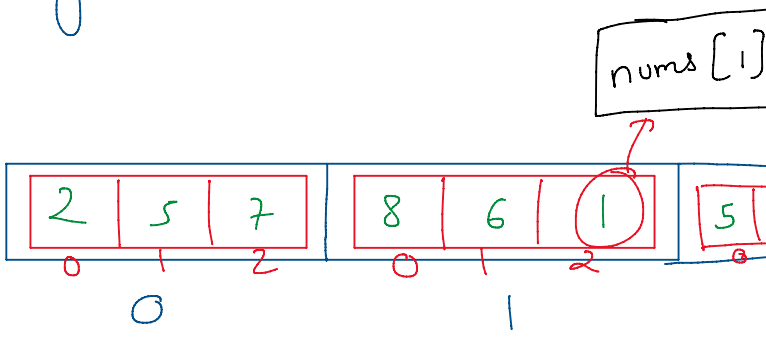
Sum

★ 2D Array

Array inside another Array.

3	5	6
7	8	1
10	2	9

nums




```
int nums[][] = new int[3][3];
```

[0][0]

[0][1]

[0][2]

[1][0]

[1][1]

[1][2]

[2][0]

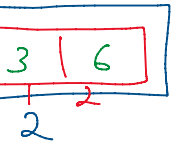
[2][1]

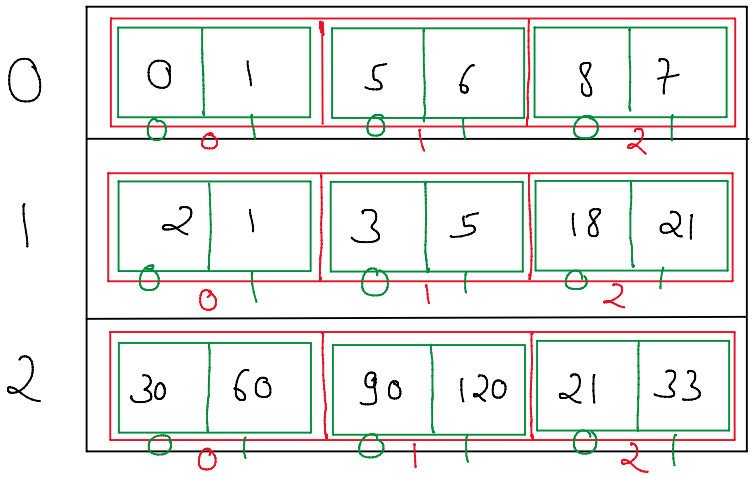
[2][2]

★ 3D Array ★

Array inside an Array inside an Array


[2];


3
6
2
2

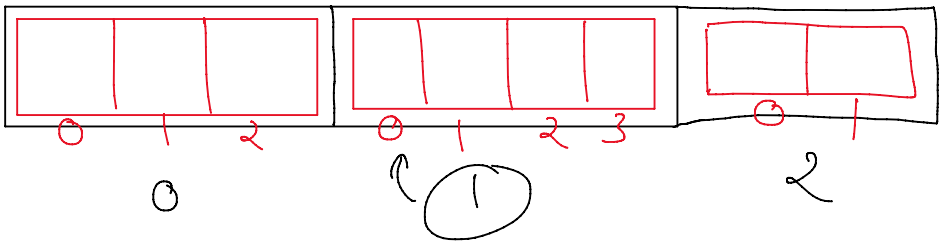



```
for
{
    for
    {
        for
        {
        }
    }
}
```

0 0 0
0 0 1
0 1 0
0 1 1
0 2 0
⋮

```
int nums[][][] = new int[3][3][2];
```

★ Jagged Array ★

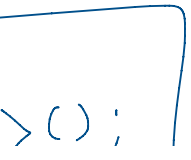



```
int nums[][] = new int[3][];
nums[0] = new int[3];
nums[1] = new int[4];
nums[2] = new int[2];
```

★ Arraylist ★

(Variable sized Array)

Arraylist<



(Vaue

Declaration : Arraylist < Wrapper class > name = new Arraylist<

Methods (Predefined):

1. add
2. get
3. remove
4. Size (length of Arraylist);
5. Set

★ OOPS ★ (Solve: Real Life Problems - Virtually)

- Plan (class)
- object (object)

Every object

- Certain Attributes
- Functions (Methods)

Plan (class) Cars

Attributes

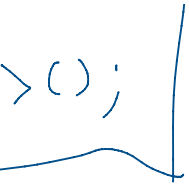
- model

Object instance

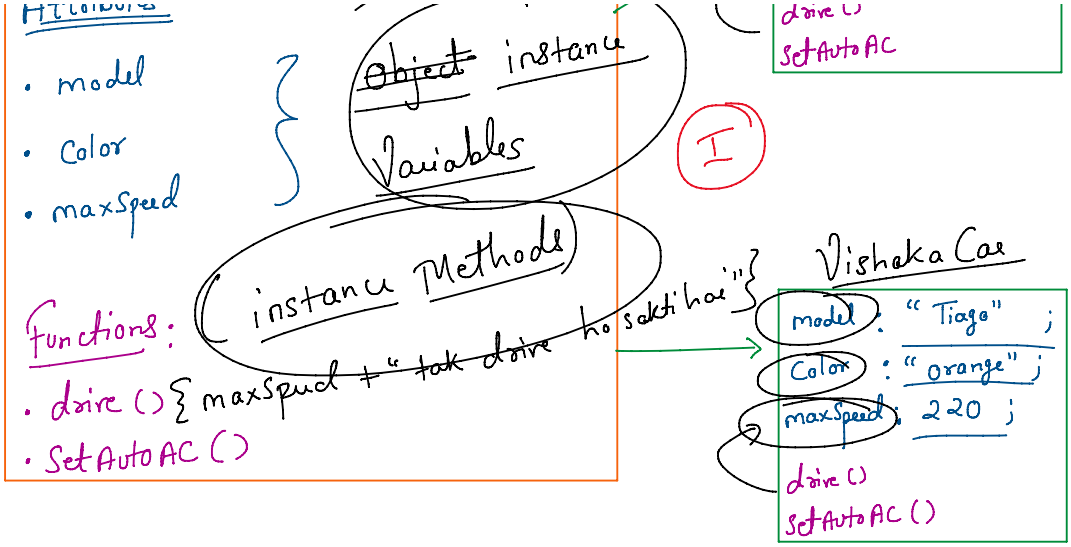
RahulCar

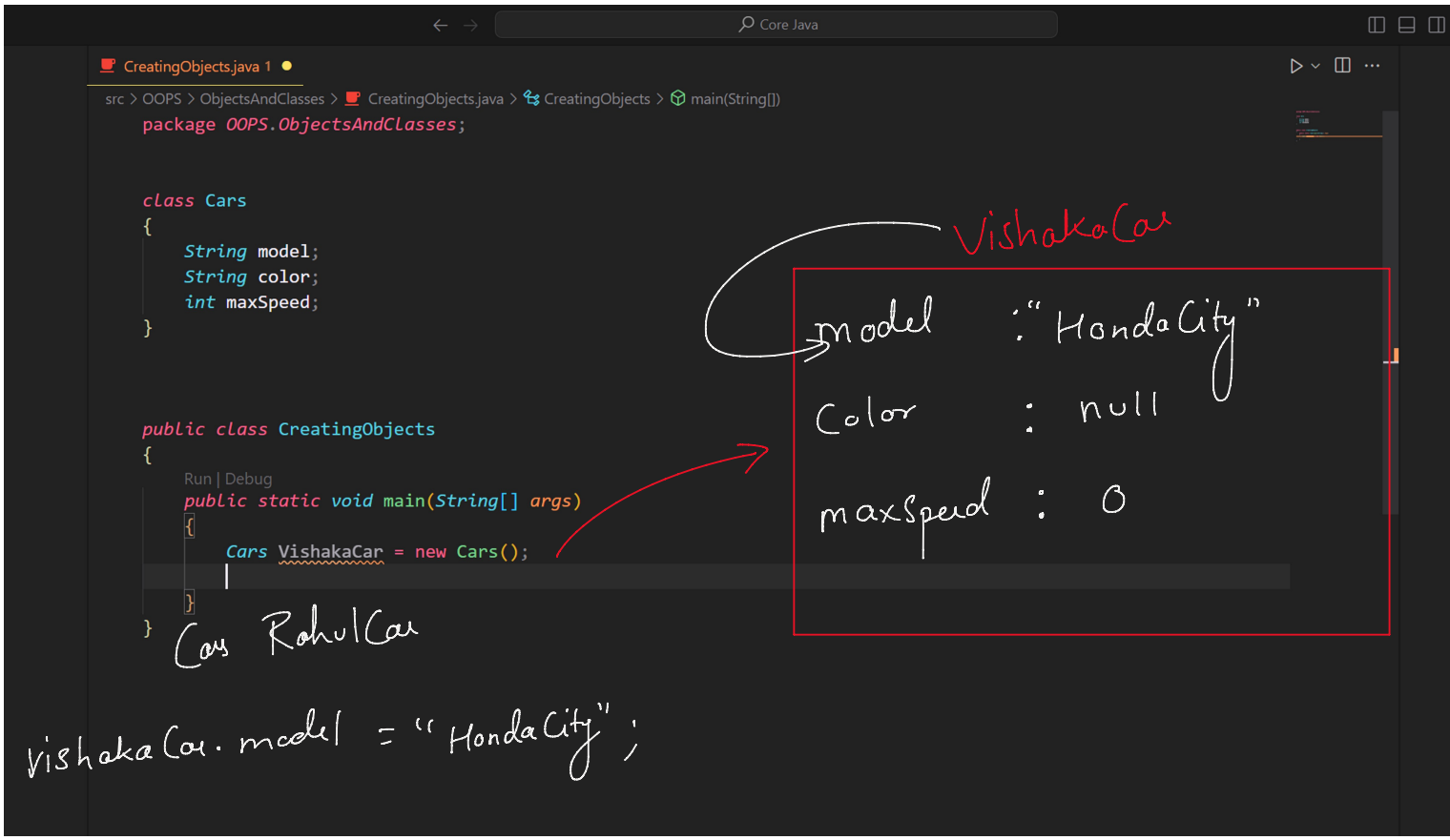
model : "Honda City";
Color : "black";
maxSpeed : 240 ;
drive()
setAutoAC

An object

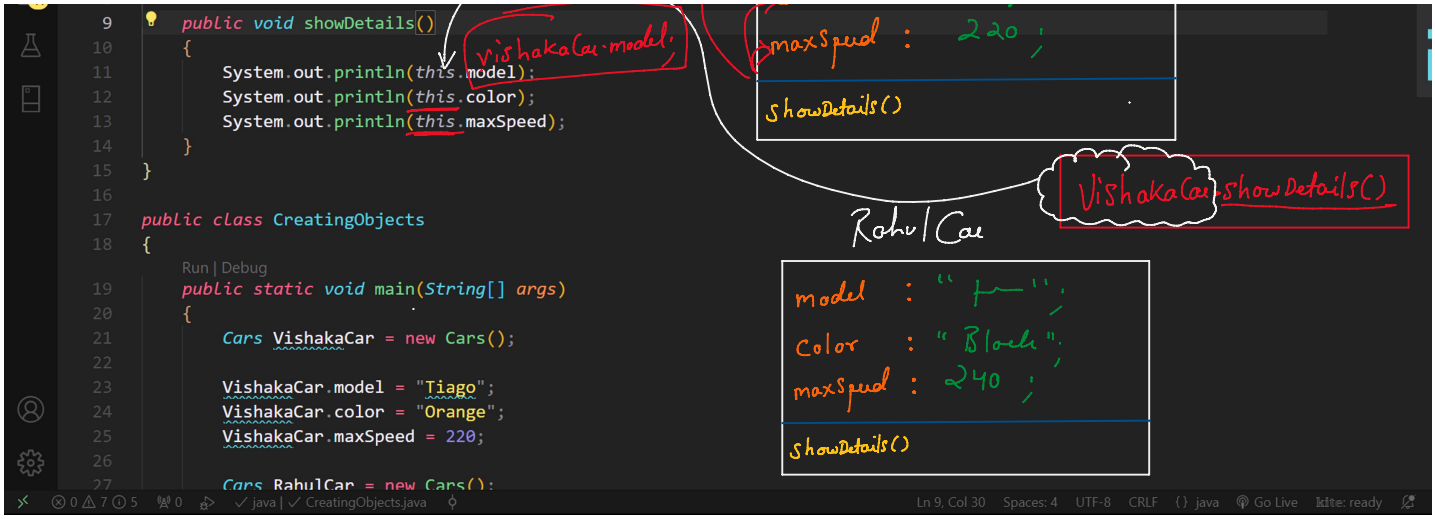

>();

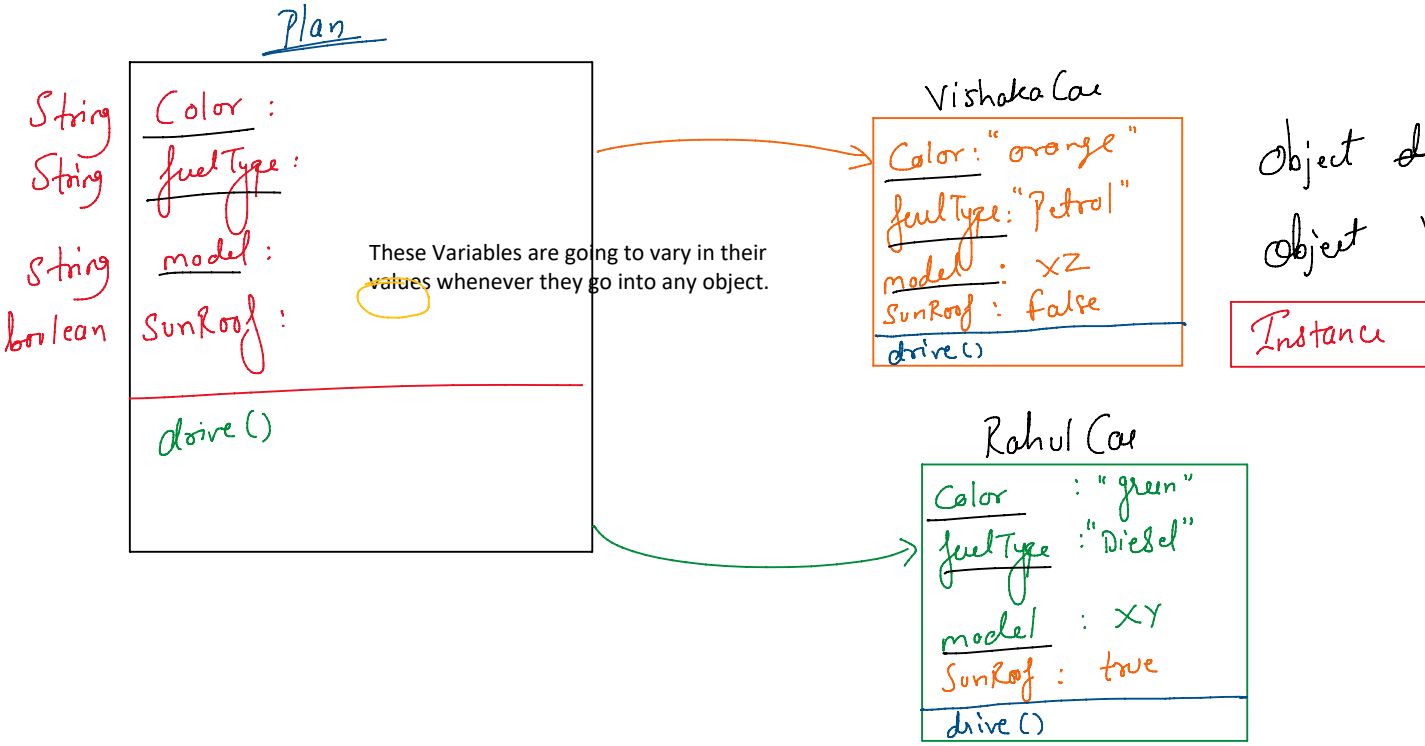

is
ud


Attributes
• model
• Color
• maxSpeed
Object instance Variables
Functions:
• drive () { maxSpeed + " tak drive ho sakti hai" }
• SetAutoAC ()
instance Methods
I
drive ()
SetAutoAC
Vishaka Car
model : " Tiago " ;
Color : " orange " ;
maxSpeed : 220 ;
drive ()
SetAutoAC ()
Static


Core Java
CreatingObjects.java 1
src > OOPS > ObjectsAndClasses > CreatingObjects.java > CreatingObjects > main(String[])
package OOPS.ObjectsAndClasses;
class Cars
{
String model;
String color;
int maxSpeed;
}
public class CreatingObjects
{
Run | Debug
public static void main(String[] args)
{
Cars VishakaCar = new Cars();
}
}
Cars RahulCar
VishakaCar
model : "HondaCity"
Color : null
maxSpeed : 0
VishakaCar.model = "HondaCity";


File Edit Selection View Go Run
Core Java
CreatingObjects.java
src > OOPS > ObjectsAndClasses > CreatingObjects.java > Cars > showDetails()
1 package OOPS.ObjectsAndClasses;
3 class Cars
4 {
5 String model;
6 String color;
7 int maxSpeed;
9 public void showDetails()
10 {
11 System.out.println(this.model);
VishakaCar.model
WHAT
WHY
VishakaCar
model : " Tiago " ;
Color : " orange "
maxSpeed : 220 ;


public void showDetails()
{
System.out.println(this.model);
System.out.println(this.color);
System.out.println(this.maxSpeed);
}
}
public class CreatingObjects
{
Run | Debug
public static void main(String[] args)
{
Cars VishakaCar = new Cars();
VishakaCar.model = "Tiago";
VishakaCar.color = "Orange";
VishakaCar.maxSpeed = 220;
Cars RahulCar = new Cars();
VishakaCar.model
maxSpeed : 220 ;
showDetails()
VishakaCar.showDetails()
Rahul Car
model : " ";
Color : " Black ";
maxSpeed : 240 ;
showDetails()


Plan
String Color :
String fuelType :
String model :
boolean SunRoof :
drive()
These Variables are going to vary in their values whenever they go into any object.
Vishaka Car
Color : "orange"
fuelType : "Petrol"
model : XZ
SunRoof : false
drive()
Rahul Car
Color : "green"
fuelType : "Diesel"
model : XY
SunRoof : true
drive()
object
object
Instance

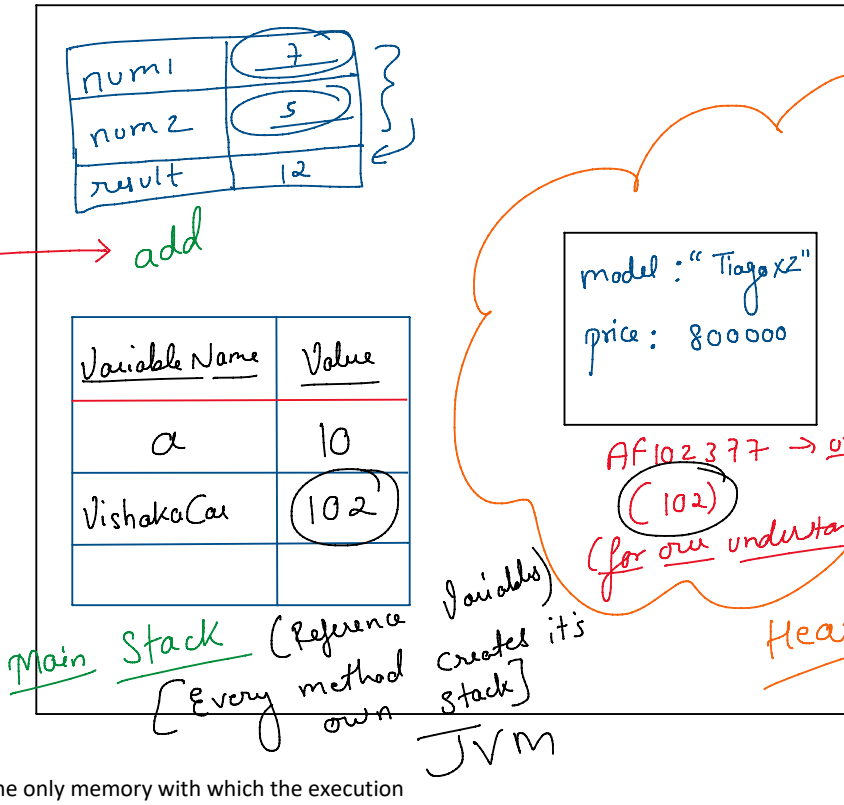
☆ Heap & Stack ☆

~~ding~~ Variables

iables

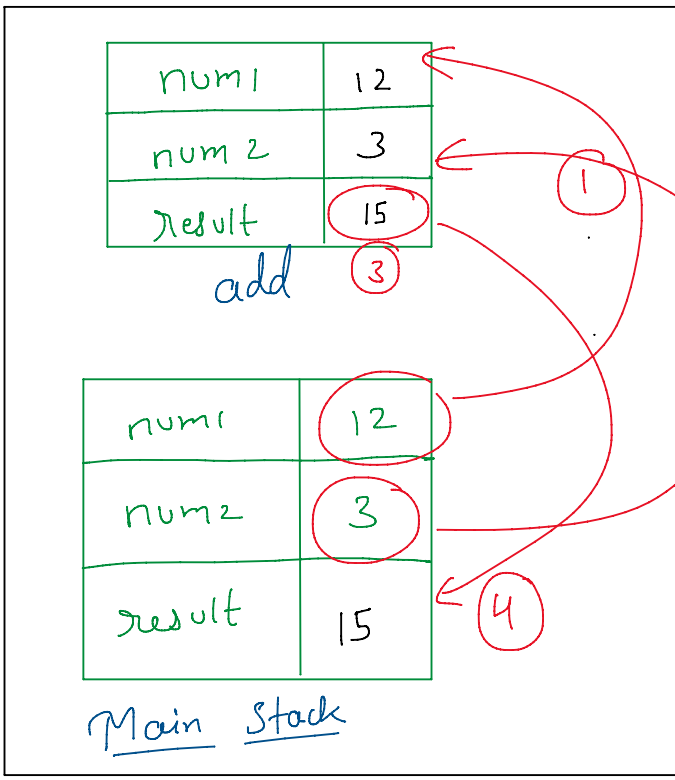
iables

```java
package OOPS.stackAndheap;

class Cars
{
    String model;
    float price;
}
public class Example
{
    Run | Debug
    public static void main(String[] args)
    {
        int a = 10;

        Cars VishakaCar = new Cars();
        VishakaCar.model = "Tiago XZ";
        VishakaCar.price = 8_00_000f;
        System.out.println(VishakaCar.price);
    }
}
```

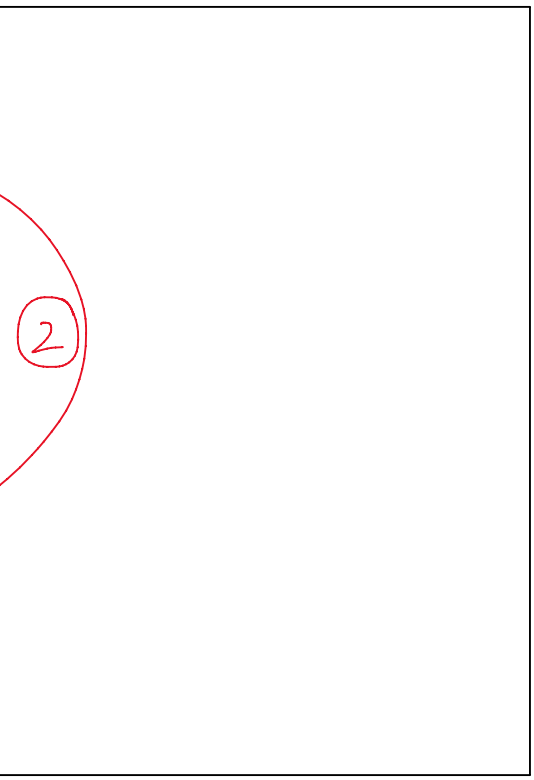


Stack memory is the only memory with which the execution engine can interact. That means our execution engine cannot interact with heap. So stack is a way which creates a link between heap and itself.

```java
package OOPS.stackAndheap;

public class TwentyDecember
{
    public static int add(int num1, int num2)
    {
        int result = num1 + num2;
        return result;
    }

    Run | Debug
    public static void main(String[] args)
    {
        int num1 = 12;
        int num2 = 3;
        int result = add(num1, num2);
        System.out.println(result);
    }
}
```



Each stack can have variables of unique



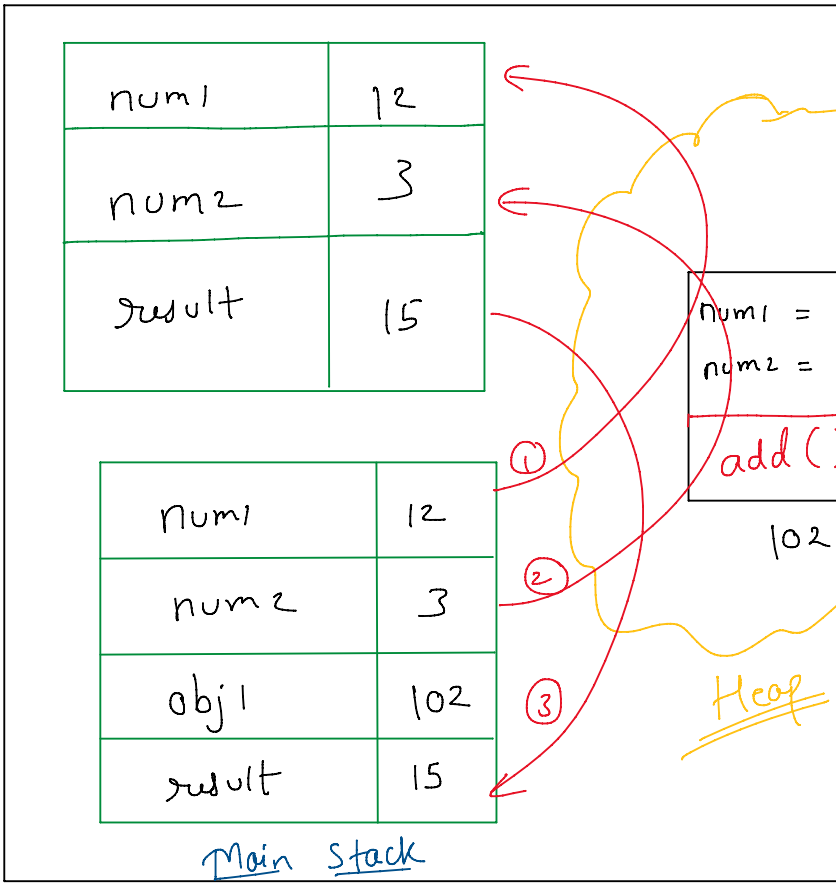
names

only

TwentyDecember2.java

src > OOPS > stackAndheap > TwentyDecember2.java > ...

```java
package OOPS.stackAndheap;

class ArithmeticCalc
{
    int num1;
    int num2;

    public ArithmeticCalc(int NUM1, int NUM2)
    {
        this.num1 = NUM1;
        this.num2 = NUM2;
    }

    public int add(int num1, int num2)
    {
        int result = num1 + num2;
        return result;
    }
}

public class TwentyDecember2
{
    Run | Debug
    public static void main(String[] args)
    {
        int num1 = 12;
        int num2 = 3;

        ArithmeticCalc obj1 = new ArithmeticCalc(1, 2);

        int result = obj1.add(num1, num2);
        System.out.println(result);
    }
}
```

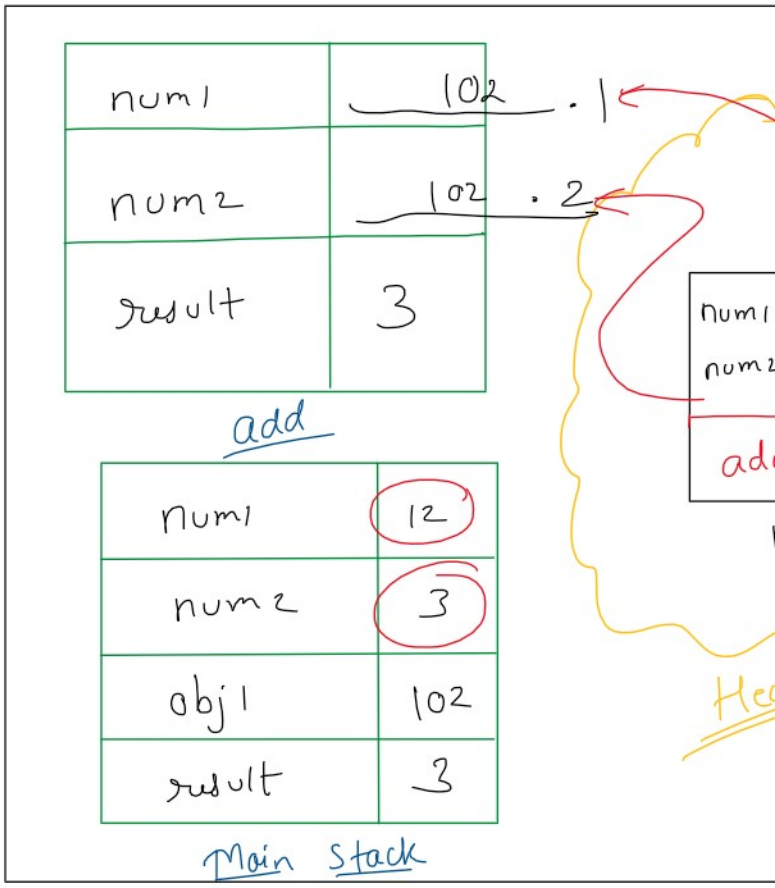



1
2

TwentyDecember2.java

src > OOPS > stackAndheap > TwentyDecember2.java > ArithmeticCalc >

```java
package OOPS.stackAndheap;

class ArithmeticCalc
{
    int num1;
    int num2;

    public ArithmeticCalc(int NUM1, int NUM2)
    {
        this.num1 = NUM1;
        this.num2 = NUM2;
    }

    public int add(int num1, int num2)
    {
        num1 = this.num1;
        num2 = this.num2;
        int result = num1 + num2;
        return result;
    }
}

public class TwentyDecember2
{
    Run | Debug
    public static void main(String[] args)
    {
        int num1 = 12;
        int num2 = 3;

        ArithmeticCalc obj1 = new ArithmeticCalc(1, 2);

        int result = obj1.add(num1, num2);
        System.out.println(result);
    }
}
```



★ Strings ★

1. Every String object has indices in it
   [Every letter has an index]
2. Strings are stored inside Heap memory in the String Constant Pool
3. "Strings in java are immutable"
   [Can't be modified / changed]

Str = "RAHUL"

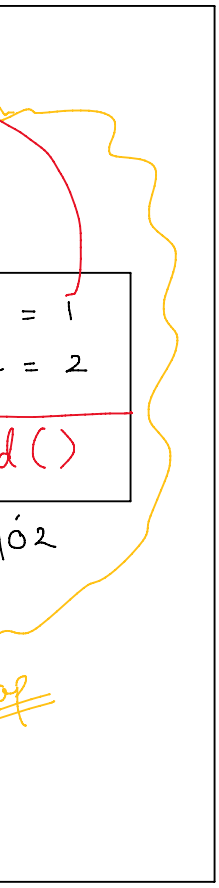
0 1 2 3 4

Sout(str.charAt(3)); 'U'

revStr

[str.length() - 1]

for(int index = ; index >=0; index


= 1
= 2
d()
02

Sout(str.charAt( ));

name = "VISHAKA"

0 1 2 3 4 5 6

rev_str = " ";

AKAHSIV

for(int index = ; index >=0; index

{

revStr = revStr + Str.charAt(ind

AK

}

Q:1

"White Kurti"	"floral top"	"Shrug"

"Skirt"	"Jeans"	"Leggings"

White Kurti : Skirt

:

:

floral top :

White Kurti : Skirt	White kurti : Jeans

Sout all the Combined Elements which are inside arraylist using foreach Loop.

CombString = + " :

outfits.add( );

Q-2

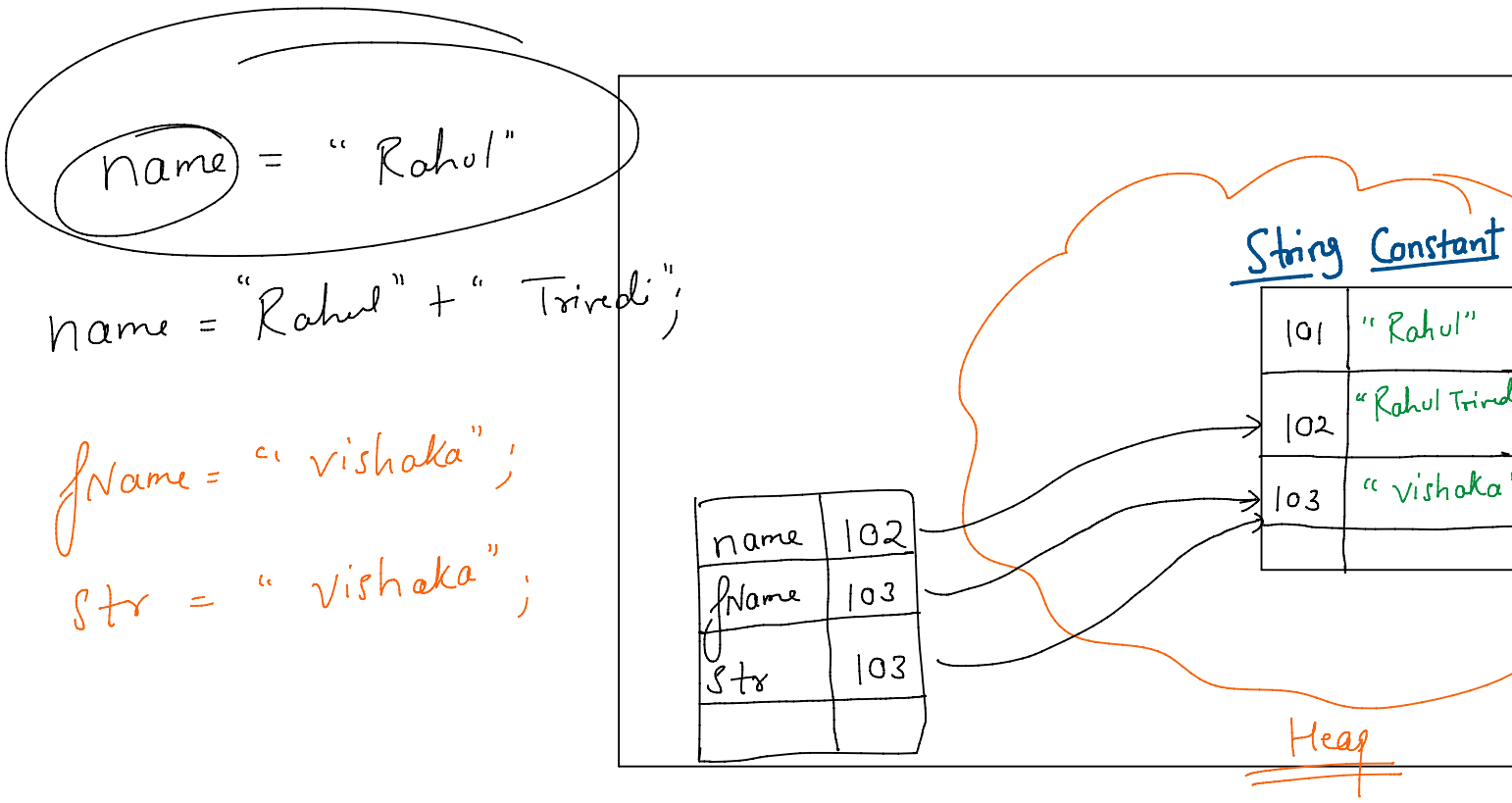
"RAHUL"	"VISHAKA"	"SNEHA"
0	1	2

name

var

"LUHAR"	"AKAHSIV"	"AHENS"

nam

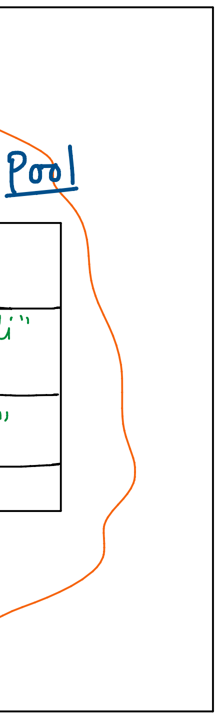


Whenever We make any updations in the existing string object, Another object gets created with the updated value but the original object stays unaffected. Hence, it is said that strings in Java are immutable because the original object is not experiencing any mutation.

Str = "RAHUL"

"RAHUL"

nes



(by-default method

(str == name)

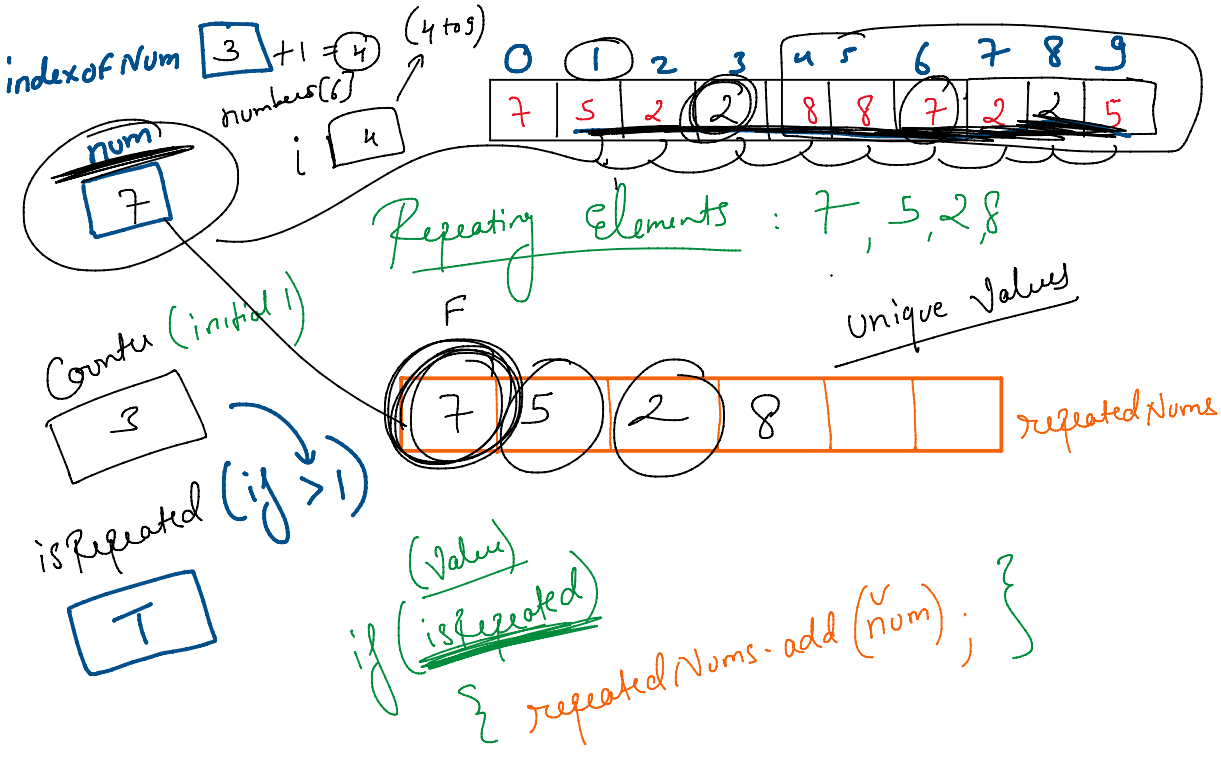
name = "RAHUL"

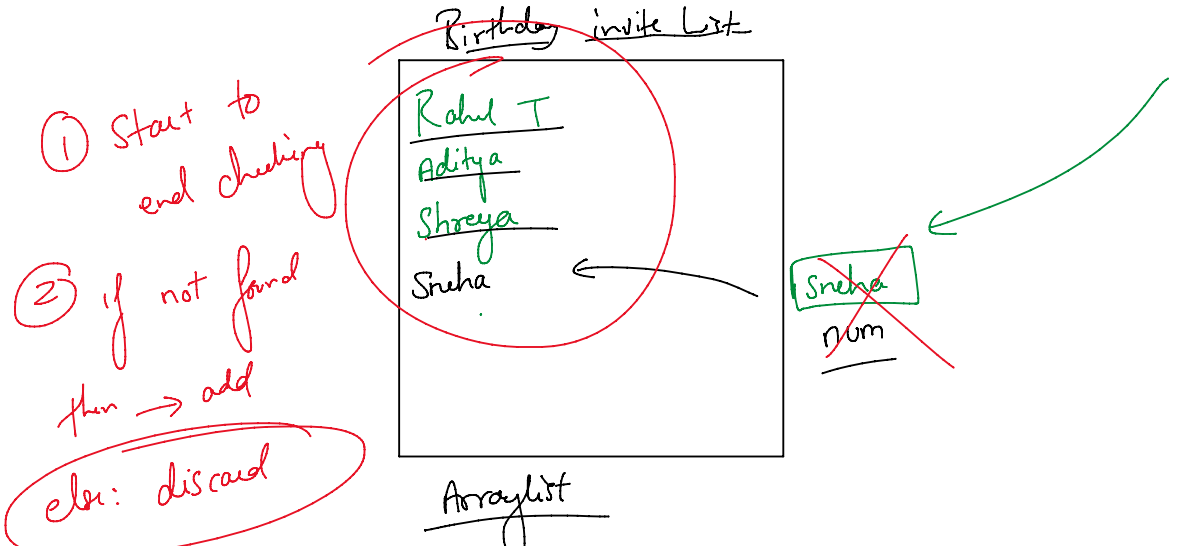
String class

Sort(Str.equals(name)

true

(7,8) →





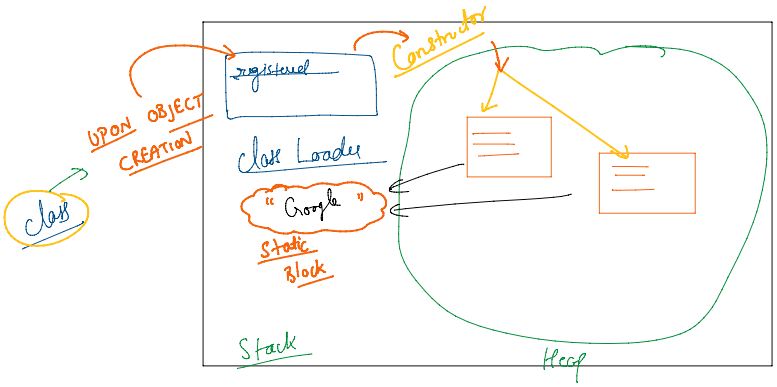
Sout(str == name)

↓

false

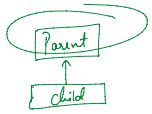
ele: discard

ArrayList

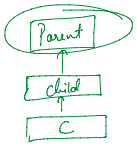


Types of Inheritance in Java :

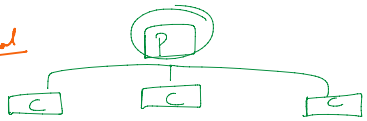
1. Single



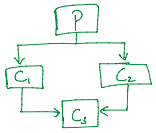
2. Multilevel :



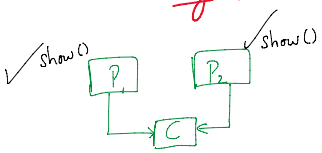
3. Hierarchical

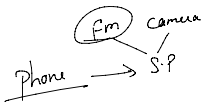


4. Hybrid



5. Multiple inheritance (Not Supported using classes)

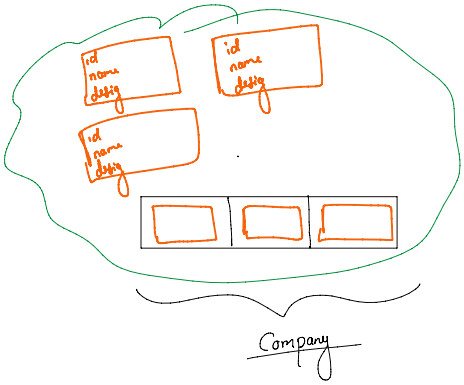


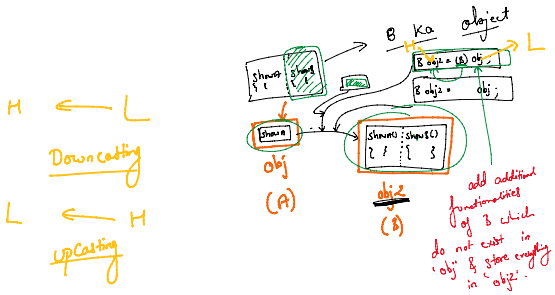


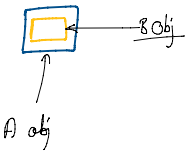
interface Fm
{
}
// features

[A newer version of Something cannot be a successor of two things (though it may implement features/concepts of other things)]

class SmartPhone extends Phone implements Fm, Camera


id
name
dobj
id
name
dobj
id
name
dobj
Company


H
L
Down casting
L
H
UpCasting
B ka object
H
L
obj
(A)
obj
(B)
add additional functionalities of B which do not exist in 'obj' & store everything in 'obj2'.


B obj
A obj


File Edit Selection View Go
Example1.java
src > needForInterfaces > Example1.java > Example1 > main(String[])
class Laptop {
public void runTheCode() {
System.out.println("code, compile & run if all OK !!!");
}
}
class Developer {
public void developApp(Laptop obj) {
System.out.println("Developing an app....");
obj.runTheCode();
}
}
public class Example1 {
Run | Debug
public static void main(String[] args) {
Developer Rahul = new Developer();
Laptop HPOmen15 = new Laptop();
Rahul.developApp(HPOmen15);
}
obj
101
HPOmen15
101
runTheCode()
{ c,c,r }
101