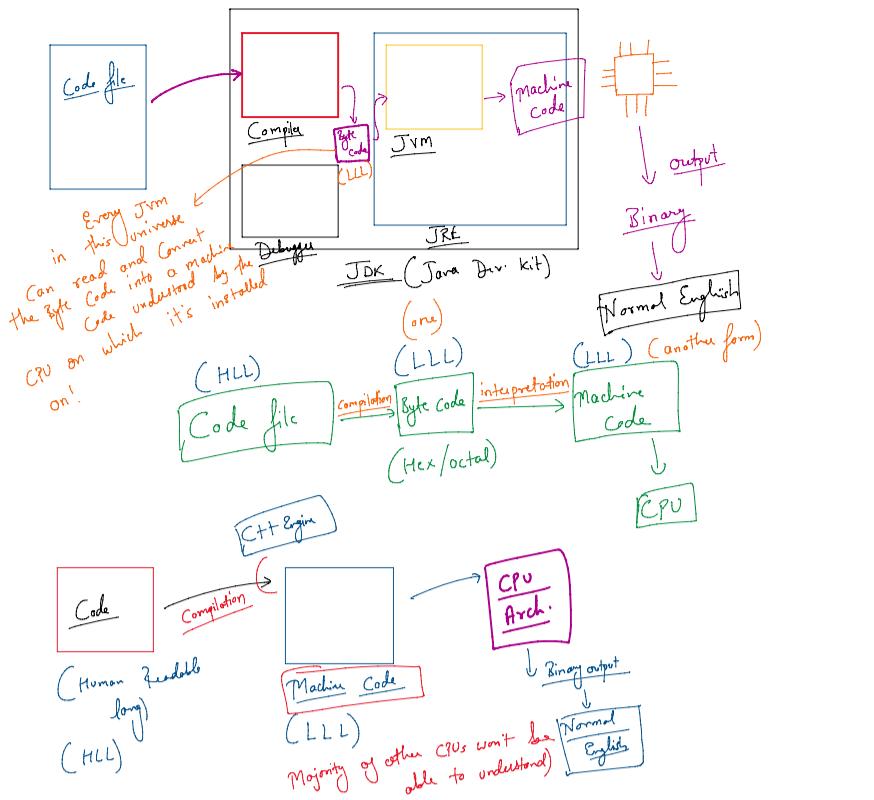
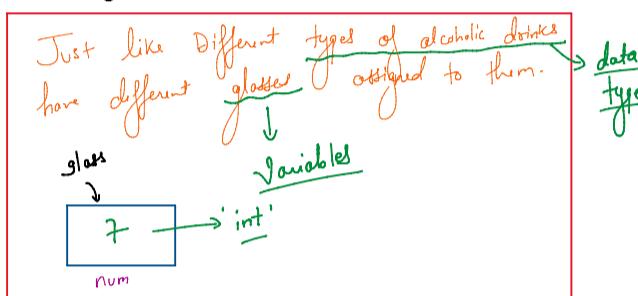


★ How Java Code Runs? ★

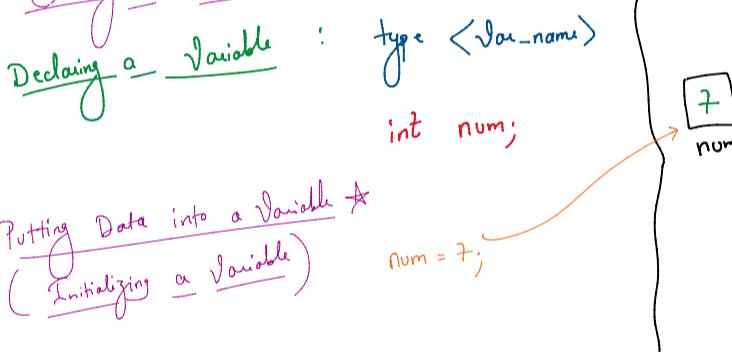


★ Variables & Data Types ★

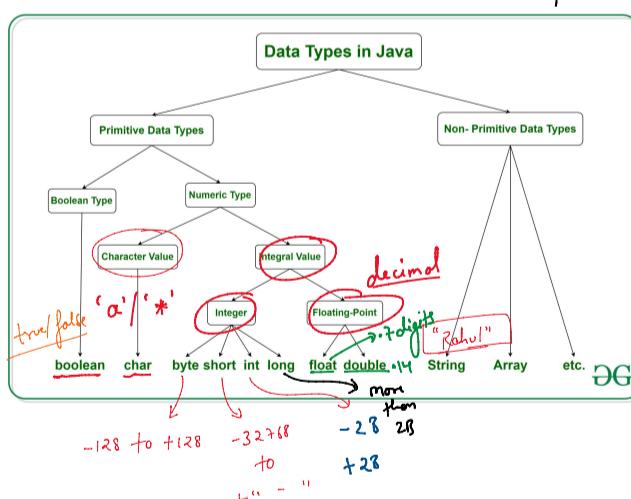
Containers



★ Creating a Variable ★



★ Putting Data into a Variable ★



byte id
 1 float num = 7.2f;
 2 double num = 7.25f;
 :
 10 long num = 25L;
 String name = "Rahul";

Capital
 char ch = '/';
 char ch = ' ';

(Space is also a character)

int num = 7;
 String name = "Rahul";

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

★ Output Method ★


```

String num = " ";
System.out.print("Number :" + num);
System.out.print("Name :" + name);

```

Number : 7 Name: Rohit

★ Operators ★

① Arithmetic: +, -, /, %, *, ++, -- (quotient) (remainder)

```

double a = 5;
double b = 2;
double quotient = 5 / 2;           2   — quotient
double rem = 5 % 2;               2   - 4
(1)                                1 — Rem
int num = -2;

```

++ (Increment)

pre-increment
 $a = 100;$
 $b = \textcircled{a} + a$ (Left most)

output: $a \rightarrow 101$
 $b \rightarrow 101$

post-increment
 $a = 100;$
 $b = a \textcircled{+}$

output: $a \rightarrow 101$
 $b \rightarrow 100$

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

③ Assignment: =, +=, -=, *=, /=, ./.=

$a \textcircled{+}= b$
 \downarrow
 $a = a \textcircled{+} b$

④ Logical: AND, NOT, OR
&&, !, ||

a b \rightarrow
T T T

$\text{double marks} = 75;$
if (marks > 55 $\textcircled{\&\&}$ marks < 80)

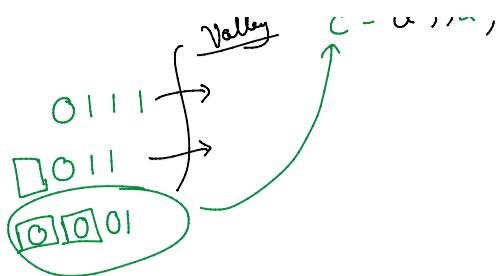
{ Sout("Grade B");

}

⑤ Bitwise:
int a = 7;
int b = 3;
int c = a & b;

$\begin{array}{r} 0111 \\ 0011 \\ \hline 0011 \end{array}$ Decimal $\rightarrow 3$

$\textcircled{1} c = a \gg 2;$

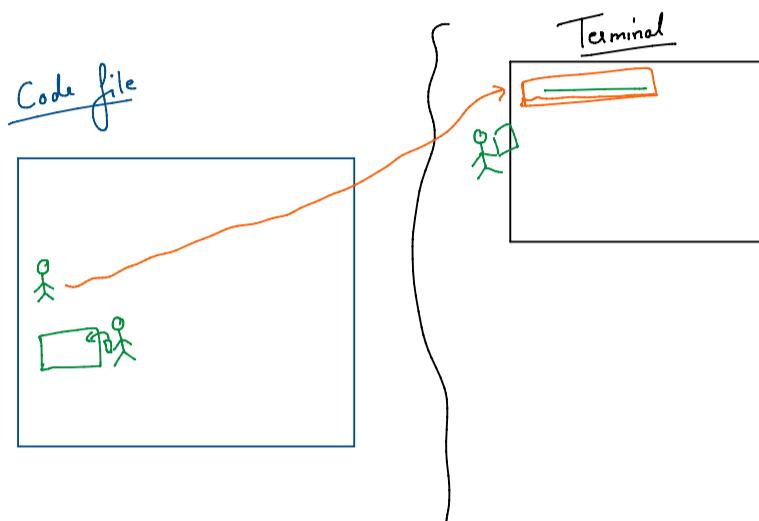


File Name - Class Name

Class Name - Pascal Convention

Variable/function/object Names - Camel Case

Every folder Containing Java files is called 'A Package'



```

Example.java
src > loops > nestedLoop > Example.java ...
package loops.nestedLoop;

public class Example
{
    public static void main(String[] args)
    {
        for (int outerVar = 1; outerVar <= 3; outerVar++)
        {
            for (int innerVar = 1; innerVar <= 4; innerVar++)
            {
                System.out.println("outerVar : " + outerVar + ", innerVar : " + innerVar);
            }
        }
        System.out.println("End of program");
    }
}

```

OUTPUT

outerVar : 1 , innerVar : 1	
outerVar : 1 , innerVar : 2	
outerVar : 1 , innerVar : 3	
outerVar : 1 , innerVar : 4	
outerVar : 2 , innerVar : 1	
: 2 : 2	
2 2	
3 3	
3 3	
3 4	

End of program

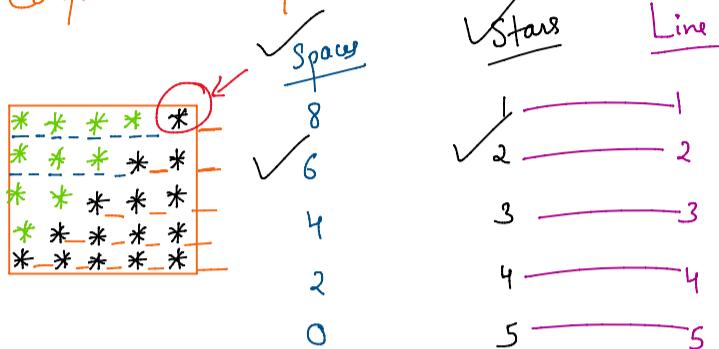
RightAngledTriangle.java

```

1 package patternsUsingNestedLoop;
2
3 /*
4  * Expected Output :
5   *          line:      stars:
6   *          |       | (1)
7   *          *       2 (1, 2)
8   *          **      3 (1, 2, 3)
9   *          ***     4 (1, 2, 3, 4)
10  *          ****    5 (1, 2, 3, 4, 5)
11 */
12
13
14
15
16
17
18 public class RightAngledTriangle {
19
20     public static void main(String[] args)
21     {
22     }
23 }

```

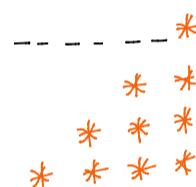
① Complete the Square



$$\begin{array}{l} \text{total no of lines} \\ \text{initial - Space:} \end{array} = 5$$

$$= 4 ((*^2) - 2)$$

$$= 6$$



InvertedRightAngledTriangle.java

```

1 package patternsUsingNestedLoop;
2
3 public class InvertedRightAngledTriangle {
4
5     public static void main(String[] args)
6     {
7         int totalLines = 5;
8         int initialSpace = (totalLines * 2) - 2;
9
10        for (int line = 1; line <= totalLines; line++)
11        {
12            for (int spaces = initialSpace, spaces > 0; spaces--)
13            {
14                System.out.print(" ");
15            }
16            for (int star = 1; star <= line; star++)
17            {
18                System.out.print("* ");
19            }
20
21            initialSpace = initialSpace - 2;
22            System.out.print("\n");
23        }
24    }
25 }

```

Increasing order of spaces allotted to data types

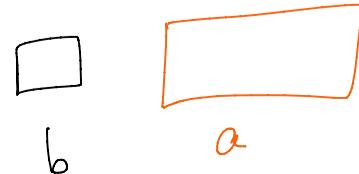
byte < char < short < int < long < float < double

[ASCII]

byte b = 127;

int a = 3;

X b = a



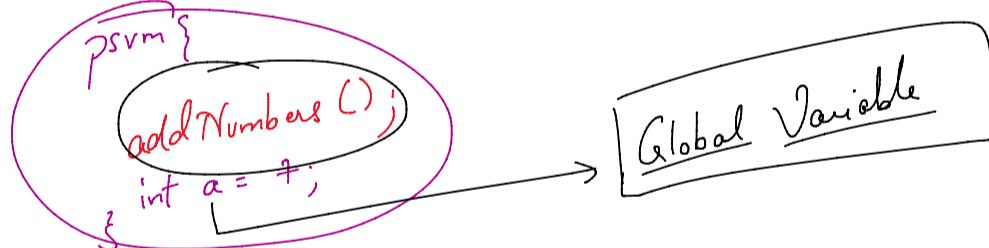
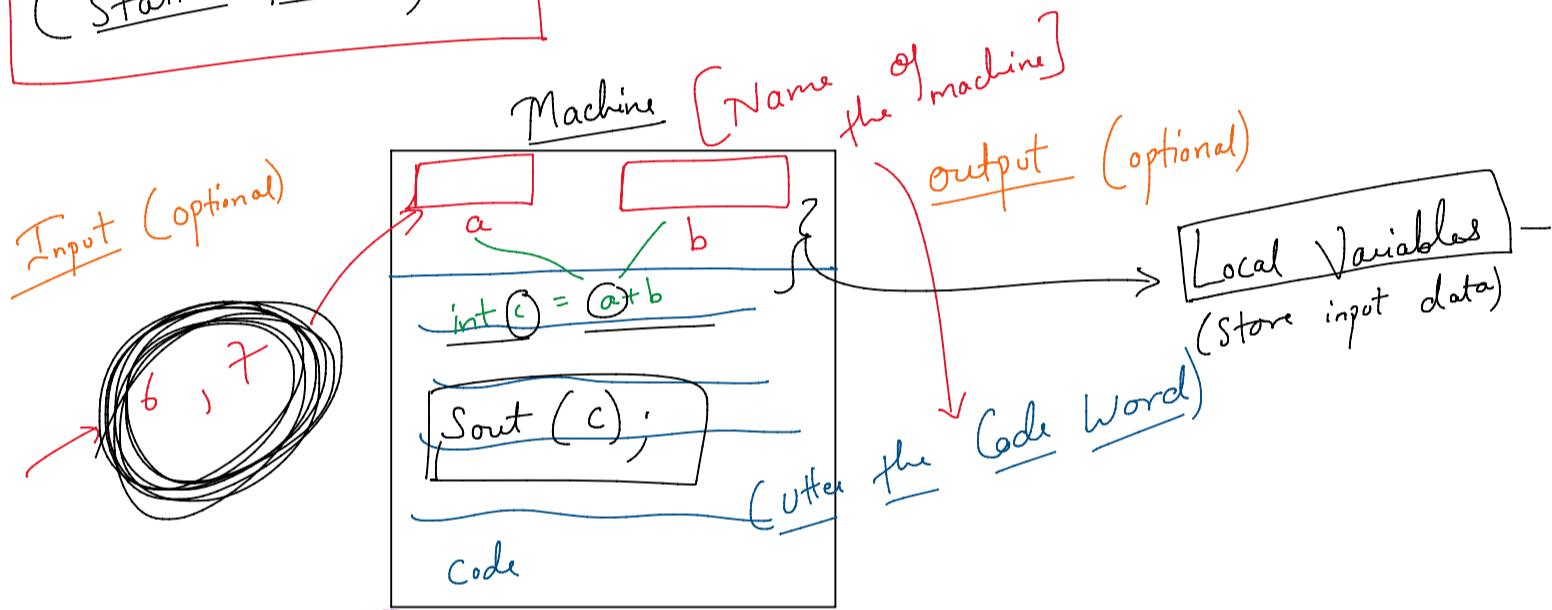
(Widening) type Conversion
(Narrowing) " - " Casting

~~$b = a$~~ b a
 ✓ $b = (\text{byte}) a;$
 trim the data of a
 to fit in the provided space by 'byte' data type

int $a = 7;$
 float $f;$
 $f = a;$ Type Conversion

★ Functions ★

(Static Methods)



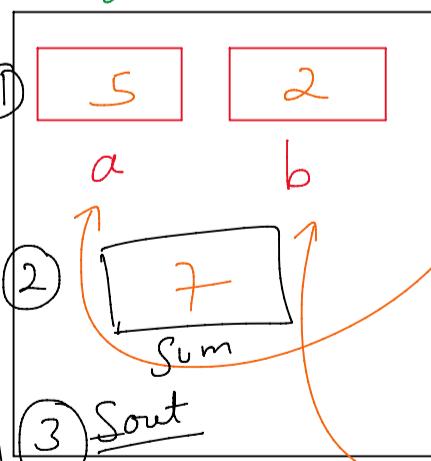
Creating a static method

public static void addNums(int a, int b)

addNums

parameters
int sum = a+b;

Sout ("Addition is :" + sum)



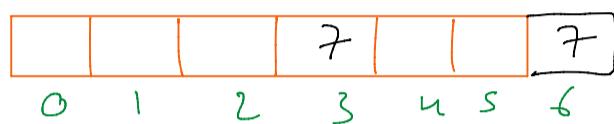
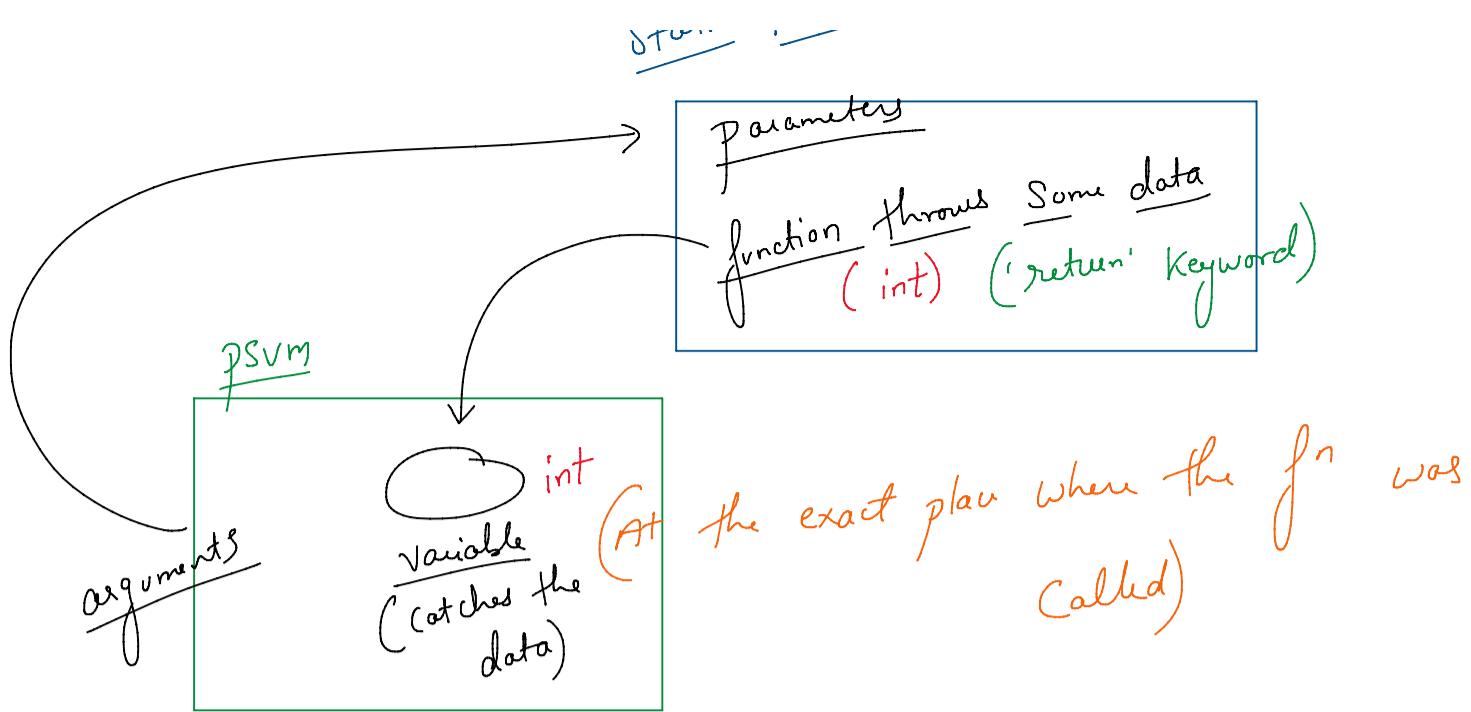
PSVM
addNums(7, 6);

addNums(5, 2);

arguments

get Copied into

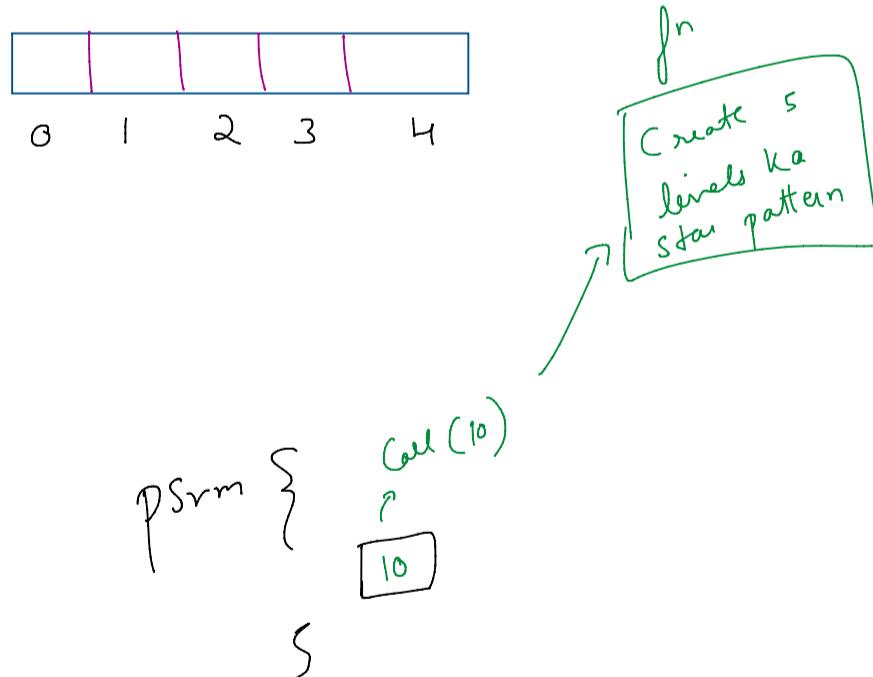
Static Method



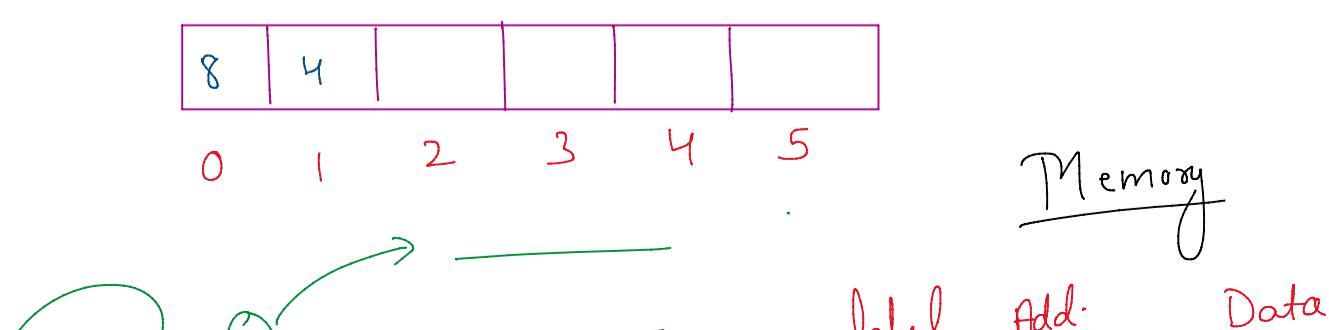
array.push(7)
.push(3, 7);



★ Array ★ (Same data-type)



- ① Elements are stored in consecutive memory location
- ② Each element can be considered as a variable.



int a = 5;
 int b = 7;

label	Add.	Data
a	3001 _H	0000 0000 1 byte
	3002	" — "
	3003	" — "
	3004	0000 0101
	3005	0 0 0 1

1 bit = Smallest unit
 8 bits = 2 Nibbles = 1 byte

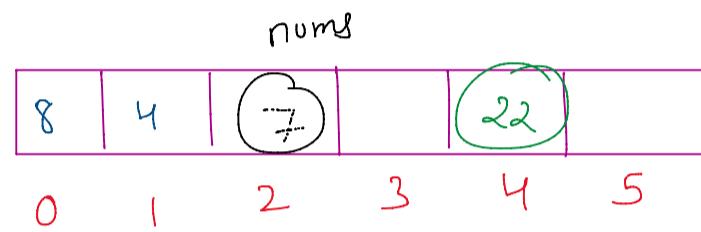
1024 bytes = 1 Kb

1024 Kb = 1 mb

1024 mb = 1 Gb

1024 Gb = 1 Tb

1024 Tb = 1 Pb



name of array
 } indices +
 elements

nums[2] = 7;

nums[4] = 22;

[Arrays have fixed size]

* Creating an Empty Array *

No. of elements that array shall hold.

int [] nums = new int[5];

int [] nums

int nums[] =

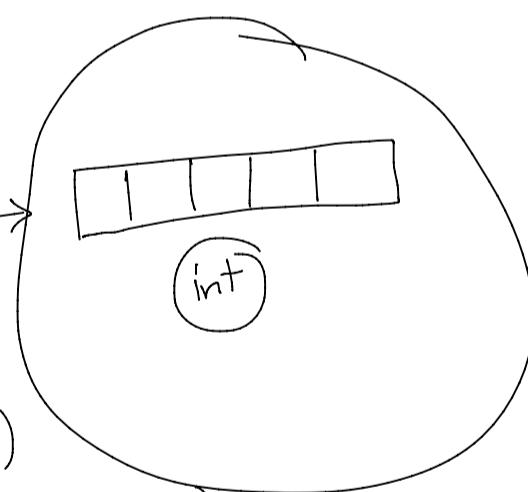
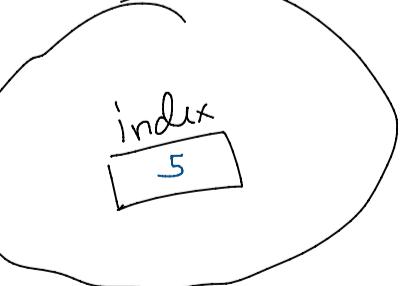
int nums[] =

nums[0] = _____;

nums[1] = _____;

nums[2] = _____;

name of 1st element



SC
 for(int index = 0; index < nums.length; index++)
 {
 cout (" Element Ki Jahan enter karo: ");
 nums[index] = SC.nextInt();

background (nums[4] = 22)

for (

)

)


```

    {
        cout ( 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]

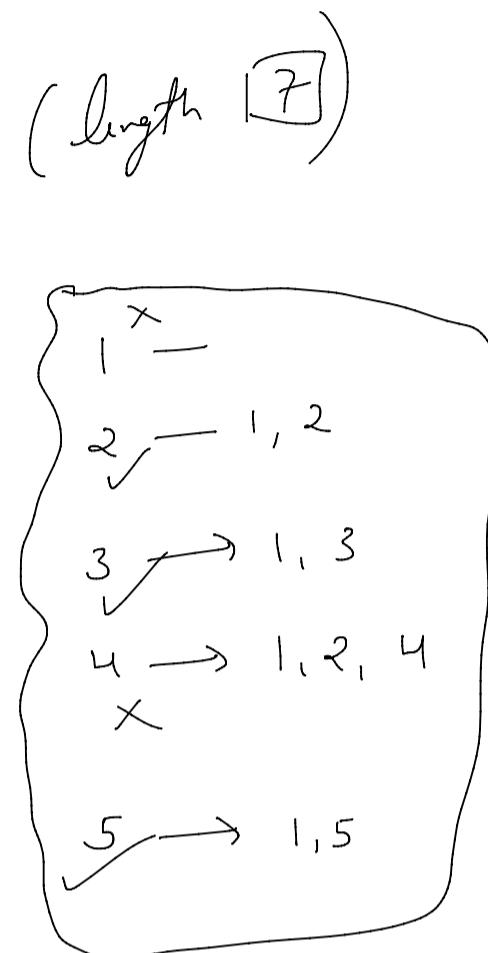
5 | 21 | 7 | 10 | 2

OR

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

i.e 11

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



①

5 | 21 | 7 | 10 | 2

②

Pick out an element & store it into a variable

picked Element
5

③

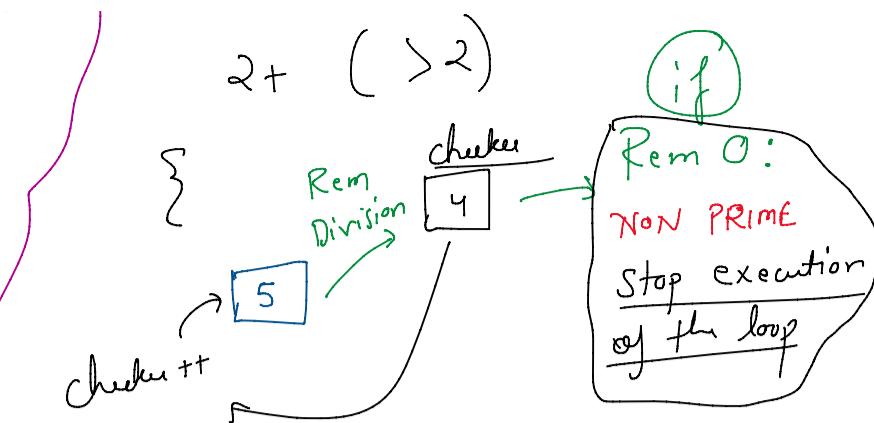
CHECK whether it's prime

1 → cout (Neither prime nor composite)
2 → cout (prime num)
2+ (> 2) if

③ CHECK WHETHER IT IS TRUE

④ Sout(Fact);

2 - (n-1)



The image shows a Java code snippet for a prime number checker. The code uses a for loop to check if a number is divisible by any number from 2 to itself minus one. If it finds a divisor, it sets a status variable to false and breaks out of the loop. If no divisors are found, it sets the status to true. Finally, it prints the result based on the status.

```
else if (pickedNumber == 2)
{
    System.out.println(pickedNumber + " is a Prime number ");
}
else if (pickedNumber > 2)
{
    for (int checker = 2; checker < pickedNumber; checker++)
    {
        if (pickedNumber % checker == 0)
        {
            primeStatus = false;
            break;
        }
        else
        {
            primeStatus = true;
        }
    }
}
if(primeStatus == true)
{
    System.out.println(pickedNumber + " is a PRIME NUMBER");
}
```

Annotations:

- A large circle labeled "pickedNumber" contains the value "8".
- A smaller box labeled "checker" contains the value "5".
- An oval labeled "primeStatus" has a red border and contains the letter "F", indicating the variable's state after the loop.
- A handwritten note "checked" is written near the top right of the code area.

① for (i = 100
 ↓
 pickedNumber

The prime nos in the range of 1 to 100 are

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

Elsp

②

Checking

(mins elements) ()]

2 | 5 | 7 | 8 | 6

2
3
NOT A

(Sum of all the Elements of Array)

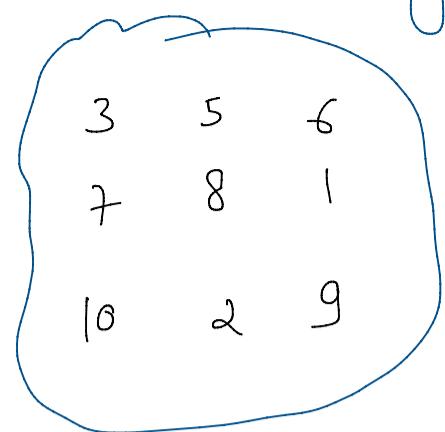
The diagram illustrates a step in a sorting algorithm. An array is shown as a row of five boxes containing the numbers 2, 5, 7, 8, and 6. Brackets are placed over the first four boxes, with the label '(mins elements)' written above them. The fifth box is enclosed in a bracket by itself, with a closing bracket character ')' positioned above it. Above the array, the word 'Checking' is written in cursive. To the right of the array, there is a red annotation: the number '2' is connected by a red line to the first box; the number '3' is connected by a red line to the second box; and the text 'NOT A' is written in red, also connected by a red line.

28

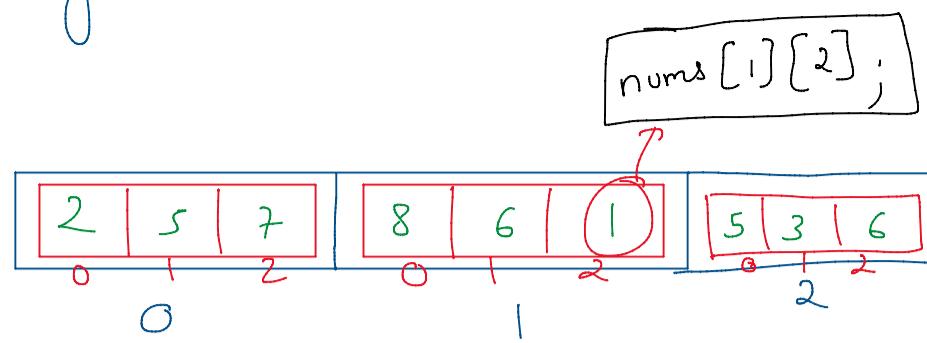
Sum

★ 2 D Array Array inside another Array.

U Array inside another Array



nums



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

[0][0]

[0][1]

[0][2]

[1][0]

[1][1]

[1][2]

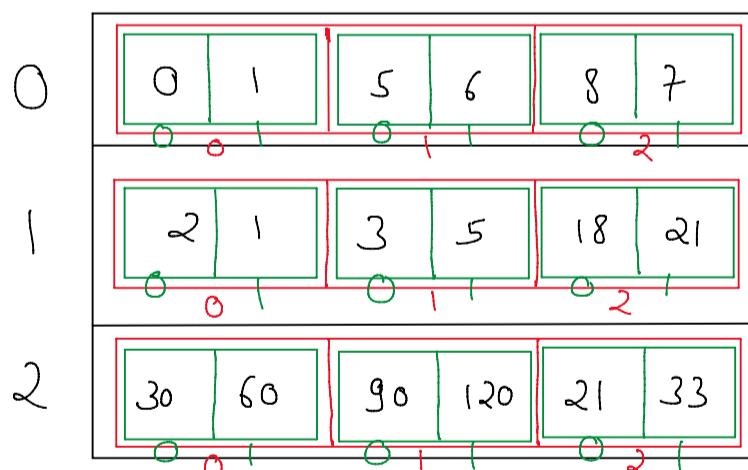
[2][0]

[2][1]

[2][2]

★ 3D Array ★

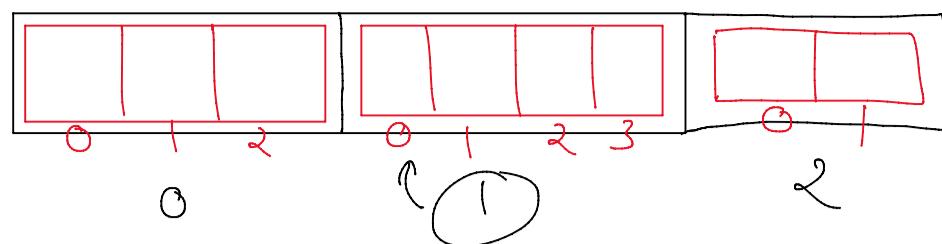
Array inside an Array inside an Array



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)

Declaration : `ArrayList < wrapper class > name = new ArrayList<>();`

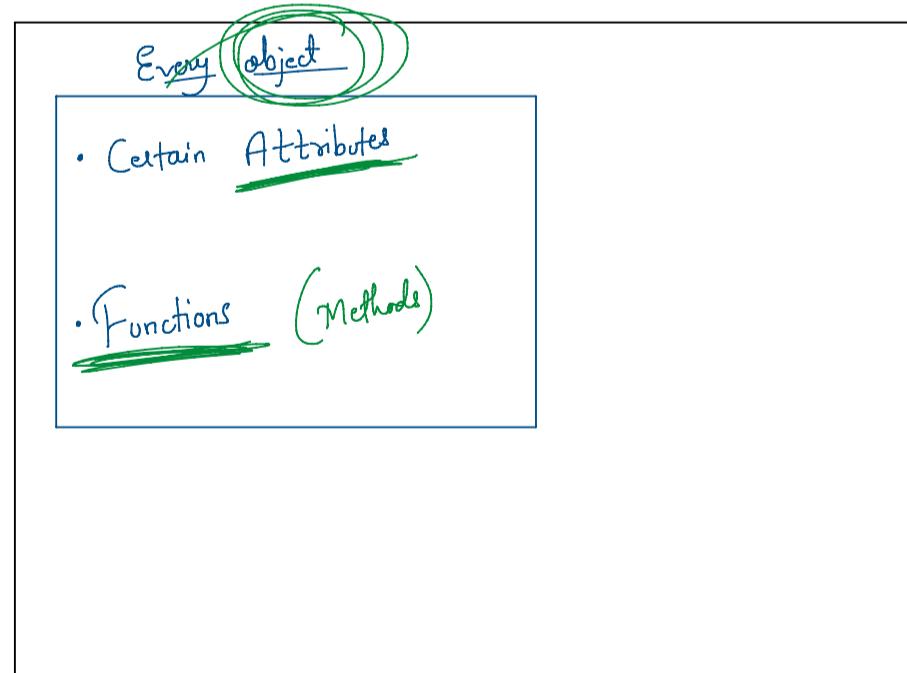
Methods (Predefined):

- ① add
- ② get
- ③ remove
- ④ size (length of ArrayList);
- ⑤ Set

* OOPS *

(Solve: Real Life Problems - Virtually)

• Plan → (class)
• object → (object)



Plan (class) Cars

Attributes

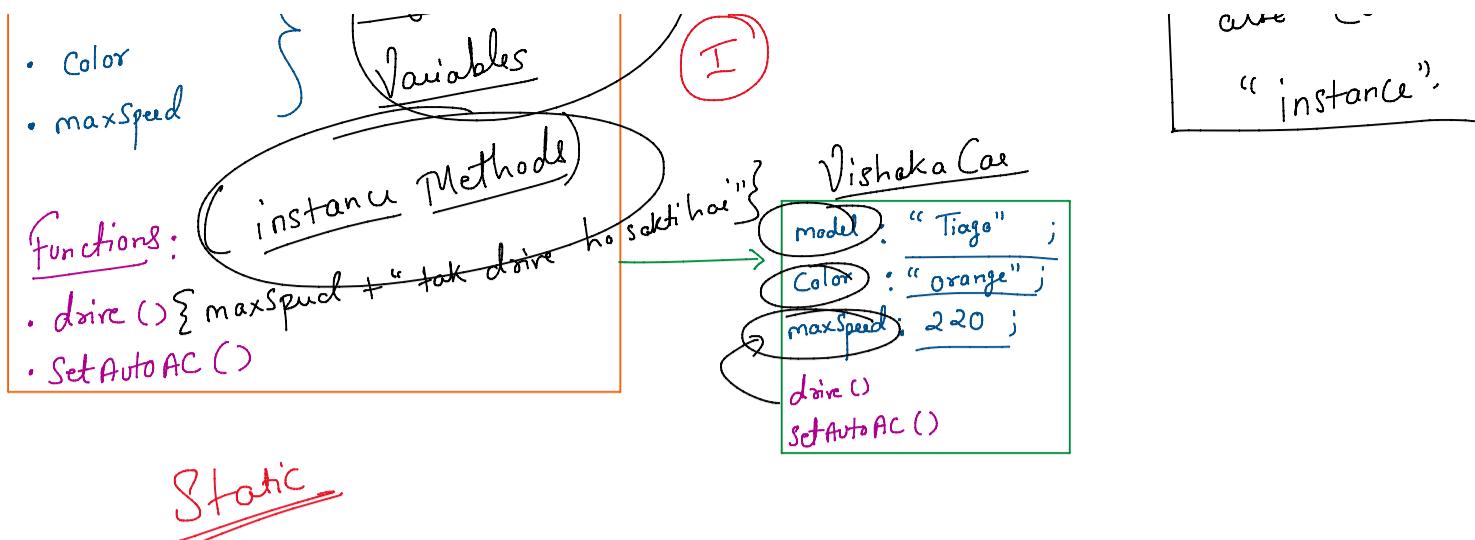
- model
- color
- ...

{
object instance
Variables

RahulCar

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

An object is
also called
"instance".



```

CreatingObjects.java 1 ●
src > OOPS > ObjectsAndClasses > CreatingObjects.java > main(String[])
package OOPS.ObjectsAndClasses;

class Cars
{
    String model;
    String color;
    int maxSpeed;
}

public class CreatingObjects
{
    public static void main(String[] args)
    {
        Cars VishakaCar = new Cars();
    }
}

```

VishakaCar

model : "HondaCity"
color : null
maxSpeed : 0

VishakaCar.model = "HondaCity";

```

CreatingObjects.java X
src > OOPS > ObjectsAndClasses > CreatingObjects.java > Cars > showDetails()
1 package OOPS.ObjectsAndClasses;
2
3 class Cars
4 {
5     String model;
6     String color;
7     int maxSpeed;
8
9     public void showDetails()
10    {
11        System.out.println(this.model);
12        System.out.println(this.color);
13        System.out.println(this.maxSpeed);
14    }
15
16 public class CreatingObjects
17 {
18     public static void main(String[] args)
19     {
20         Cars VishakaCar = new Cars();
21
22         VishakaCar.model = "Tiago";
23         VishakaCar.color = "Orange";
24         VishakaCar.maxSpeed = 220;
25
26         Cars RahulCar = new Cars();
27     }
}

```

VishakaCar

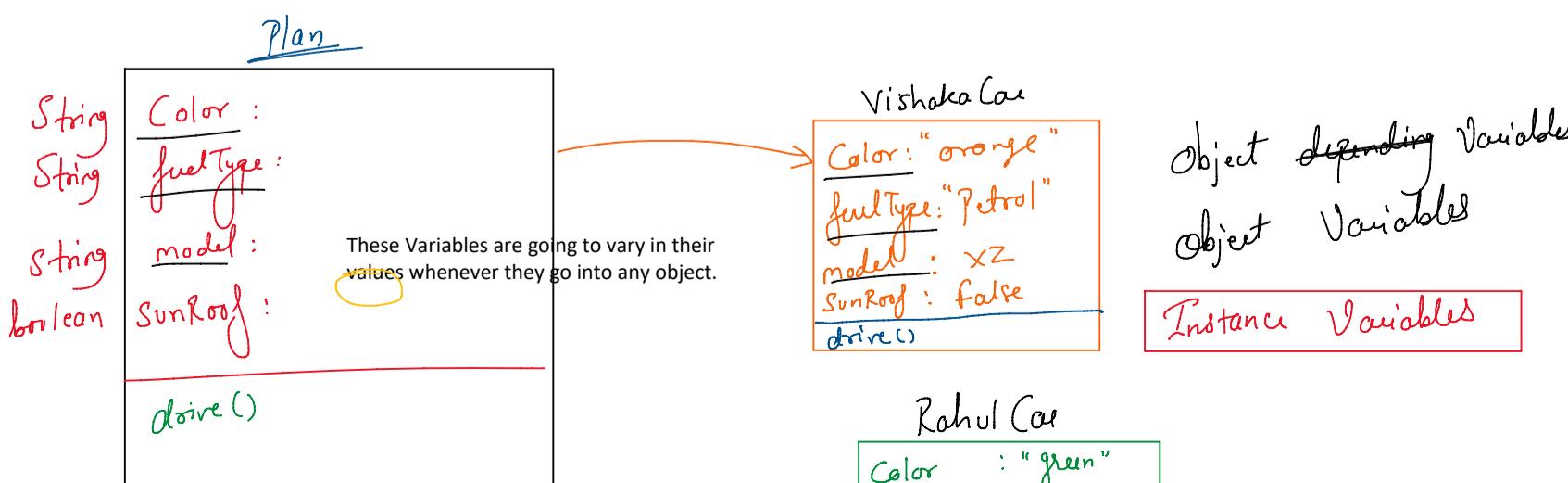
WHAT

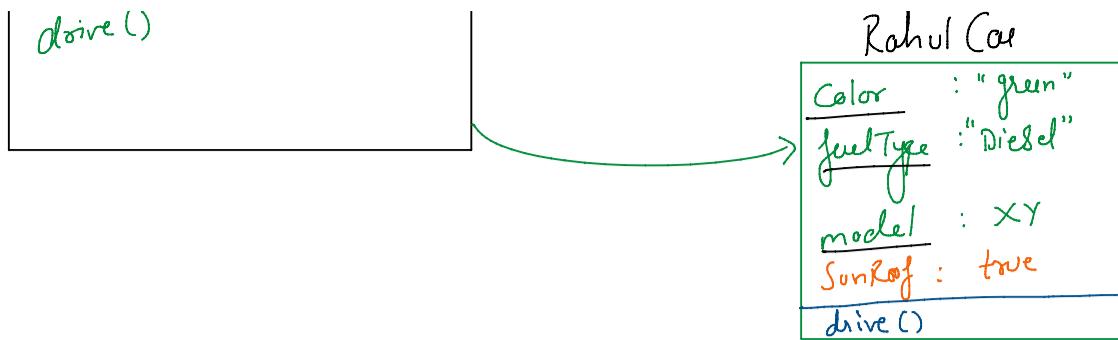
model : ("Tiago");
color : ("orange");
maxSpeed : 220;
showDetails()

RahulCar

VishakaCar.showDetails()

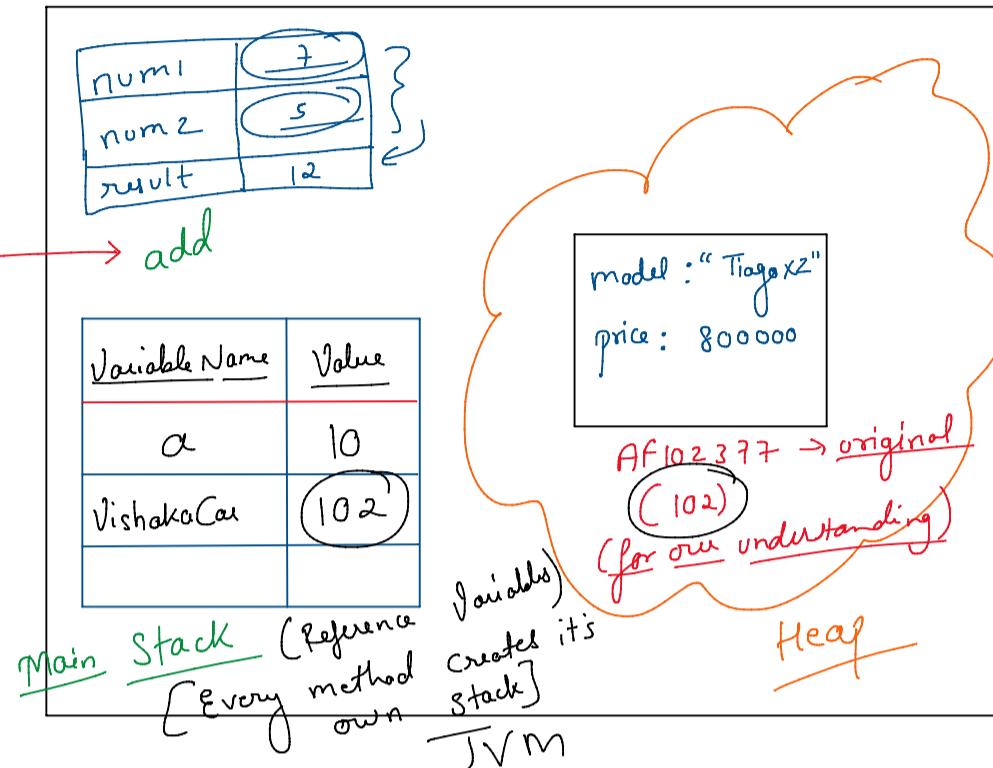
model : "Tiago";
color : "Orange";
maxSpeed : 220;
showDetails()





★ Heap & Stack ★

```
Example.java 1 X
src > OOPS > stackAndheap > Example.java > Example > main(String[])
1 package OOPS.stackAndheap;
2
3 class Cars
4 {
5     String model;
6     float price; int num1, num2
7 } ps void add(num1, num2)
8 public class Example
9 {
10
11     Run | Debug
12     public static void main(String[] args)
13     {
14         int a = 10;
15
16         Cars VishakaCar = new Cars();
17         VishakaCar.model = "Tiago XZ";
18         VishakaCar.price = 8.00_000f;
19         System.out.println(VishakaCar.price);
20     }
21 }
```

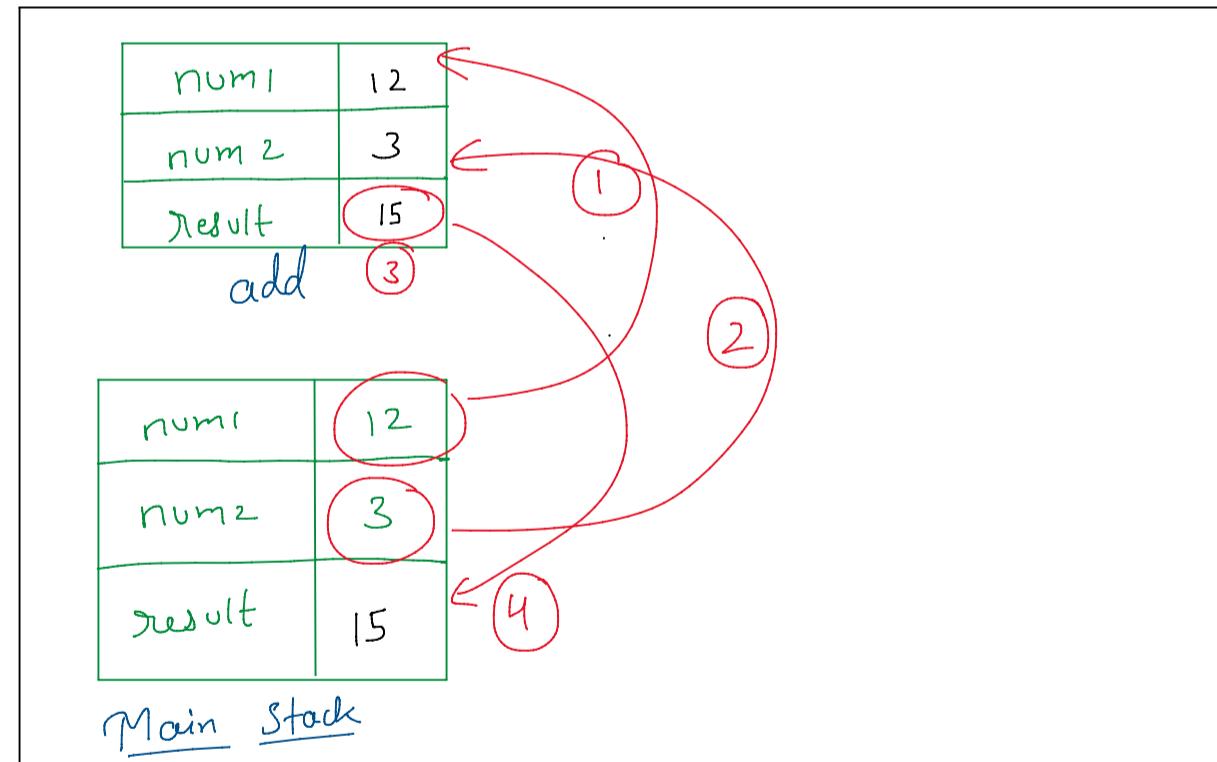


Stack memory is the only memory with which the execution engine can interact. That means our execution engine cannot interact with heap. So stack is a way which creates a link between heap and itself.

```
TwentyDecember.java X
src > OOPS > stackAndheap > TwentyDecember.java > TwentyDecember > main(String[])
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);
    }
}
```

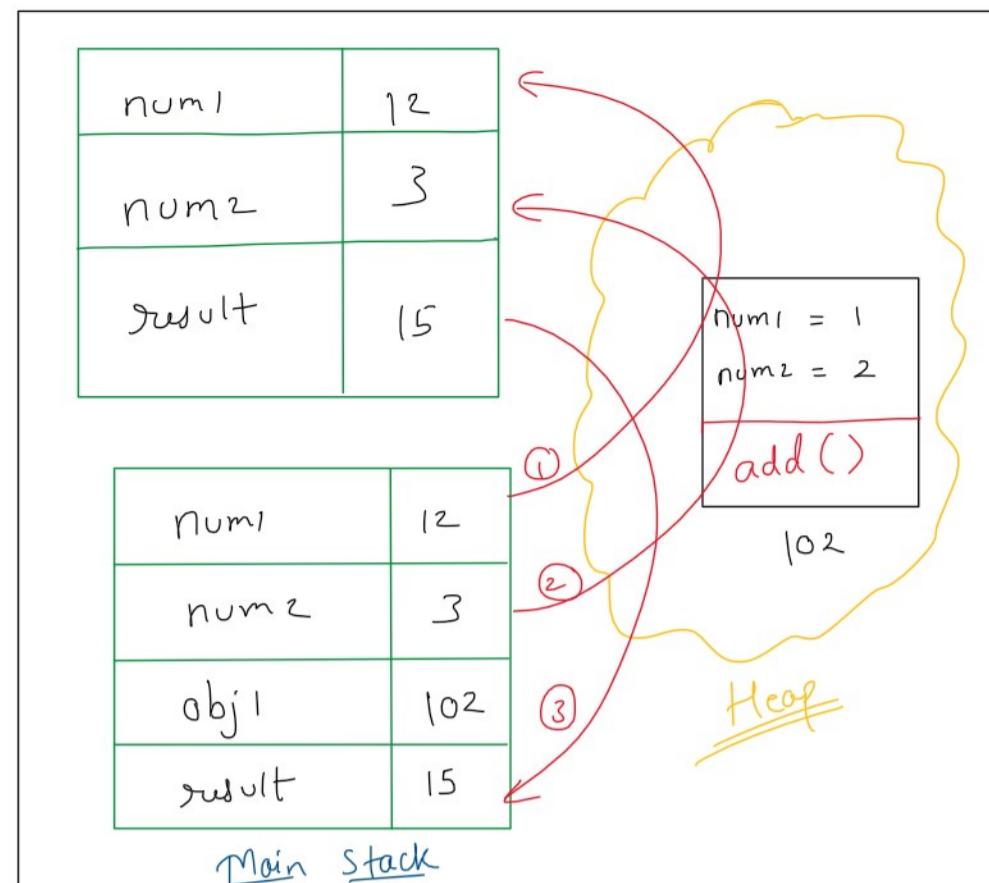


Each stack can have variables of unique names
Only


```

TwentyDecember2.java X
src > OOPS > stackAndheap > TwentyDecember2.java > ...
1 package oops.stackAndheap;
2
3 class ArithmeticCalc
4 {
5     int num1;
6     int num2;
7
8     public ArithmeticCalc(int NUM1, int NUM2)
9     {
10        this.num1 = NUM1;
11        this.num2 = NUM2;
12    }
13
14     public int add(int num1, int num2)
15     {
16        int result = num1 + num2;
17        return result;
18    }
19
20 }
21
22 public class TwentyDecember2
23 {
24     Run | Debug
25     public static void main(String[] args)
26     {
27         int num1 = 12;
28         int num2 = 3;
29
30         ArithmeticCalc obj1 = new ArithmeticCalc(1, 2);
31
32         int result = obj1.add(num1, num2);
33         System.out.println(result);
34     }
35 }

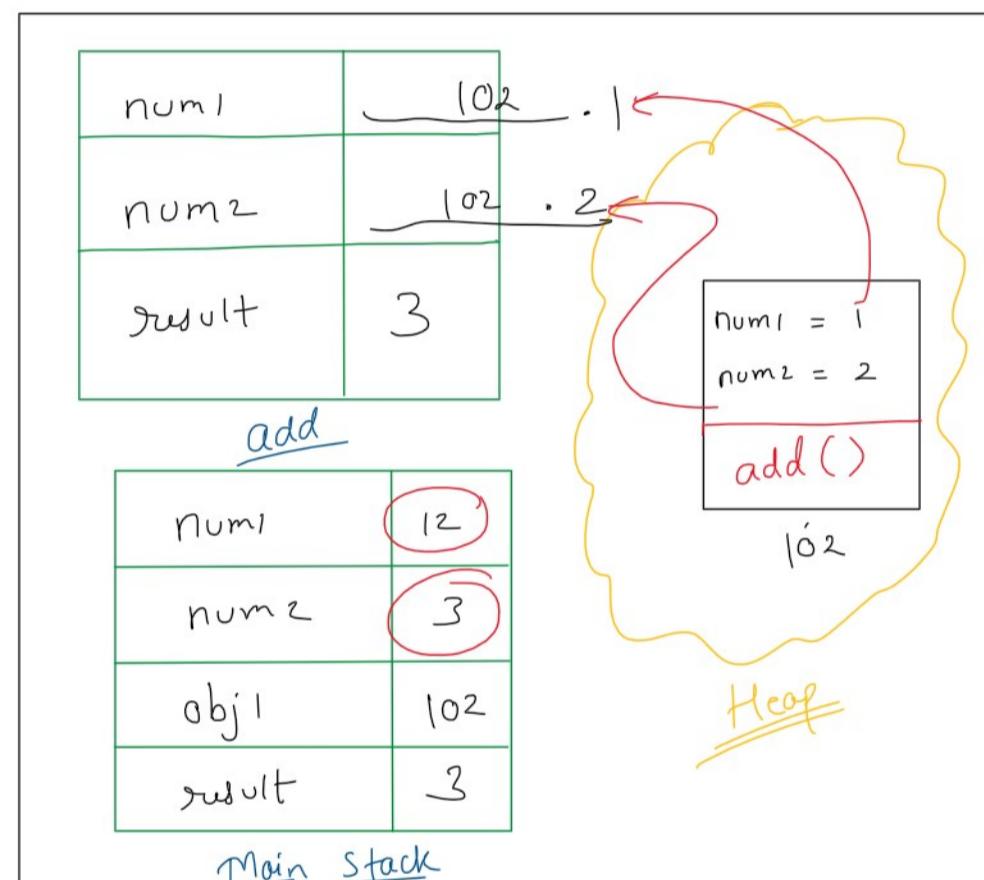
```



```

TwentyDecember2.java X
src > OOPS > stackAndheap > TwentyDecember2.java > ArithmeticCalc > ...
1 package oops.stackAndheap;
2
3 class ArithmeticCalc
4 {
5     int num1;
6     int num2;
7
8     public ArithmeticCalc(int NUM1, int NUM2)
9     {
10        this.num1 = NUM1;
11        this.num2 = NUM2;
12    }
13
14     public int add(int num1, int num2)
15     {
16        num1 = this.num1;
17        num2 = this.num2;
18        int result = num1 + num2;
19        return result;
20    }
21
22 }
23
24 public class TwentyDecember2
25 {
26     Run | Debug
27     public static void main(String[] args)
28     {
29         int num1 = 12;
30         int num2 = 3;
31
32         ArithmeticCalc obj1 = new ArithmeticCalc(1, 2);
33
34         int result = obj1.add(num1, num2);
35         System.out.println(result);
36     }
37 }

```



★ 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 = "RATHUL"
 ↓↓↓↓
 0 1 2 3 4

System.out.println(str.charAt(3));

str.length() - 1
 0 1 ... (str.length() - 1)


```
Sout(str.charAt(3));
```

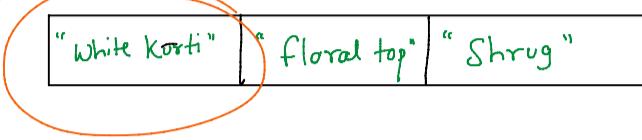
name = "VISHAKA"

0 1 2 3 4 5 6

revStr = " "; AKAH SIV

for (int index=7; index >=0; index--) {
 revStr = revStr + str.charAt(index)
 AK }

Q-1



"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 = C + ":" + o
outfits.add();

Q-2

"RAHUL" | "VISHAKA" | "SNEHA"

Var

0

2

named

"LUHAR" | "AKAH SIV" | "AHENS"

0

1

2

names

name = "Rahul"

name = "Rahul" + " Trivedi";

fName = "vishaka";

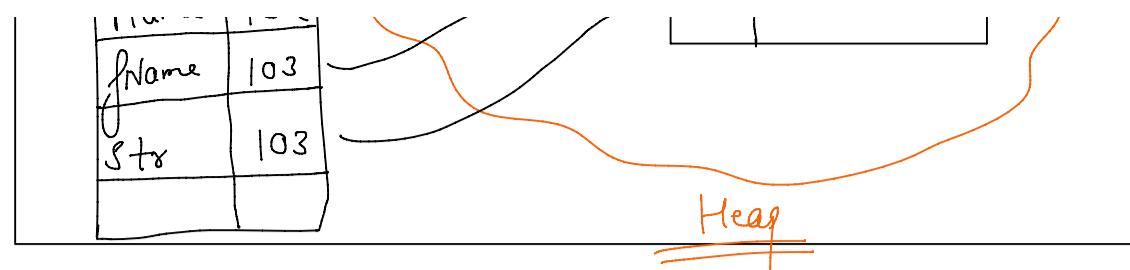
str = "vishaka";

String Constant Pool

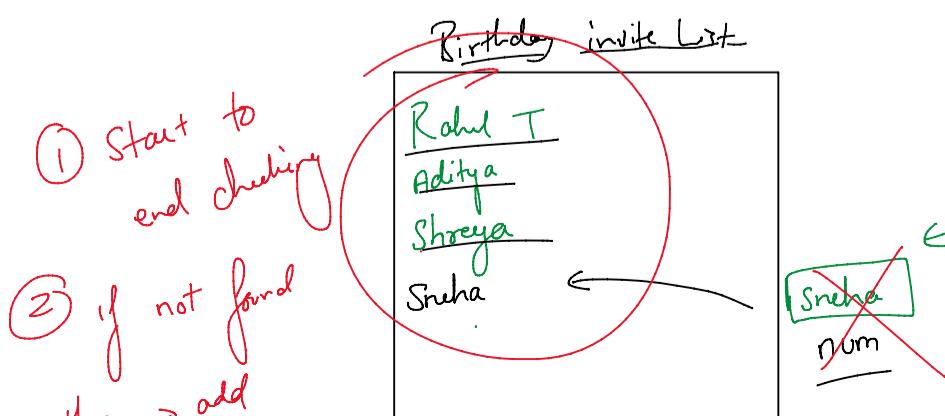
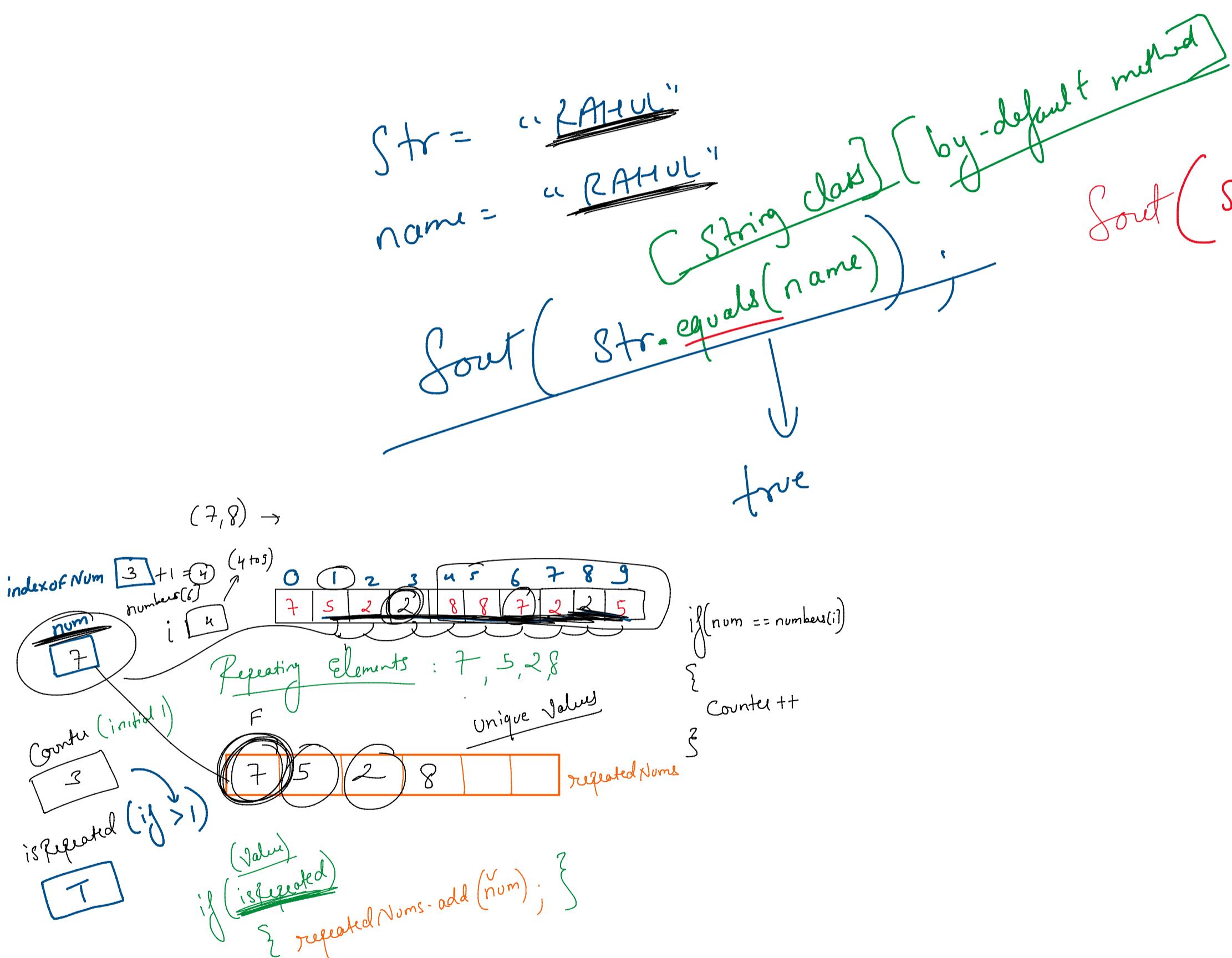
name	102
fName	103

101	"Rahul"
102	"Rahul Trivedi"
103	"vishaka"

`str = "vishaka";`



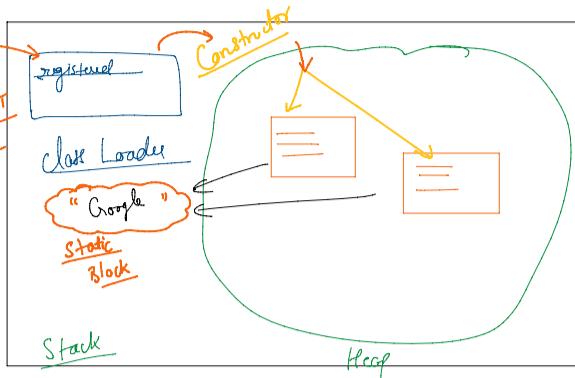
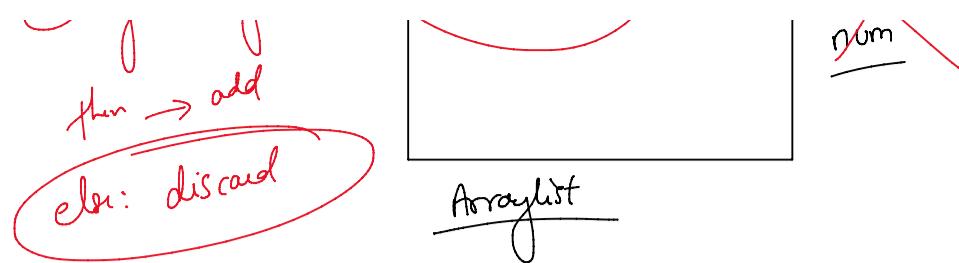
Whenever We make any updatations in the existing string object, Another object gets created with the updated value but the original object stays unaffected. Hence, it is said that strings in Java are immutable because the original object is not experiencing any mutation.



(tr == name)

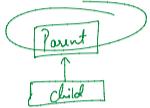


false

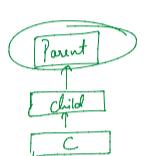


Types of Inheritance in Java :

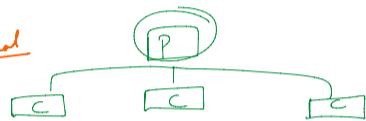
① Single



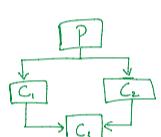
② Multilevel :



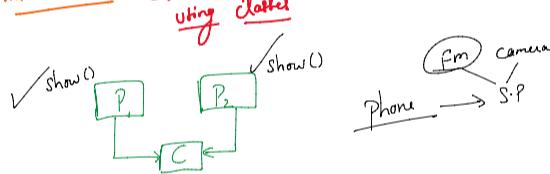
③ Hierarchical



④ Hybrid



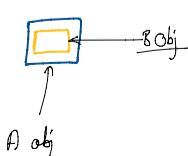
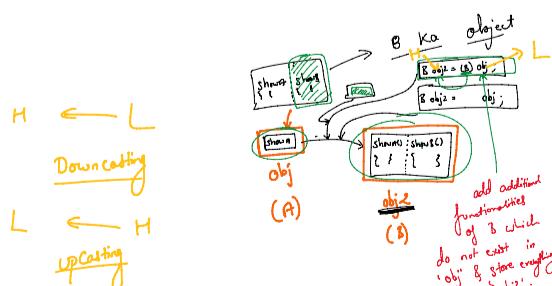
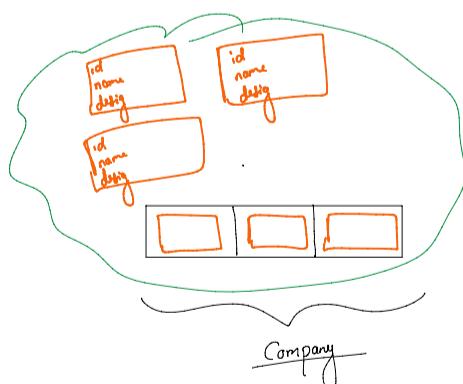
⑤ Multiple inheritance (Not Supported using classes)



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




```

class Laptop {
    public void runTheCode() {
        System.out.println("code, compile & run if all OK !!!");
    }
}

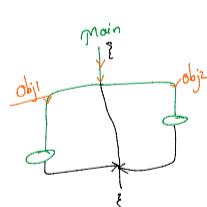
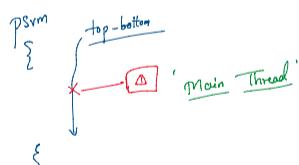
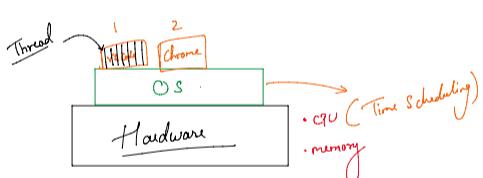
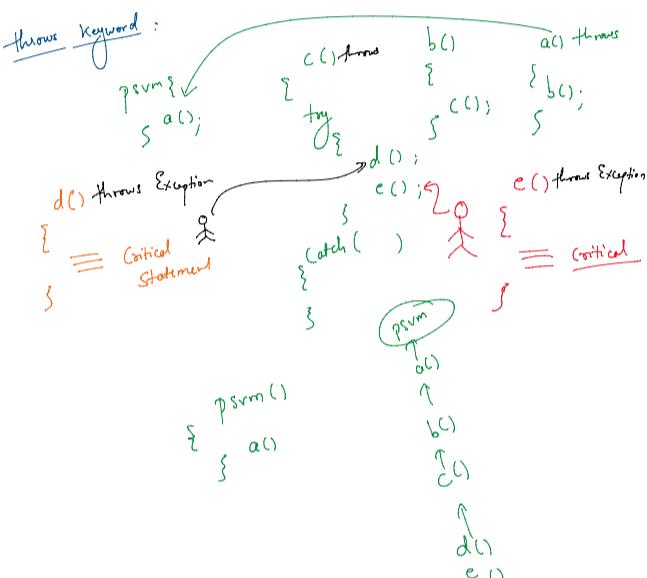
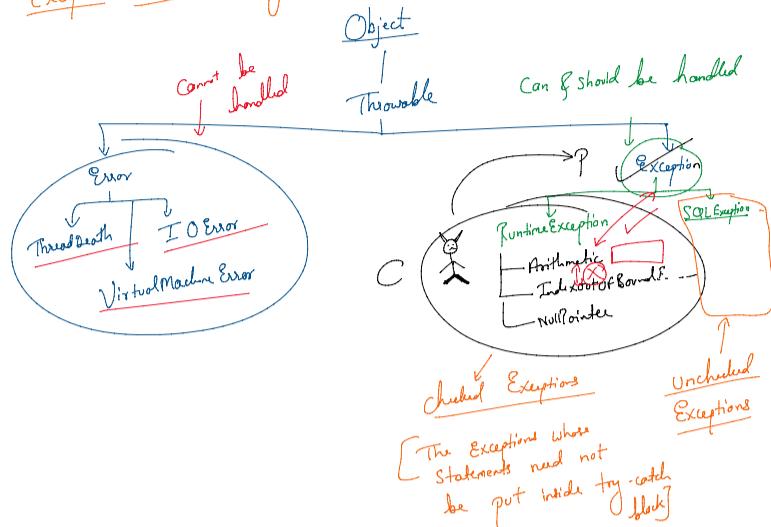
class Developer {
    public void developApp(Laptop obj) {
        System.out.println("Developing an app....");
        obj.runTheCode();
    }
}

public class Example1 {
    public static void main(String[] args) {
        Developer Rahul = new Developer();
        Laptop HPOpen15 = new Laptop();
        Rahul.developApp(HPOpen15);
    }
}

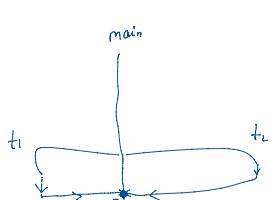
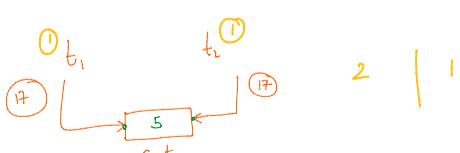
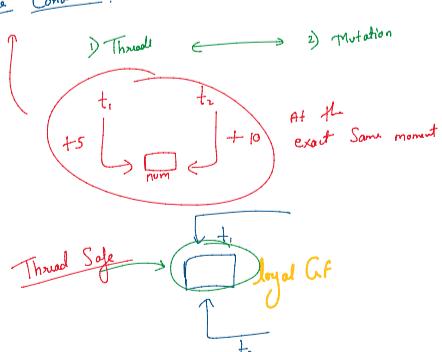
```

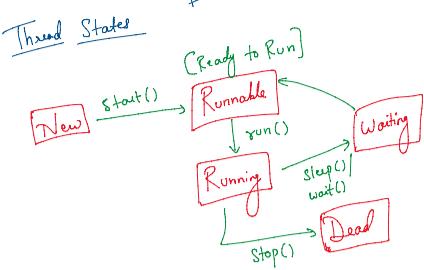
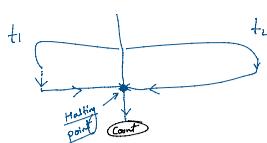
All the classes in Java are child classes of the 'Object' class

Exception class Hierarchy

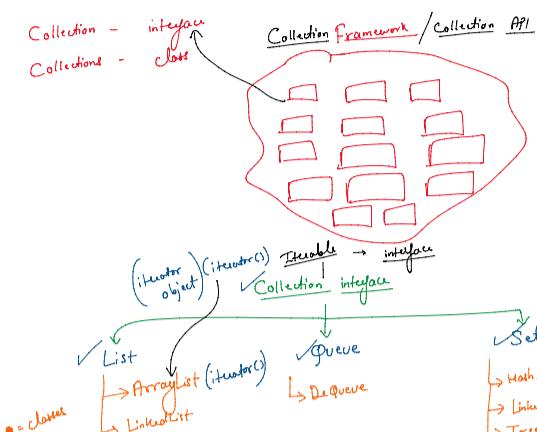


Race Condition:

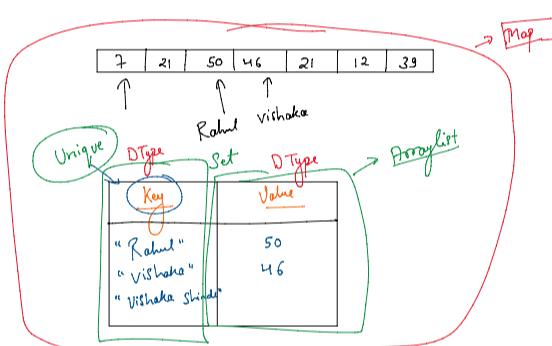
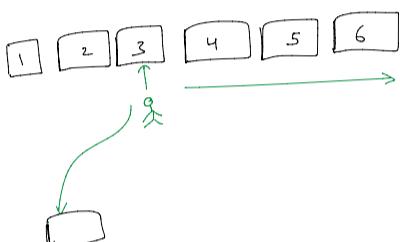




Collection - interface
Collections - class



Map



[7, 2, 5, 12, 24, 36]

