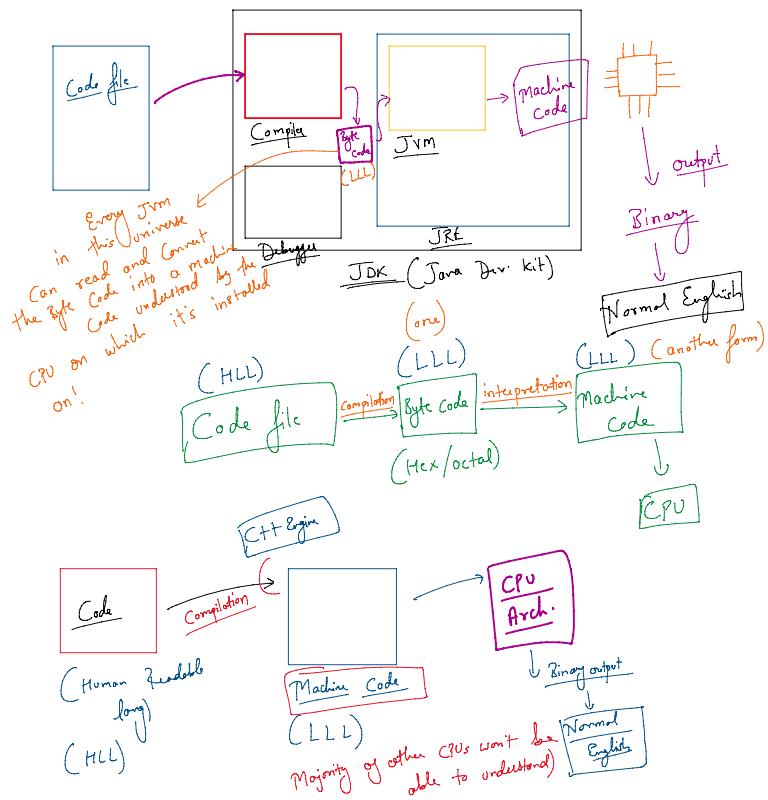
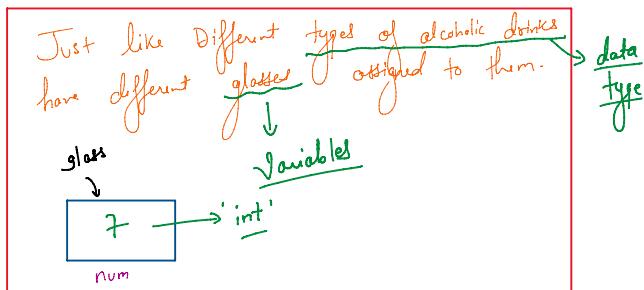


★ How Java Code Runs? ★



★ Variables & Data Types ★

Containers



★ Creating a Variable ★

Declaring a Variable : type <Var_name>

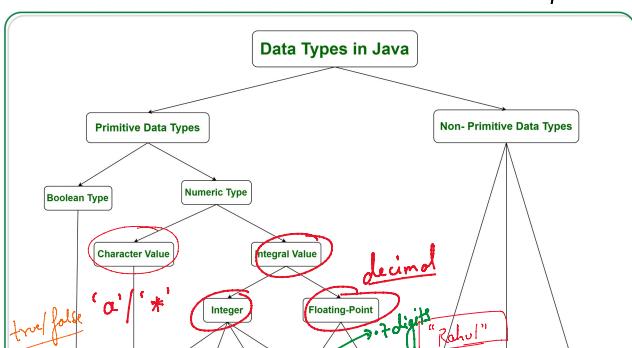
int num;

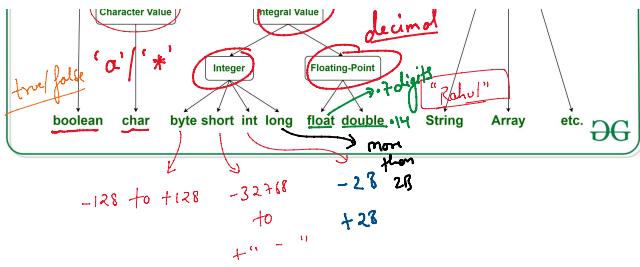
num

★ Putting Data into a Variable ★ (Initializing a Variable)

num = 7;

7





byte

```

id
|
1
2
:
10

```

float num = 7.2f;

double num = 7.25d;

long num = 25L;

String name = "Rahul";

char ch = '/';

char ch = ' ';

(Space is also a character)

int num = 7;

String name = "Rahul";

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

Number : 7

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

★ Output Method ★

(quotient) (Remainder)

① Arithmetic : +, -, /, %, *, ++, --

double a = 5;

double b = 2;

double quotient = 5 / 2;

double rem = 5 % 2

int num = -2;

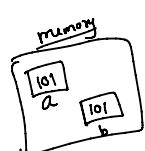
$$\begin{array}{r} 2 \text{ --- quotient} \\ 2 \sqrt{5} \\ \underline{-4} \\ 1 \text{ --- rem} \end{array}$$

++ (Increment)

pre-increment

a = 100;

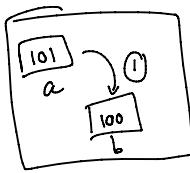
b = a + a



(Left most)

post-increment

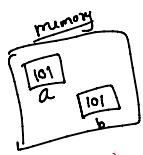
a = 100;
b = a++;



pre-increment

$$a = 100;$$

$$b = \textcircled{1} + a$$

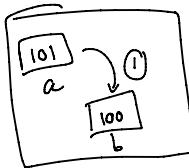


(Left most)

post-increment

$$a = 100;$$

$$b = a \textcircled{1} +$$



Output: $a \rightarrow 101$
 $b \rightarrow 101$

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

② Conditional: $<$, $>$, \geq , \leq , $!=$, $==$ (Exactly Equal to?)

③ Assignment: $=$, $+=$, $-=$, $*=$, $/=$, $\%=$

$$a \textcircled{1} = b$$

$$a = a \textcircled{1} + b$$

④ Logical: AND, NOT, OR

$\&\&$, $!$, $\|$

$$\begin{array}{cc} a & b \\ \checkmark & \checkmark \\ T & T \end{array} \rightarrow T$$

double marks = 75;
if (marks > 55 $\&\&$ marks < 80)

{ Sout("Grade B");

}

int a = 7; $\&$, $!$, $<<$, \gg

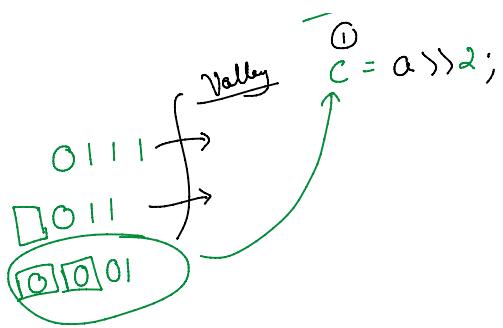
int b = 3;

int c = a $\&$ b;

$$\begin{array}{r} 0111 \\ 0011 \\ \hline 0011 \end{array}$$

Decimal $\rightarrow 3$

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

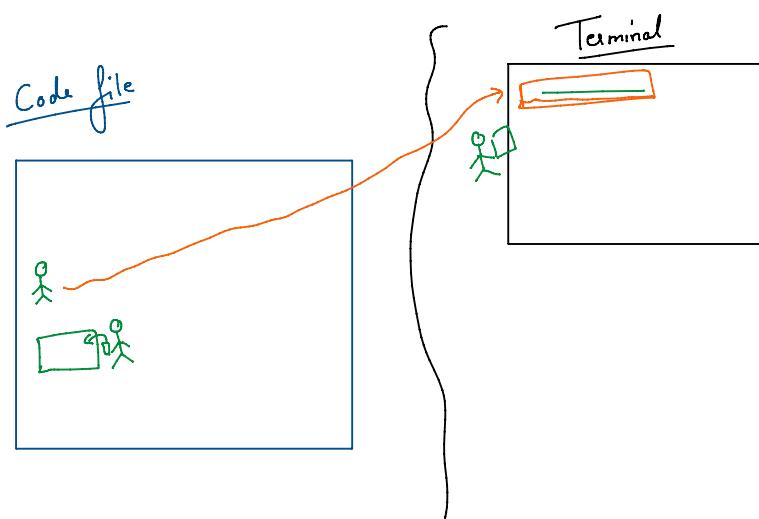


File Name - Class Name

Class Name - Pascal Convention

Variable/function/object Names - Camel Case

Every folder Containing Java files is called 'A Package'



```

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

public class Example
{
    public static void main(String[] args) f
    {
        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");
    }
}
  
```

Annotations:

- Circled 'f' over the `main()` method.
- Circled '2' over the variable `outerVar`.
- Circled '2' over the variable `innerVar`.
- A box labeled **OUTPUT** contains the printed results:

outerVar	innerVar
1	1
1	2
1	3
1	4
2	1
2	2

: 2 : 2

2 3
2 4

3 1
3 2
3 3
3 4

End of program

File Edit Selection View Go ... ← → ⌘ Core Java

RightAngledTriangle.java

```
src > patternsUsingNestedLoop > RightAngledTriangle.java > RightAngledTriangle
```

1 package patternsUsingNestedLoop;
2
3 /*
4 * Expected Output :
5 *
6 * * | 1 | 1 (1)
7 * * * | 2 | 2 (1,2)
8 * * * * | 3 | 3 (1,2,3)
9 * * * * * | 4 | 4 (1,2,3,4)
10 */
11
12
13
14
15
16
17 public class RightAngledTriangle {
18 public static void main(String[] args) {
19 Run | Debug
20 public static void main(String[] args)
21 {
22 }
23 }
24 }

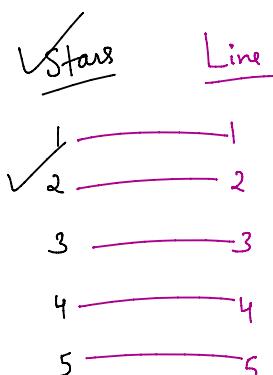
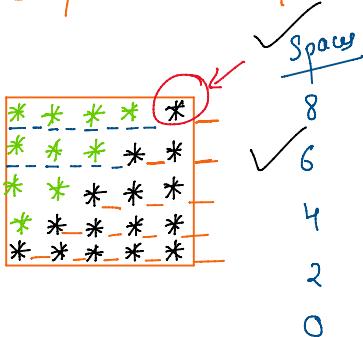
line: stars:
1 (1)
2 (1,2)
3 (1,2,3)
4 (1,2,3,4)
5 (1,2,3,4,5)

Q1

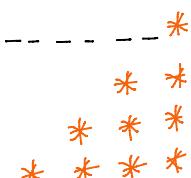
1 2 3 4 5	5 4 3 2 1
1 2 3 4	5 4 3 2
1 2 3	5 4 3
1 2	5 4
1	5

Ln 6, Col 29, Spaces: 4, UU, CRLF, {}, java, Go Live, Idle: ready

① Complete the Square



total no of lines - 4 $((*^2) - 2)$
initial_Space: - 6



不
* * * *
* * * *

```

RightAngledTriangle.java   InvertedRightAngledTriangle.java X
Core Java
Run | Debug
public static void main(String[] args)
{
    int totalLines = 5;
    int initialSpace = (totalLines * 2) - 2;

    for (int line = 1; line <= totalLines; line++)
    {
        for (int spaces = initialSpace; spaces > 0; spaces--)
        {
            System.out.print(" ");
        }
        for (int star = 1; star <= line; star++)
        {
            System.out.print("* ");
        }

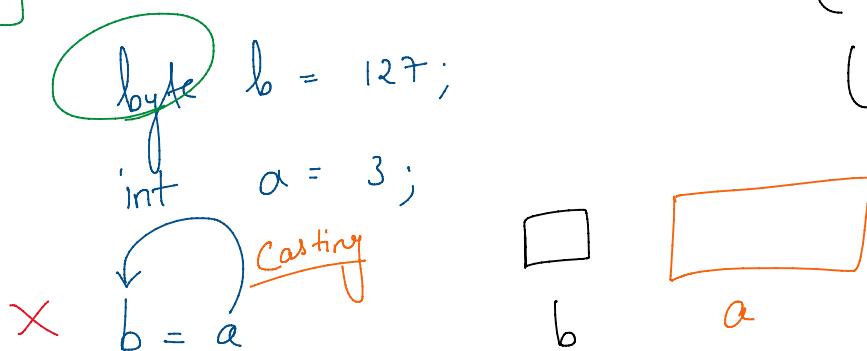
        initialSpace = initialSpace - 2;
        System.out.println("\n");
    }
}

```

Increasing order of spaces allotted to data types

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

[ASCII]



✓ **b = (byte)a;**

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

int a = 7;

(Widening) type conversion
(Narrowing) "—" casting


```
int a = 7;
```

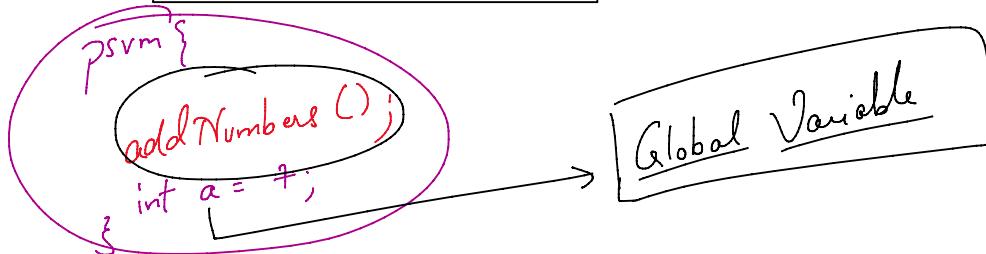
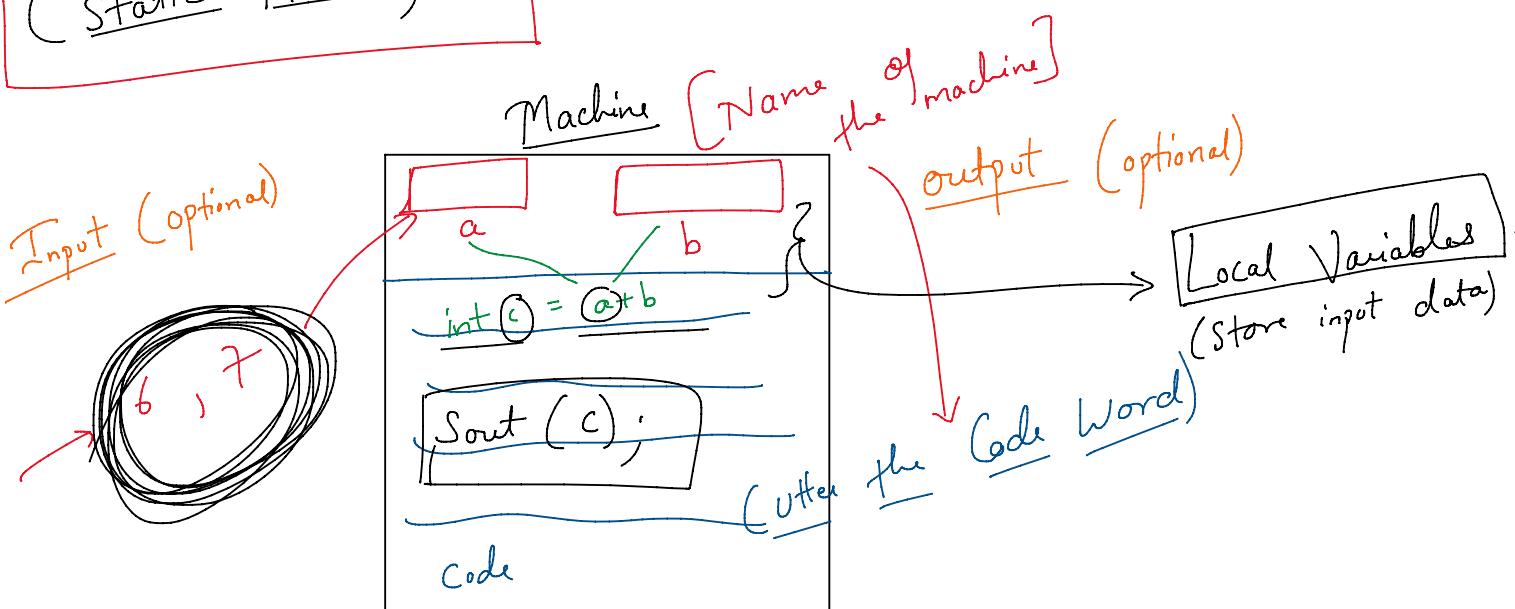
```
float f;
```

```
f = a;
```

Type Conversion

★ Functions ★

(Static Methods)



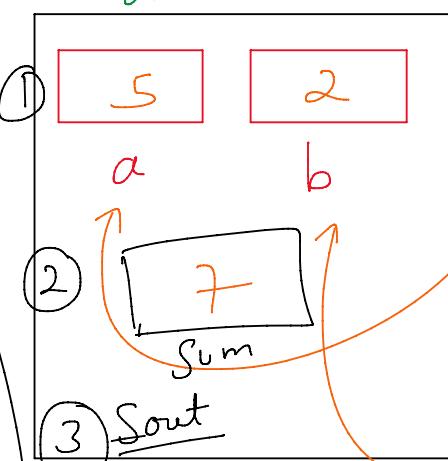
Creating a static Method

```
public static void addNums(int a, int b)
```

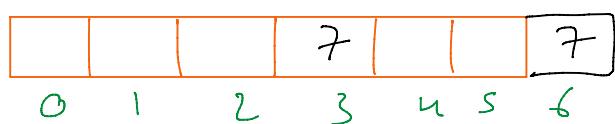
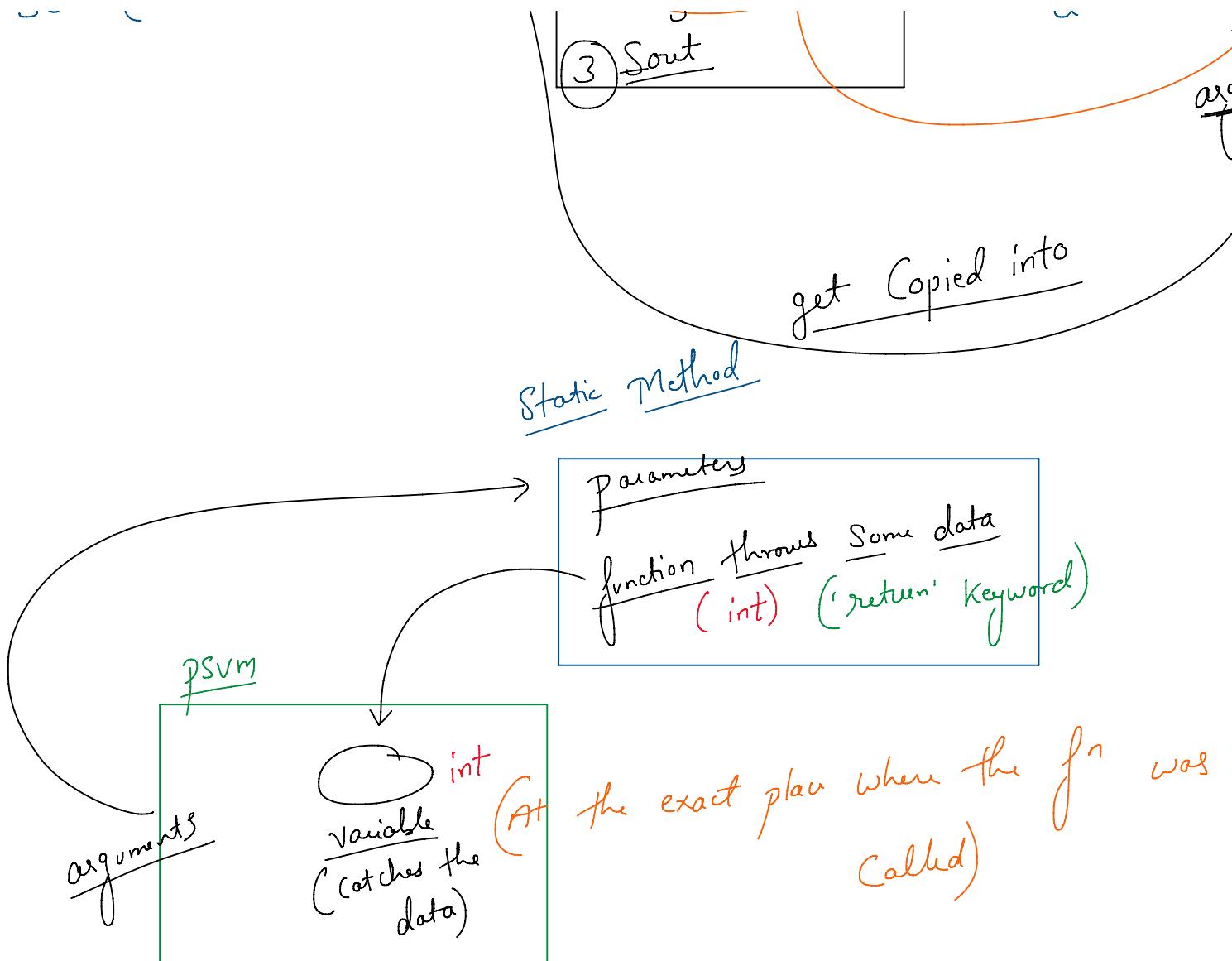
addNums

```
int sum = a + b;
```

```
Sout("Addition is :" + sum);
```



$(7, 6)$;
 $(5, 2)$;
...ments

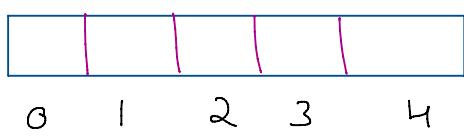


array.push (7)
 .push (3, 7);
 ↴

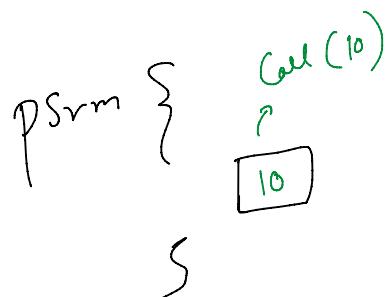


ument's

★ Array ★ (Same data-type)

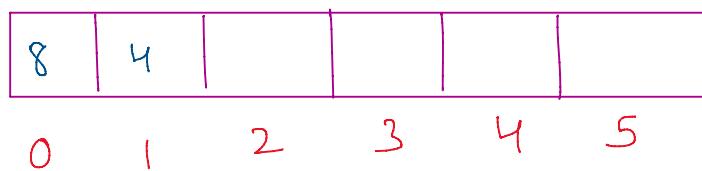


n
Create 5
levels ka
star pattern

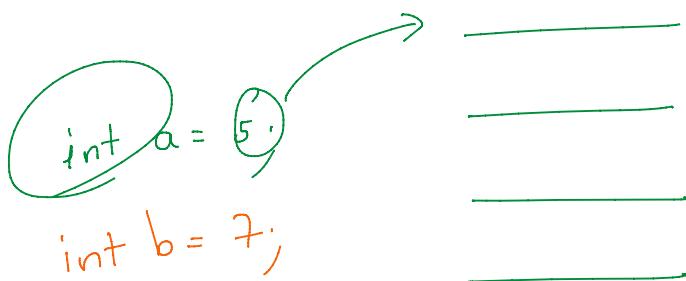


★ Arrays ★

- ① Elements are stored in Consecutive memory
- ② Each element can be considered as a



Memory



label
a

Add.	D
3001 _H	000
3002	"
3003	"
3004	000
3005	"

1 bit = Smallest unit

8 bits = 2 Nibbles = 1 byte

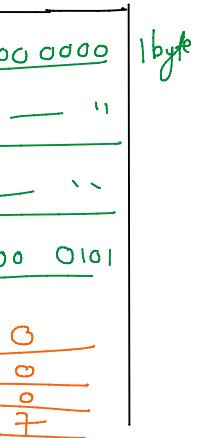
1024 bytes = 1 Kb

1024 Kb = 1 mb

1 - 1 Gb

location
variable.

Data

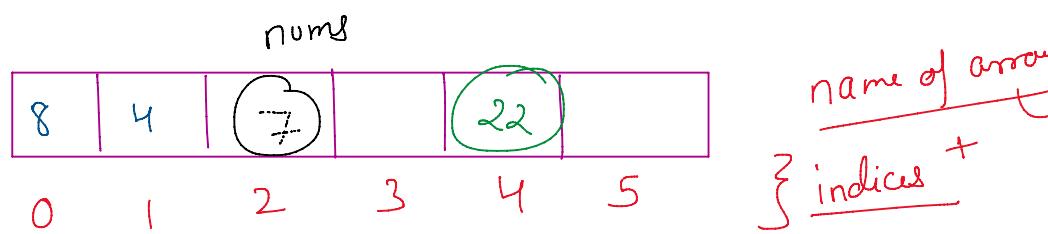


$$1024 \text{ Kb} = 1 \text{ mb}$$

$$1024 \text{ mb} = 1 \text{ Gb}$$

$$1024 \text{ Gb} = 1 \text{ Tb}$$

$$1024 \text{ Tb} = 1 \text{ Pb}$$



$$\text{nums}[2] = 7;$$

$$\text{nums}[4] = 22;$$

[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

index
5

SC 5
`for(int index = 0; index < nums.length;)`

Sout ("Element Ki Value enter Kar
nums[index] = SC.nextInt();

background z → (nums[4] = 22)

} refer to
elements

}

shall
hold.



index ++)
a: ")'

background → $\text{nums}[4] = 22$

$\therefore \frac{22}{\text{for}(\text{index})}$

```
for (int index = 0; index < 5; index++) {
    cout << nums[index];
}
```

Declaration + Initialization of an Array:

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

PRIME NUMBER PROGRAM USING ARRAYS :

[Divisible only by 1 or themselves]

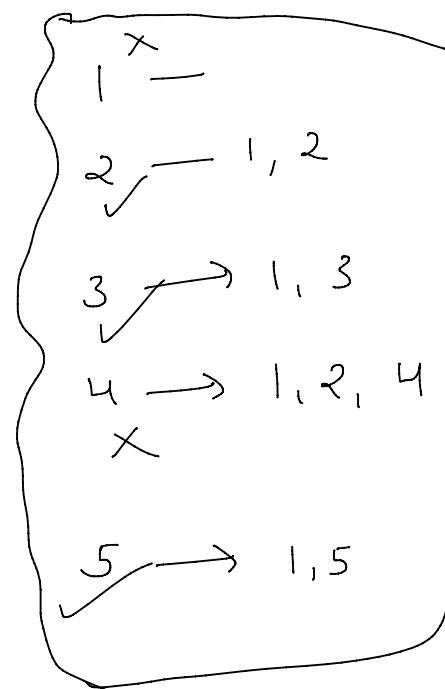
5	21	7	10	2
---	----	---	----	---

OR

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

i.e. 11

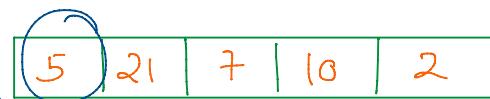
∴ If a Number (n) gets divided





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

①



②

Pick out an element & store it into a variable

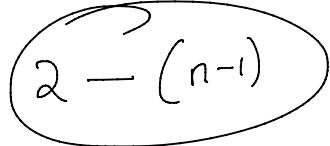


③

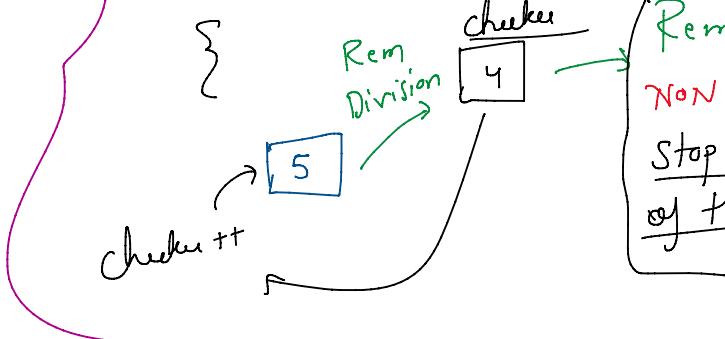
CHECK whether it's prime

④

Sout(Fact);



\rightarrow 1 → Sout (Neither Prime Nor
 \rightarrow 2 → Sout (PRIME H
 \rightarrow 2+ (> 2)



for (Composite)
A1)

f
n 0:
PRIME
execution
In loop

Core Java

```

PrimeNumberProgram.java 1 X
src > arrays > primeNumberProgram > PrimeNumberProgram.java > PrimeNumberProgram > main(String[])
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");
}

```

①

for (i = 100

PickedNumber

Checking

(min's elements)

The prime nos in the range

7, ,

Else

2 — PRIME

3 — NOT A

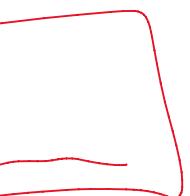
②

2 | 5 | 7 | 8 | 6

(Sum of all the Element
Array)

28

of 1 to 100 cm

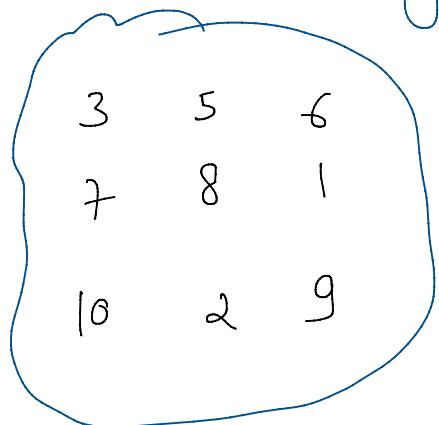


L_s of

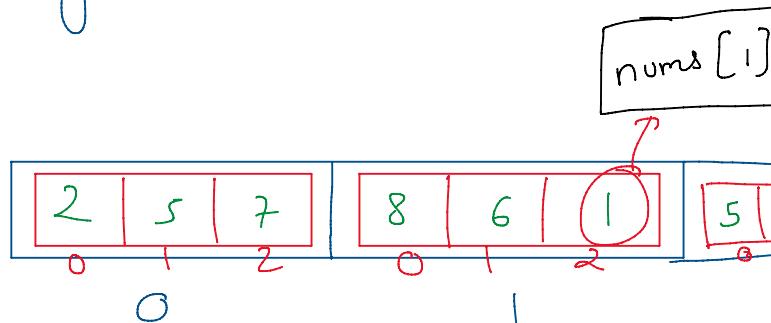
Sum

* 2D Array

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]`

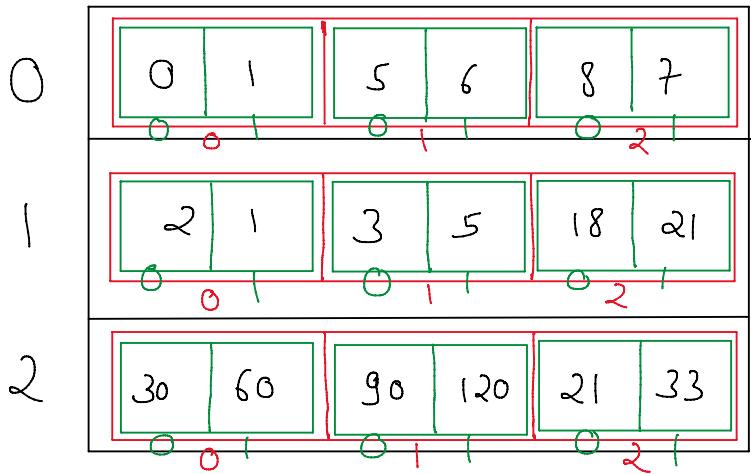
`[2][2]`

* 3D Array *

Array inside an Array inside an Array

$[2]$;

$\begin{array}{|c|c|} \hline 3 & 6 \\ \hline 1 & 2 \\ \hline 2 & \\ \hline \end{array}$



```

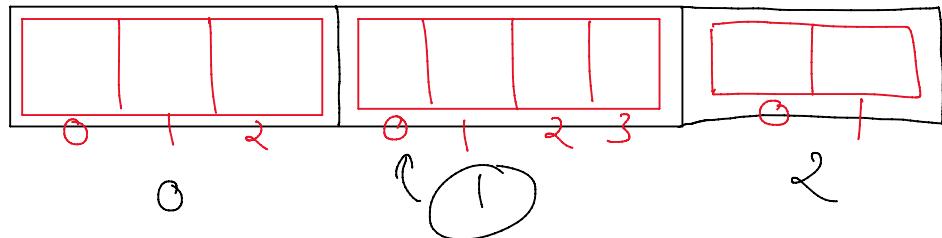
for {
    for {
        for {
            {
                {
                    {
                        {
                            {
                                {
                                    {
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

0 0 0
 0 0 1
 0 1 0
 0 1 1
 0 2 0
 ...

```
int num[ ][ ][ ] = 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 <

>();

U

Variable

Declaration : `ArrayList < Wrapper class > name = new ArrayList <`

Methods (Predefined):

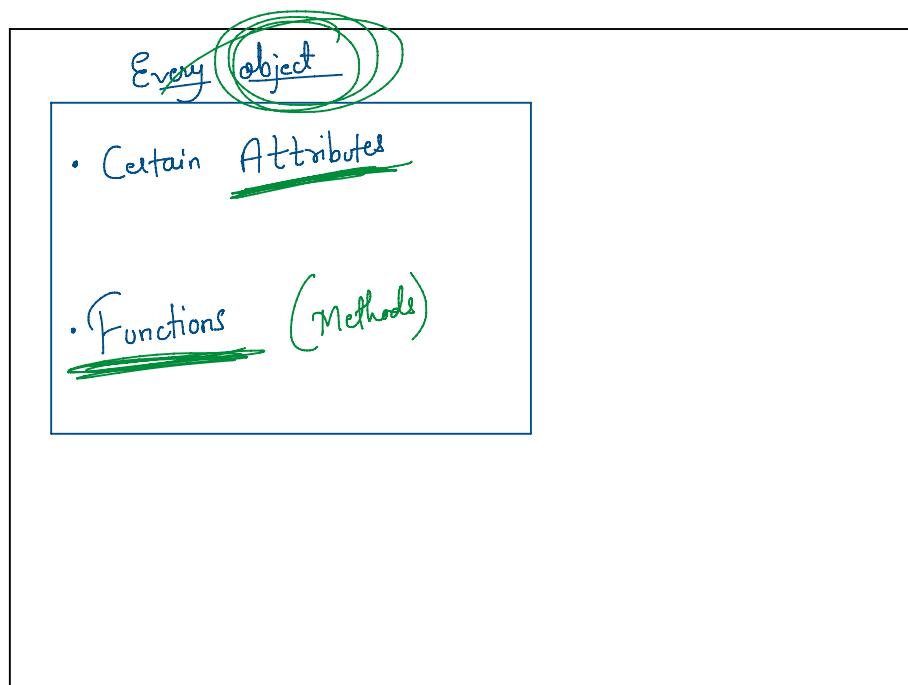
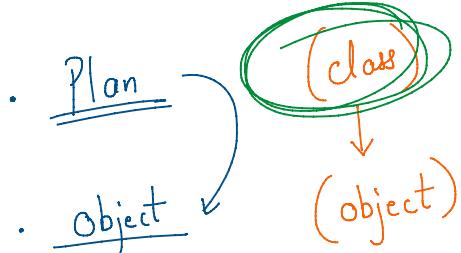
① add ⑤ Set

② get

③ remove

④ size (length of ArrayList);

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



Plan (class) Cars

Attributes

• model

? Object instance

RahulCar

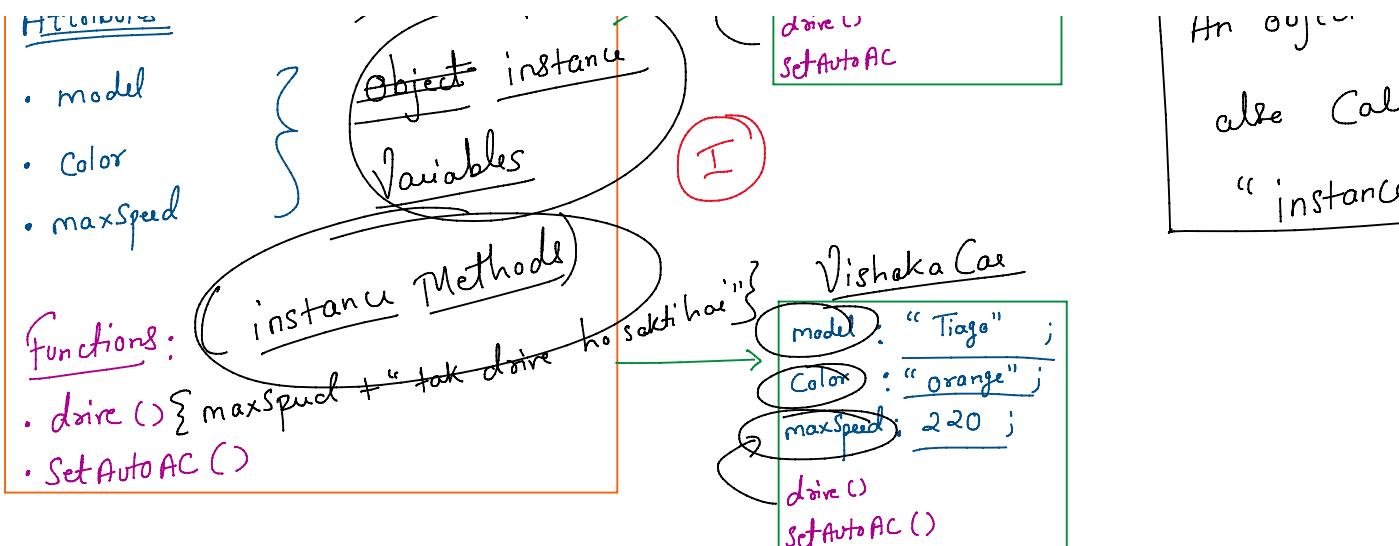
```

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

An object
↳ Car

>());

is
used



Static

Core Java

CreatingObjects.java 1 ●

src > OOPS > ObjectsAndClasses > CreatingObjects.java > 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 RohulCar

VishakaCar

model : "HondaCity"
color : null
maxSpeed : 0

VishakaCar.model = "HondaCity";

Core Java

CreatingObjects.java X

src > OOPS > ObjectsAndClasses > CreatingObjects.java > Cars > showDetails()

```
package OOPS.ObjectsAndClasses;

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

    public void showDetails()
    {
        System.out.println(this.model);
        System.out.println(this.color);
    }
}
```

VishakaCar

WHAT

WHAT

model : "Tiago";
color : "orange";
maxSpeed : 220;

end
")

0%

```

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
17 public class CreatingObjects
18 {
19     Run | Debug
20     public static void main(String[] args)
21     {
22         Cars VishakaCar = new Cars();
23
24         VishakaCar.model = "Tiago";
25         VishakaCar.color = "Orange";
26         VishakaCar.maxSpeed = 220;
27
28         Cars RahulCar = new Cars();
29
30     }
31 }

```

VishakaCar State:

- model : "Tiago"
- color : "Orange"
- maxSpeed : 220

RahulCar State:

- model : "Fiat"
- color : "Black"
- maxSpeed : 240

Ln 9, Col 30 Spaces: 4 UTF-8 CRLF {} java Go Live Kite: ready

Plan

String
String
String
boolean
SunRoof :

Color :

fuelType :

model :

These Variables are going to vary in their values whenever they go into any object.

drive()

Vishaka Car

Color: "orange"
fuelType: "Petro"
model: XZ
SunRoof: false
drive()

Object dep...

Object Va...

Instance Va...

Rahul Car

Color : "green"
fuelType : "Diesel"
model : XY
SunRoof : true
drive()

★ Heap & Stack ★

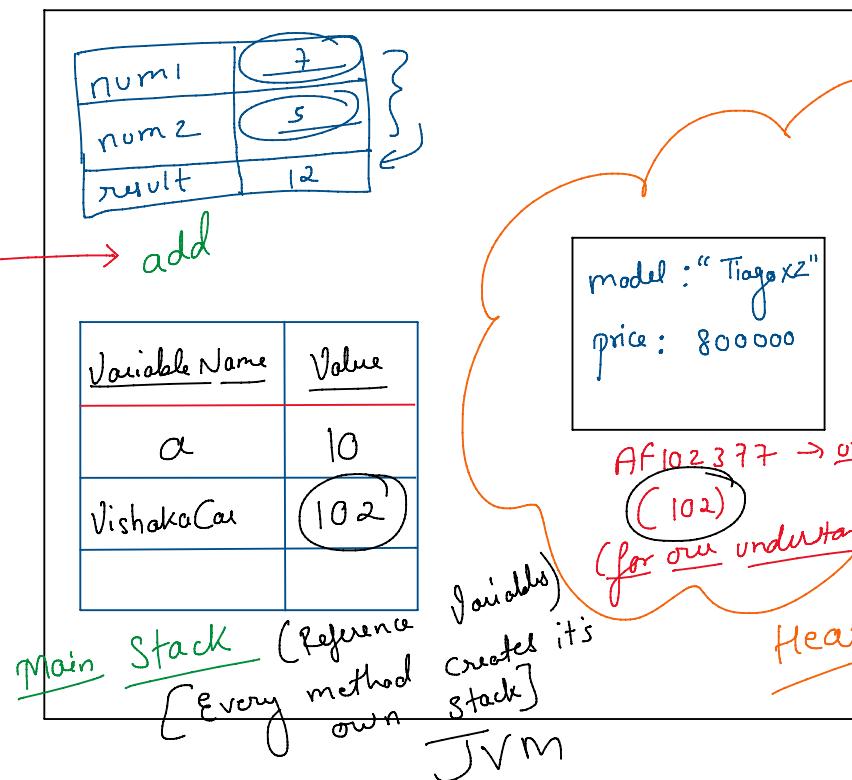
Using Variables
Variables

variables

```

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 }
8 public void add(num1, num2)
9 {
10    Run | Debug
11    public static void main(String[] args)
12    {
13        int a = 10;
14
15        Cars VishakaCar = new Cars();
16        VishakaCar.model = "Tiago XZ";
17        VishakaCar.price = 8.00_000f;
18        System.out.println(VishakaCar.price);
19    }
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.

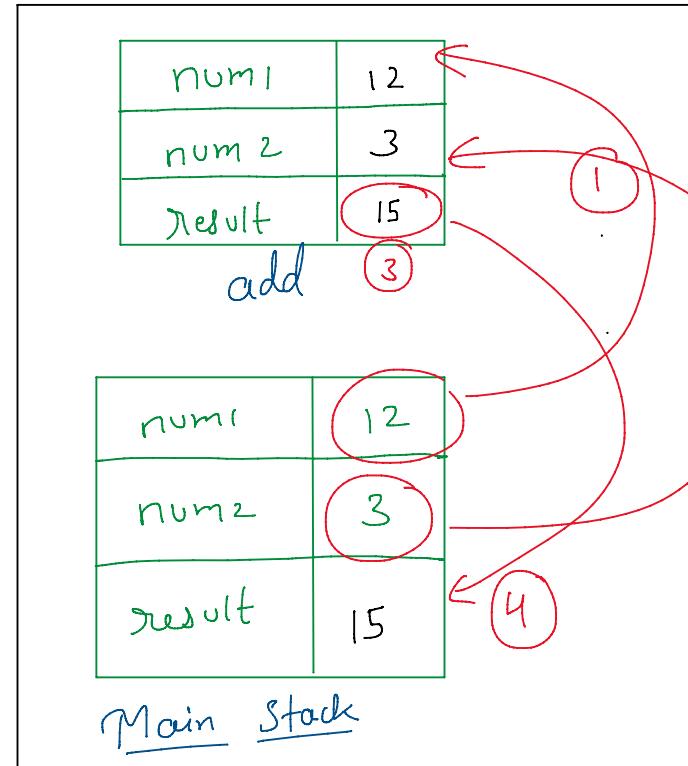
```

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

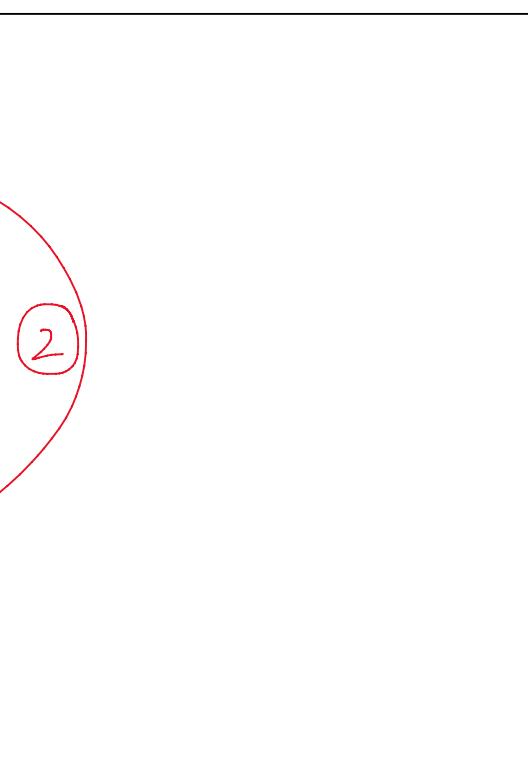
public class TwentyDecember
{
    public static int add(int num1, int num2)
    {
        int result = num1 + num2;
        return result;
    }

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

```



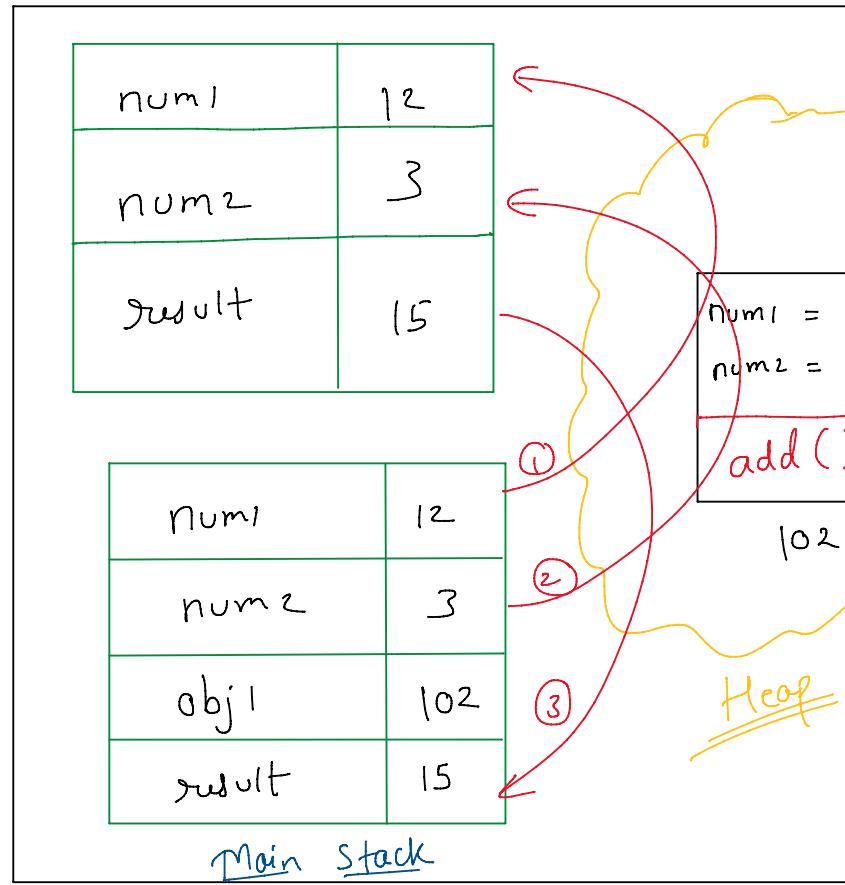
Each stack can have variables of unique

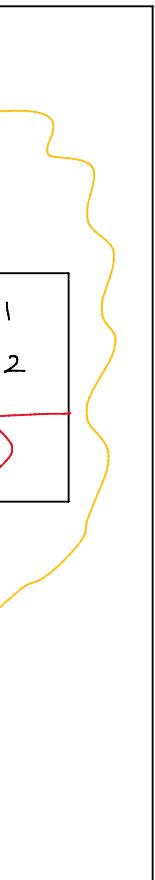


named

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 public class TwentyDecember2
22 {
23     Run | Debug
24     public static void main(String[] args)
25     {
26         int num1 = 12;
27         int num2 = 3;
28
29         ArithmeticCalc obj1 = new ArithmeticCalc(1, 2);
30
31         int result = obj1.add(num1, num2);
32         System.out.println(result);
33     }
34 }
```

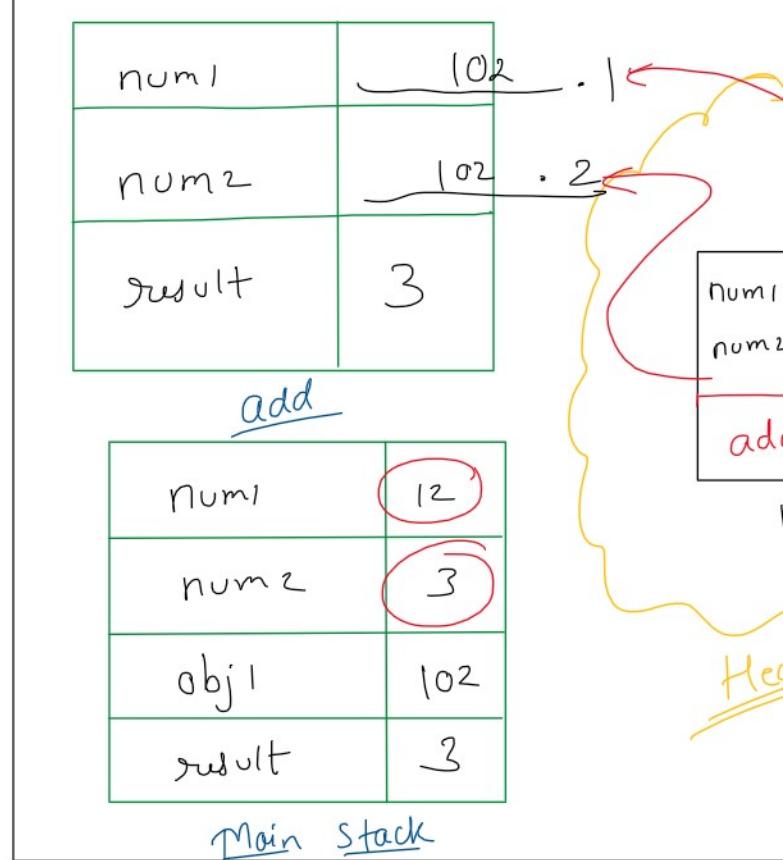




```

TwentyDecember2.java ×
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 public class TwentyDecember2
24 {
25     Run | Debug
26     public static void main(String[] args)
27     {
28         int num1 = 12;
29         int num2 = 3;
30
31         ArithmeticCalc obj1 = new ArithmeticCalc(1, 2);
32
33         int result = obj1.add(num1, num2);
34         System.out.println(result);
35     }
36 }

```



★ 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"

↓↓↓↓↓

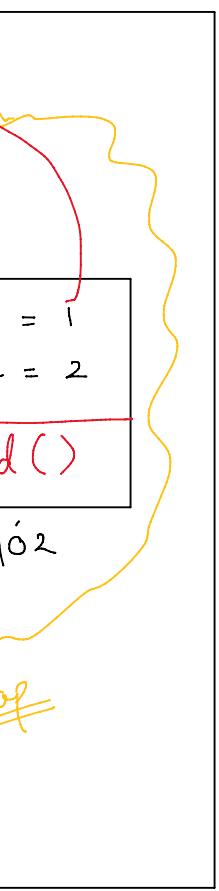
0 1 2 3 4

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

str = "RAHUL"

for (int index = 0; index < str.length(); index++)

System.out.print(str.charAt(index));



- -)

`Sout(str. concat());`

`name = "VISHAKA"`

0 1 2 3 4 5 6

`revStr = " ";`

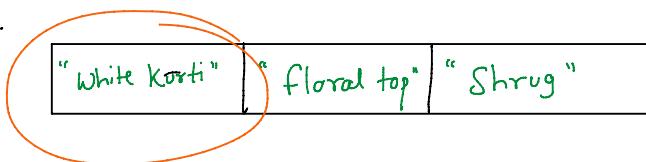
AKAH SIV

`for (int index=1; index >= 0; index--){}`

`revStr = revStr + str.charAt(index)`

AK
S

Q1



white Kurti : skirt | white Kurti : Jeans

"Skirt" | "Jeans" | "Leggings"

White Kurti : skirt

 : _____

 : _____

floral top : _____

 : _____

Sout all the
Combined Elements which
are inside arraylist
using for each loop.

CombString = C + "
outfits.add(");

Q-2

"RAHUL" | "VISHAKA" | "SNEHA"

Var

0

1

2

name

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

name

- -)

ex)

— — —

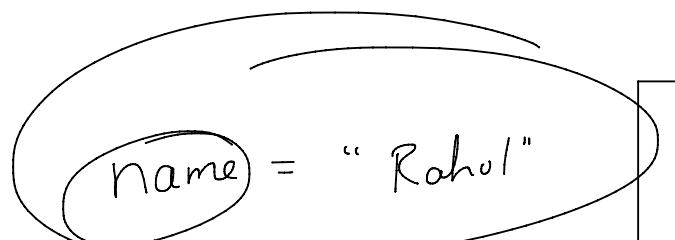
" + o

8

nes



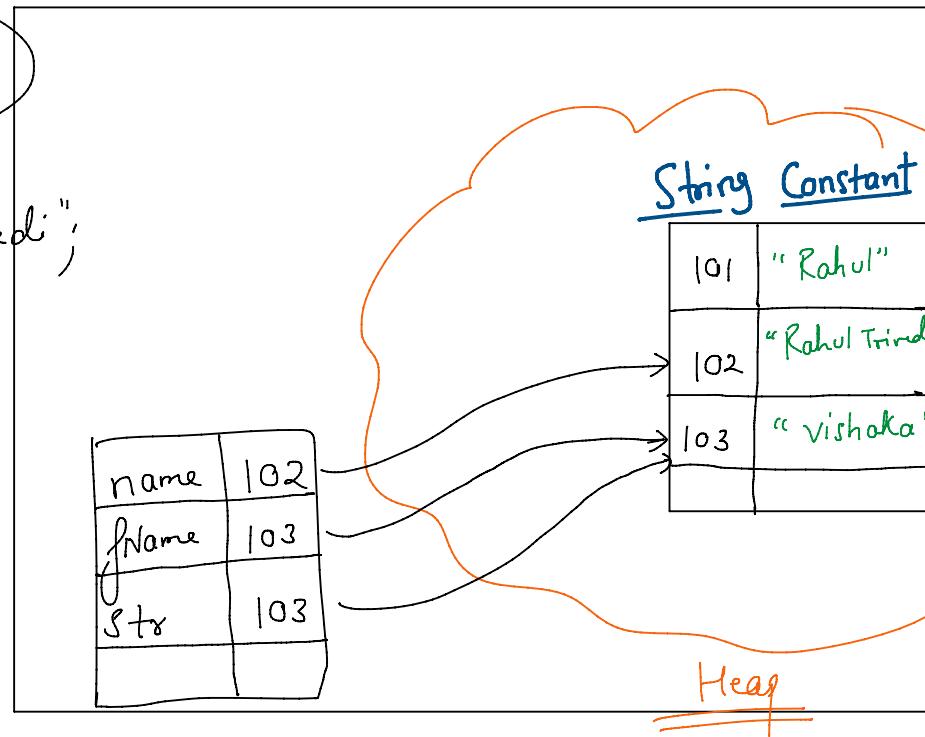
nam



`name = "Rahul" + " Trivedi";`

`fName = " vishaka";`

`str = " vishaka";`



String Constant

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

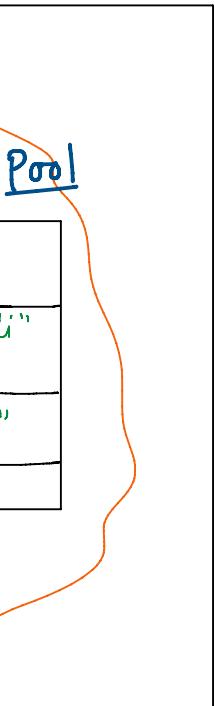
Heap

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"`

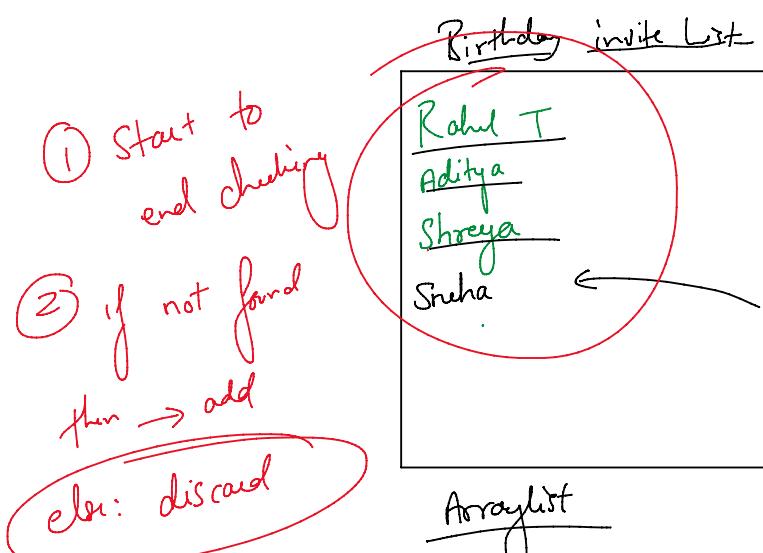
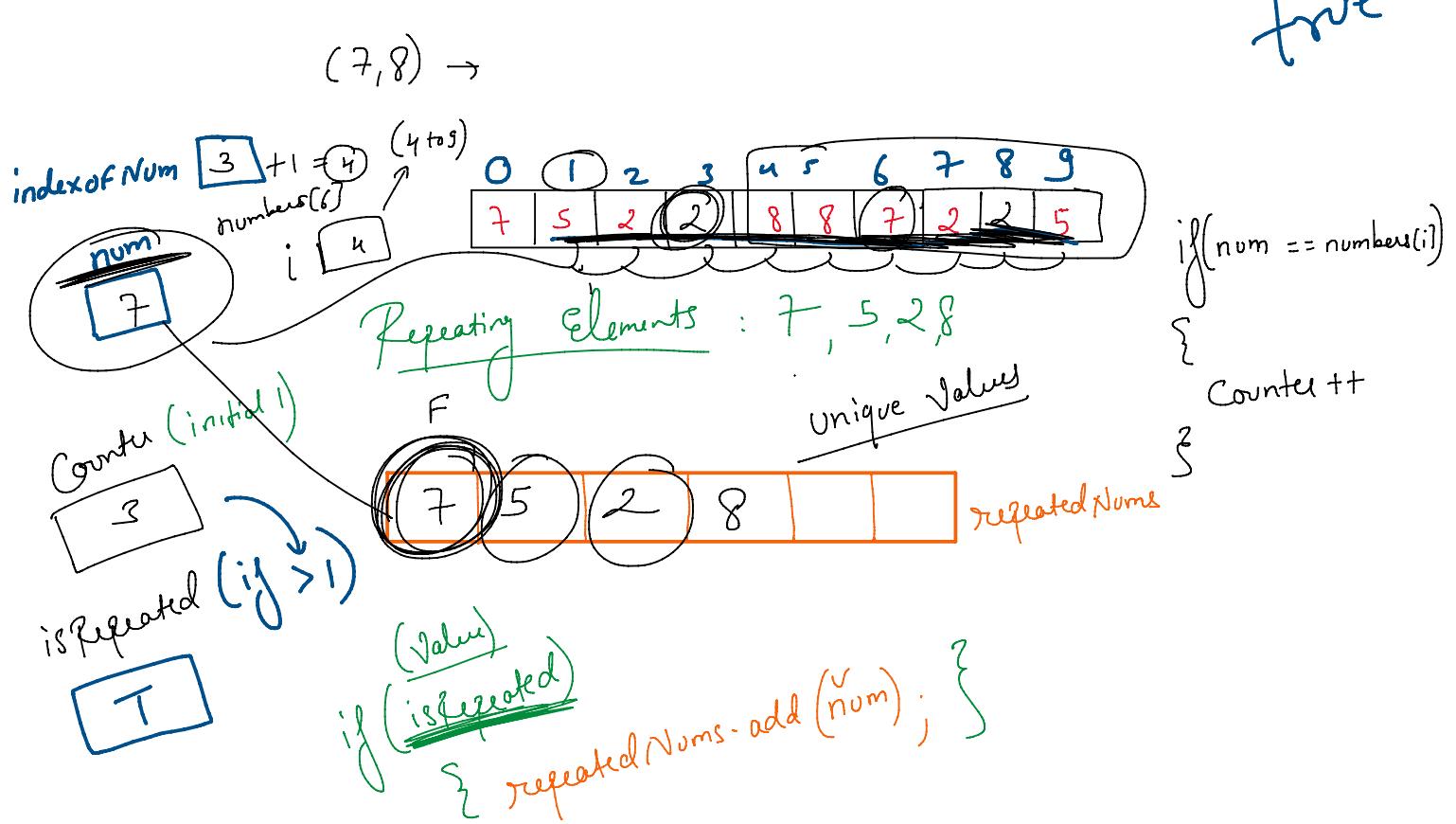
class

nel



~~(by-default method)~~
C +/ Str == name

j
 $\text{name} = "RAHUL"$
 C String class
Sort(str.equals(name)
 ↓
 true



~~if~~ if

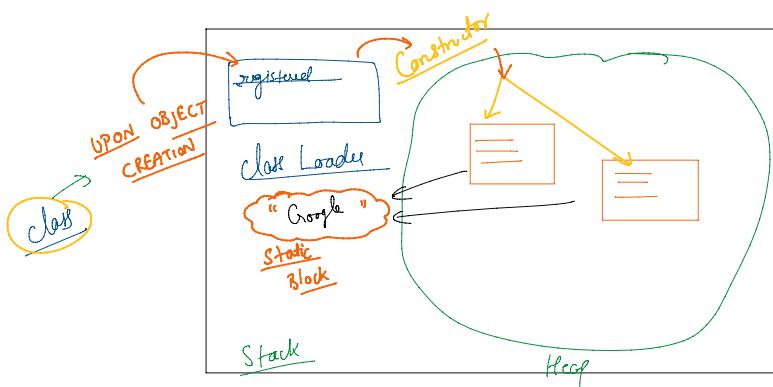
Sout (str == name)



false

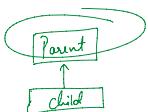
else: discard

ArrayList

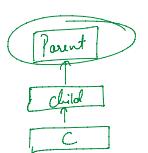


Types of Inheritance in Java:

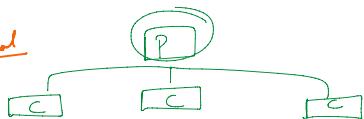
① Single



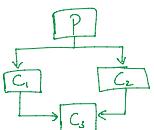
② Multilevel:



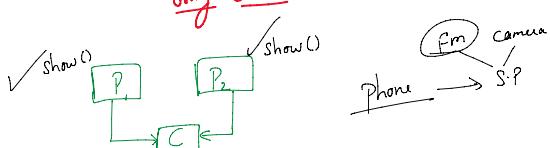
③ Hierarchical



④ Hybrid



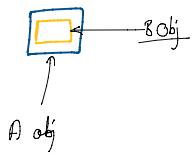
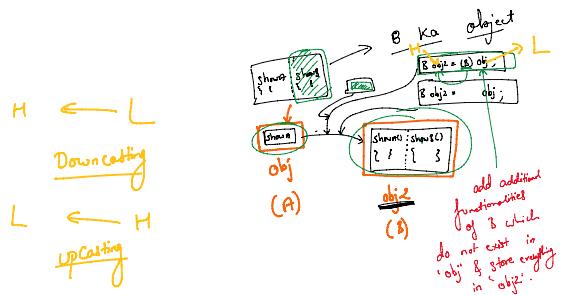
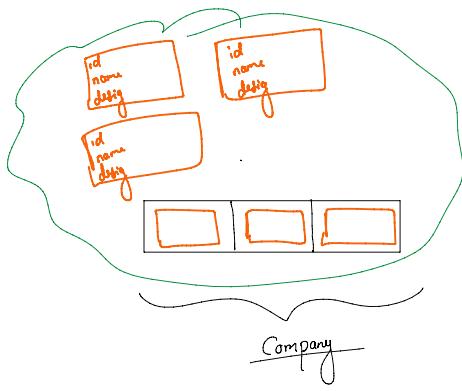
⑤ Multiple inheritance (Not Supported using classes)



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

interface fm
{
 ==> features
}



A screenshot of the Visual Studio Code interface. The left sidebar shows a file tree with 'ES-6 Javascript' selected, containing files like 'Addition.js', 'Addition.js (ES-6 Javascript)', and 'Addition.js (ES-6 Javascript)'. The main editor area displays the following code:

```
// ES-6 Javascript > Addition.js
1 // Let var1 = 12;
2 // Let var2 = 24;
3
4 // console.log(var1 + var2);
5
6 function add(num1, num2) {
7     let result = num1 + num2;
8     return result;
9 }
10
11 let resultValue = add(24, 32);
```

The terminal at the bottom shows the output of running the script:

```
36
rahul@Romeo-Alpha09 MINGW64 /r/VIMEET Training/ES-6 Javascript
$ node Addition.js
36
rahul@Romeo-Alpha09 MINGW64 /r/VIMEET Training/ES-6 Javascript
$
```

A screenshot of the Visual Studio Code interface. The left sidebar shows a file tree with 'ES-6 Javascript' selected, containing files like 'ArrayMethods.js', 'nums.filter() callback', 'Cw-1/Solution.js', and 'ArrayMethods.js'. The main editor area displays the following code:

```
ES-6 Javascript > ArrayMethods.js > nums.filter() callback
1 let nums = [1, 2, 3, 4, 5, 6, 7, 12, 22];
2
3 // nums.forEach((number) => {
4 //     console.log(number * 2);
5 // });
6
7 nums.filter((num) => {
8     return num % 2 === 0;
9});
```




>

parts

