Core JAVA:-

① Javac --version

↳ To Compile the code

② Java --version

↳ To Run the application

→ JShell was Introduced in Java 9.

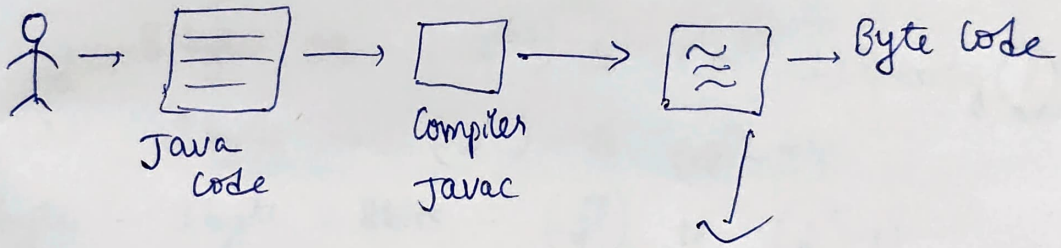
↳ To experiment Java code in short code

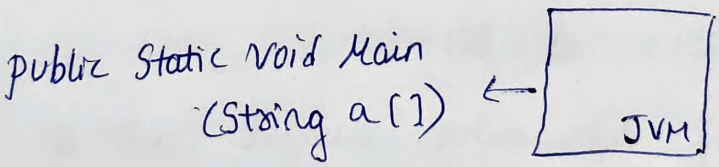
Ex: 2+3 ⇒ 5

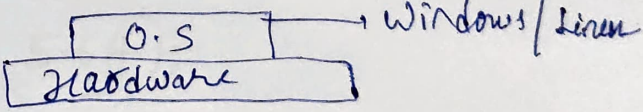
Use Methods :-

System.out.print("Hello world");

③ Java is platform Independent.







④ Every thing should be an object.

To create object we need a class.

⑤ Javac hello.java → 1st step

Java hello.class. → 2nd step.

→ we need libraries & runtime → JRE

* JVM is part of JRE (JVM + lib)

* Write once Run Anytime [JRE + JVM required on other machine].

✓ Variables

✓ Data types

primitive

Integer → byte, short, int, long

float → double, float

String

Boolean

int → 4 bytes - 32 bits - (-2^{31}) to $(2^{31}-1)$

long - 8 bytes → (-2^{63}) to $(2^{63}-1)$ Specify (l)

Short - 2 bytes → (-2^{15}) to $(2^{15}-1)$

byte - 1 byte - 8 bits - (-2^{7}) to $(2^{7}-1)$

-128 to +127.

* double is the default value for float

to declare float use

float num = 5.6(f);

* Char - 2 bytes

UNICODE → default. & Use Single quotes.

* Boolean → True & false

↳ Doesn't work with 0 & 1.

* int num1 = 10_00_00_000
↓
we can put this

* for a variable its name & type compulsory.

- Type Conversion & Casting :-

byte value into integer variable
↳ Conversion

b = byte (a)
↳ Casting.

→ Single line compile & Run (Not Recommend)

Java hello.java

? what kind of Error will be thrown If the main method is not found?

A) Runtime Error.

? default value assigned to boolean ?

A) false.

? what does a high-order bit represent for an Integer ?

A) Sign of the integer.

* Arithmetic operators :-

+, -, /, *, %

post increment / pre

* Relational Operators:

<, >, ==, !=

* Logical operators:-

& - AND | - OR ! - NOT

↳ Refer JS notes.

* Conditional Statements.

If else. If Else If

* Ternary Operator:-

result = n%2==0 ? 10 : 20;

* Switch Statement:-

```
Switch (n) {
    Case 1:
        System.out.println("Monday");
        break;
    Case 2:
        ____________
        break;
    :
```

* Loops:-

While Loop:- → when there is no Start point but Endpoint

Do while Loop:- do { } while ();

For Loop.

? Which conditional Statement is used to check multiple possible values of a variable ?

A. Switch

→ Object knows Something &
Does Something

→ Class acts like blueprint for the object.

→ JVM Creates objects in Java
↳ its my Job to get the object but you give me the blueprint.

✓ Every object has properties & Methods.
actions will be written in Method.

→ Reference Variables

```
Class Calculator
{
    int a;  ---> Instance variable
    public int add (int n1, int n2) {   (n1, n2: Local variable)
        int r = n1 + n2;
        return r;
    }
public class Demo {
    public Static void main (String a[]) {
        int num1 = 4;
        int num2 = 5;
        Calculator calc = new Calculator;
int result = calc.add (num1, num2);
        System.out.println(result);
    }
}
```

Extra classes, inbuilt classes → here JRE helps ultimately your code runs on JVM.

JDK → JRE → JVM

→ In Java we can create components with the help of classes.

→ Method Overloading :-

① Same name of Methods but different parameters of different type of parameters

You have 2/3 methods with the same name.
What is different is parameters.

→ Method Overriding :-

→ Stack & Heap :-

inside JVM Memory

↳ LIFO

↳ Open Space It Expands

① Every Method will have its own Stack.

```
Calculator obj = new Calculator();
```

obj → Variable → In Stack.

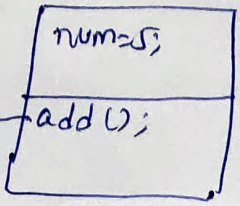
new Calculator(); ↳ Creates object in heap

② Local variables are part of Stack

③ Instance variables are part of Heap.

Heap have 2 Sections

Actual area consumed is Stack.



Calculator obj = new Calculator();
Calculator obj1 = new Calculator(); > different

obj.num = 8

System.out.println(" obj.num) => 8

" (obj.num) => 5

*Array:-

int num[] = new int[4];

int num1[] = {1,2,3,4};

* Multi Dimensional Array.

int nums[][] = new int[3][4];

* Jagged Array 3D:-

Columns are not fixed

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

nums[0] = new int[3];

nums[1] = new int[4];

nums[2] = new int[7];

* Arrays vs Collections.

* Exceptions are Runtime Errors

→ for Each loop:

for(int n : nums){

S.O.p(n)}

String :-

It is a class.

Small letters int, char, boolean → are primitive type

String name = new String ("navin");

Inside Heap.

String Constant pool

Mutable String :- String Buffer
String Builder.

Immutable String :- → Strings are Immutable

StringBuffer sb = new StringBuffer ("Navin");

sb.capacity () → 16 character
& adds more like buffer

sb.length ()

→ is thread Safe.

Static Variable :

If all objects have same variable.

```
class Mobile {
    String ~~name~~ brand;
    int price;
    String name; ⟶ Same for all objects
}                    then use
                     Static

Static String name;

↳ Mobile.name = "Smartphone";
```

Static variable is shared by all the objects.
Common variable used by all objects

Static Method.

① we can call Static Method directly with the help of Classname.

② you cannot use non-static variables inside your Static Method.

③ you can use it by passing object reference

To call main we don't need object of Demo/class.

because -

public static void main (String a[])

main is Method.

Static block :-

```
class Mobile {
    String brand;
    int price;
Static String name;
    Static {                    ⟶ Static block
        name = "phone";
    }
    public Mobile () {          ⟶ Constructor
        brand = " ";
        price = 200;
    }
    public void Show() {
        Sop ("brand+" : " + price + " : " + name);
    }
```

@ class loads & objects are instantiated

In JVM we have Special Area called class Loader

If you wants to load class without creating objects & calling it then use

```
class.forName("Mobile");
```

↳ In JDBC we use this.

class → object Now topic is Encapsulation :-

private Variables will be only accessible in the same class.

Ans: Assign & get the values by methods

```
public void setAge(int a) {
    age = a;
}
public int getAge() {
    return age;
}
```

we are binding our data with the methods

getters & setters:

right click & open Source Action.
choose generate getters & setters

This keyword :-

which represents current object
object which is calling the method.
To refer to Instance variables we use this

Constructor :-

Looks like a method.

Name same as classname.

```
Public  Human() {

}
```

→ Add Connection to the database here.

✓ parameterized Constructor

* Naming Convention :-

→ Camel case.

Class → Capital ⇔ Interfaces

Variables & Method → Small

Consts → All Capital.

* Anonymous objects :

```
new mobile();
```

* Inheritance :-

```
public class Advcal extends Calc
```

Advcal ↓ Subclass

Calc ↓ Super class.

* Single & Multilevel Inheritance :-

⇒ It will call the Constructor of Sub & Super class both.

✓ Every Constructor has super(); Even if we dont mention

↙ means

call the constructor of Super class

* Every class in Java extends object class.

this()
↳ this will execute the constructor of same class

* Method Overriding :-
Use in overriding methods of parent class

* packages :-

```
import tools.calc;      } ⟶ tools.*;
import tools.Advcalc;   }
import java.util.Arraylist;
```

inbuilt class / Use in Java belongs to a package.

by default - import java.lang.System;
.*;

→ Get Libraries from MVNrepository.com.

* Access Modifiers :-

Public :-

If same package we can Access Variables no public keyword required

If Different / outside of package then use public thats why most of the time methods are public.

Private :-
Can be used only by inside the class.

Default Access Modifier :-
↳ Can be accessed in the same package.

~~Protected~~ :- Cannot be Accessed by different package
It should be Subclass to be Accessed If outside package

- Cannot have 2 public classes in the same file.
- Keep your Instance variables as private
- methods as public
- If Subclass Access then use protected
- Avoid default.

* polymorphism :-
  - poly → many
  - morphism → behaviours

Many

Compile time
1. Early binding
2. If behaviour defined at Compile time. It is called Compile time polymorphism
3. What thing will be get Executed at the Compile time itself.

Run time
1. Late binding
2. overriding

4. Overloading

* Dynamic method dispatch :- ← Run time polymorphism
  - Type as parent
  - Object as child

```
A obj = new B();
```

Reference of Super class ↳ Object of Sub class

Same object is behaving differently with different objects.

```
A obj = new A();
obj.show();

obj = new B();
obj.show;

obj = new C();
obj.show;
```

* Final Keyword. Same as Const.
  ↳ Variable, method, class

To Stop inheritance of class use final class calc.

To Stop method overriding public final void show ()

* Object Methods :-

Creates a Single String with all the data you have
  ↳ hash code

println ("obj. toString())

Right click on Vs code & Generate

* Type Casting

Upcasting Implicitly works.

we can create a parent reference & child obj & we can also come back to the child reference

```
A obj = new B();
obj.show1();
B obj1 = (B) obj;
obj1.show2();
```

* Wrapper class:

for Every primitive type we are going to have class for it.

This class Extends the object class.

int → Integer → parseInt()

char → Character

double → Double

✓new keyword Allocate Memory for an object during runtime in Java.

✓main() Method can be overloaded in Java.

✓2 | more Methods of the same name within the same class → Method overloading.

✓Instance variables allocated in heap Memory.

✓Homogenous data types are allowed in an Array in Java.

* To print object we have to use .toString Method

→ Abstract keyword :-

defining a Method → public void drive(){}

Instead we can declare Method

↳ public void drive();

Abstract Method can only belong to abstract class

You can't create object of Abstract class.

* Inner class:-

```
A obj = new A();
obj.Show()

A.B obj1 = obj.new B();
obj1.Config();

A.B obj = new A.B()
          ↳ static class
```

✓ Static can be used only for Inner class.

Anonymous Inner class:-

Note: first you have to create the class

I want to create object of A but this is the Implementation.

```
A obj = new A() {
        public void Show(){
            S.O.p("in new Show"); };
obj.Show();
```

↳ Inner class Anonymous

used in functional Interfaces, Lambda Expressions.

* Interfaces:-

↳ It is not a class.

↳ Every method is public abstract by default.

↳ It just show you the design

class B implements A {

↳ variables are by default final & static.

↳ Interface objects don't have Memory In the heap.

one class Implementing multiple Interfaces.
↳ not in case of Abstract class

class - class → extends
Interface - Interface → extends
class - Interface → implements.

Need of Interface:-

What is enum: → class

enum Status {
Running, Failed, Pending, Success; }
↳ Named Constant
Objects.

Status s = Status.values();

enum if & Switch :-

enum class:- No you can't extend enum with any other class.

U can define Methods
Constructors.
own Variables

It Extends enum class.

Annotations :- Supplement to the runtime / Compiler / meta data.

Supply Extra Info to Compiler / runtime
@override , @Deprecated

Types of Interface:-

Normal → More than 2 methods

functional / SAM → only one Method

Marker → Blank Interface

Ex: Serialization

You can take the object and store the values of the object in hard drive

SAM: Single Abstract Method Interface:-

- we can use Anonymous Inner class.
- use Lambda expression in functional Interfaces.

* Lambda Expression:-

```
A obj = () -> {
        System.o.p("in show")
    };
obj.show();
```

- It will not create new file.

Javac Demo.java

with return:-

```
A obj = (i, j) -> i+j
```

Exceptions:-

Different types of Errors:-

① Compile time Error:- Spelling / Syntatic error.

② Logical Error: output is wrong → Bug.

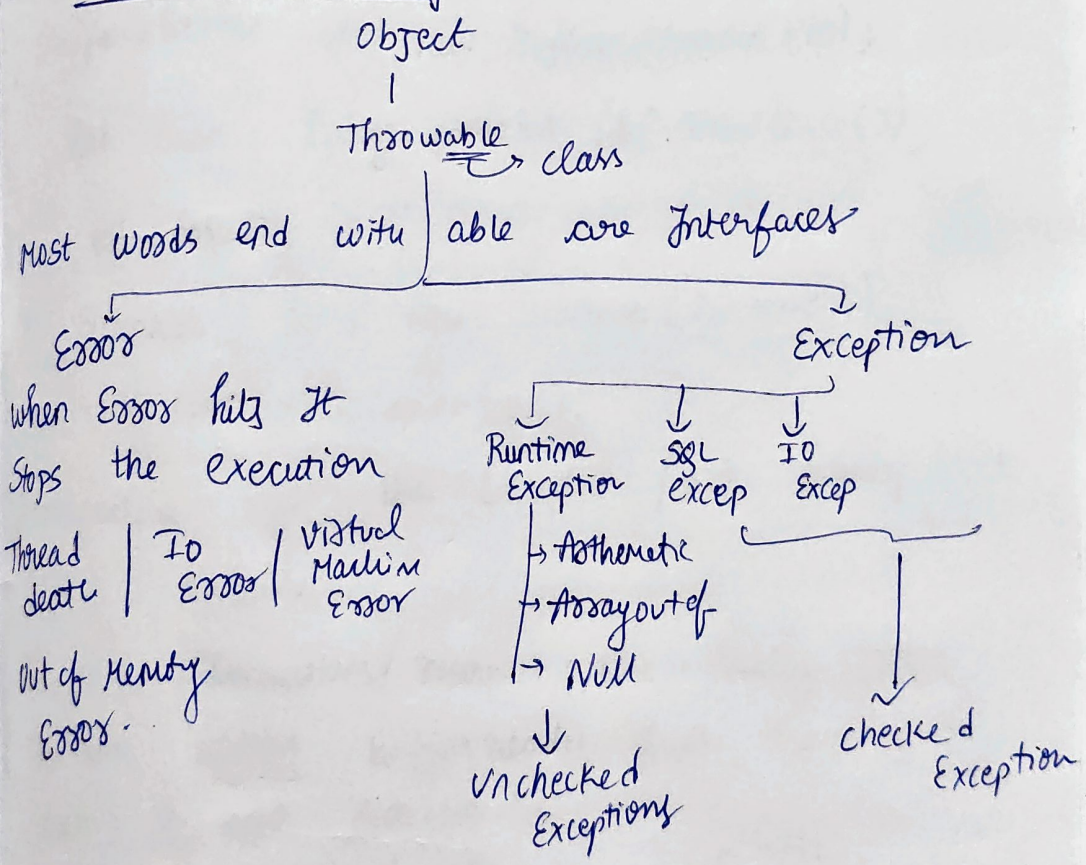
3) Runtime Errors: Code Execution will Stops.

↳ If the file exists then only open the file

↳ number dividing by zero. check before

```
try {
    j = 18/i;
}
catch(Exception e){
    Sop("Something went wrong"); }
```

* Exception hierarchy :-



Throw Keyword :-

```
If (j==0)
    throw new ArthemeticException();
```

Custom Exception:-

```
class Baseexp extends Exception {
    public Baseexp(String string) {
    super(string); }
```

* User Input using Buffered reader & Scanner

① println belongs to printstream. class.

② System.in.read() → gives Ascii value

③ InputStreamReader in = new InputStreamReader(System.in);

```
BufferedReader bf = new BufferedReader(in);
int num = Integer.parseInt(bf.readline());
bf.close();
```

④ Scanner sc = new Scanner(System.in)

```
int num = sc.nextInt();
```

Irrespective u got the Exception / not finally block will Execute.

To close Connections / resources use finally block.

If we declare bufferreader inside the () of try then it will Autocloseable.

↳ Try with resources.

Threads :-

- Smallest Unit you can work with
- Shares resources among them

① extends thread

② call Start() Method.

③ Concept of Schedular in O.S

④ Thread priority

```
obj2.setPriority(Thread.MAX_PRIORITY);
```

(1-10)

⑤ Thread.Sleep(10);

⑥ Runnable obj1 = new A();

Runnable obj2 = new B();

Thread t1 = new Thread(obj1);

Thread t2 = new Thread(obj2);

t1.Start();

t2.Start();

⑦ Thread Safe :- Race condition :-

Main started the job of 2 threads & simply print the value of count.

So use

t1.join();

t2.join();

2 threads working with same variable.

↳ use keyword

public Synchronized void increment().

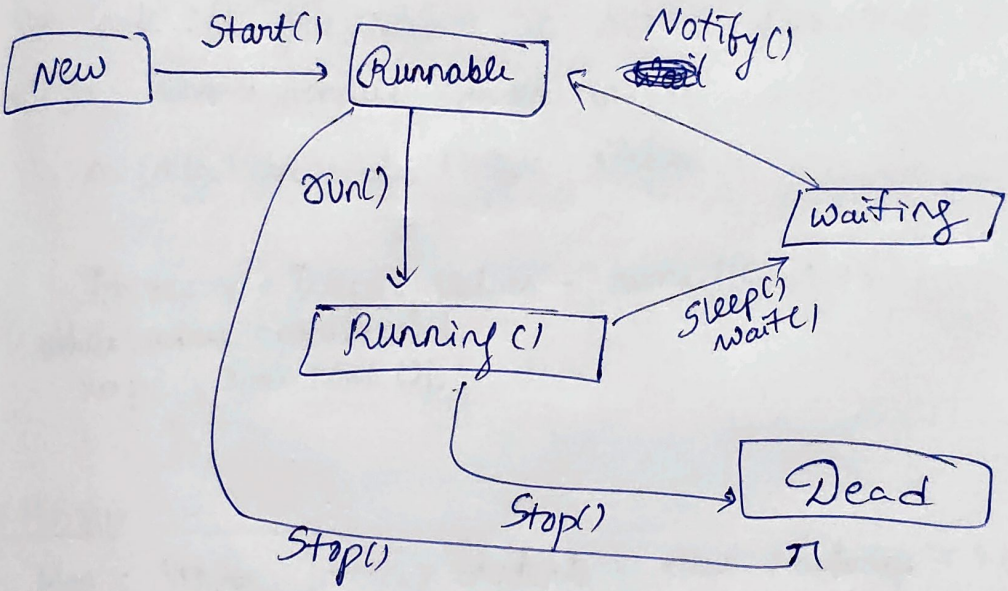
When your thread is actually running on cpu its running state

① when your thread is Executing & then it is waiting for the Schedular that is Runnable State

② Keep it on hold with sleep method. / wait method that goes into waiting State.

③ notify() → with this we can comeback to ~~running~~ Runnable State

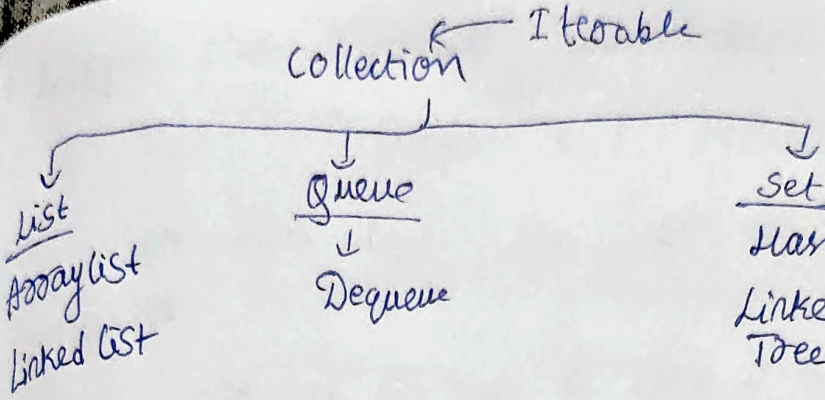
④ To Keep it dead → you can Also say stop().



Collection API :- ↳ Concept

Collection
↳ Interface

Collections
↳ class.



Collection <Integer> nums = new ArrayList <Integer>();

If need index values then use List.

List < Integer> nums = new ArrayList <Integer> ();

Set will not give values in sorted format

↳ It doesn't have index values.

↳ A collection of unique values.

```
Iterator < Integer> values = nums.iterator();
while(values.hasnext()) {
  Sop( values.next ());}
  }
```

*MAP :-

```
Map < String, Integer> students = new Hashmap <> ();
        students.put (      );
        Students.get (key);
        students.KeySet ();
```

→ Comparator :-

```
Comparator < Integer> Com = new Comparator <Integer> ()
{ public int Compare (Integer i, Integer j) {
        { If (i%10 > j%10)
                return 1;
          else
            return -1;
        }
```

* Integer class implements Comparable.

CompareTo() → method

on which logic you want to sort use comparator

If you want to powers to class itself to compare itself / to compare obj to itself

↳ we can use Comparable

*Stream API :-

nums.forEach(n → S.O.P(n));

↳ we can data to database

we can use stream only once.

nums.parallelStream()

Stream<Integer> S1 = nums.stream();

S2 => S1.filter

S3 = S2.map

result = S3.reduce

filter has boolean predicate that has test method.

Map has needs object of function

↓ it takes 2 types

has a method called apply.

Reduce take 2 param ① → type

② operation

↳ bifunction

↓

Apply method.

Multiple threads

```
int sum2 = nums.stream()
            .map(i -> i*2)
            .mapToInt(i -> i)
            .sum();
```

optional class:-

① Error is null pointer Exception

its because when you try to perform the operation on a null value. It will give you null pointer Exception.

```
Optional<String> name = names.stream()-
                        .filter(str -> str.contains("x")
                        .findFirst();
                        .orElse("Not found")
S.O.P(name.get());
name.orElse("not found");
```

Method Reference:-

```
List<String> uNames = names.stream().
                      .map(String::toUpperCase)
                      .toList();
```

Method reference

```
uName.forEach(System.out::println)
```

↳ Method reference.

You can pass the method name inside the method by specifying this method is responsible to do the work for you.

Constructor References

```
Students = names.stream()
           .map(Student::new)
           .toList();
```

? which access Modifier is used to access members of a inherited class when two classes are in different packages

A) → protected.

? Binding of Data & Methods → Encapsulation

? What is the return type of Overridden Methods

↳ Covariant return types.

? Char Sequence Interface is Implemented by String, StringBuffer, StringBuilder classes

? The Method defined by Lambda expression doesn't have a name → True.

? What does inner classes promote in Java?

→ Composition & Aggregation.

new outer().new Inner().m1();

? Can Subclass reference variable to be Assigned to a Superclass reference variable?

→ yes

? In which order Constructors Executed when a class hierarchy is created.

↳ from Superclass to Subclass