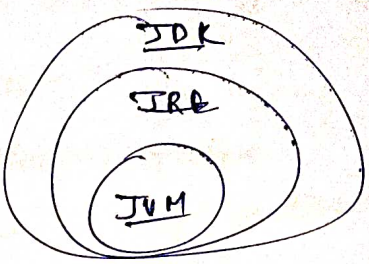
① Features of Java:-

- → Simple
- → object oriented
- → Robust (have Strong Memory Mgmt.).
- → Multithreaded.
- → Dynamic



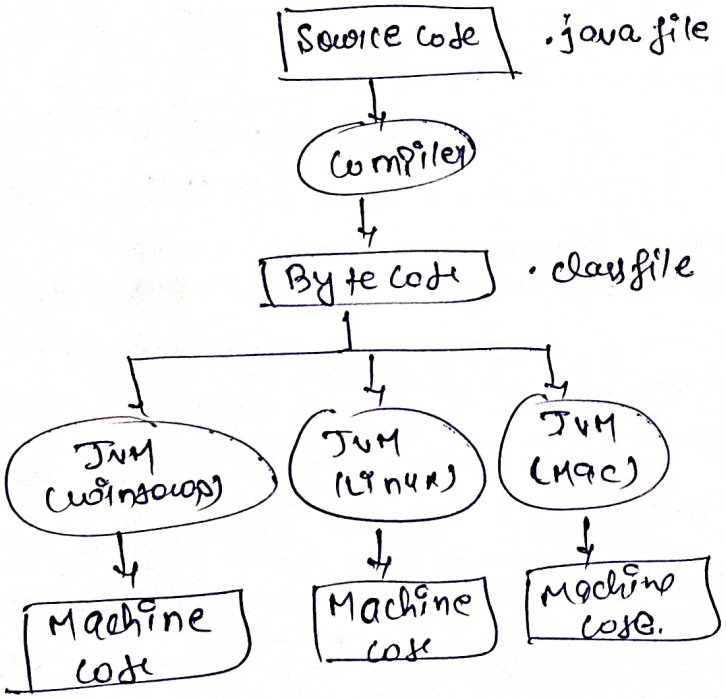
② Components of Java:-

→ JVM ⇒ compiles the code to Bytecode.
- .class extension
- Platform dependent

→ JRE ⇒ Combines JVM & Java class library & runs the Java applications.

→ JDK ⇒ To develop Java programs.
- Tools included ⇒ compiler (javac.exe.)
- → Java app launcher (java.exe).

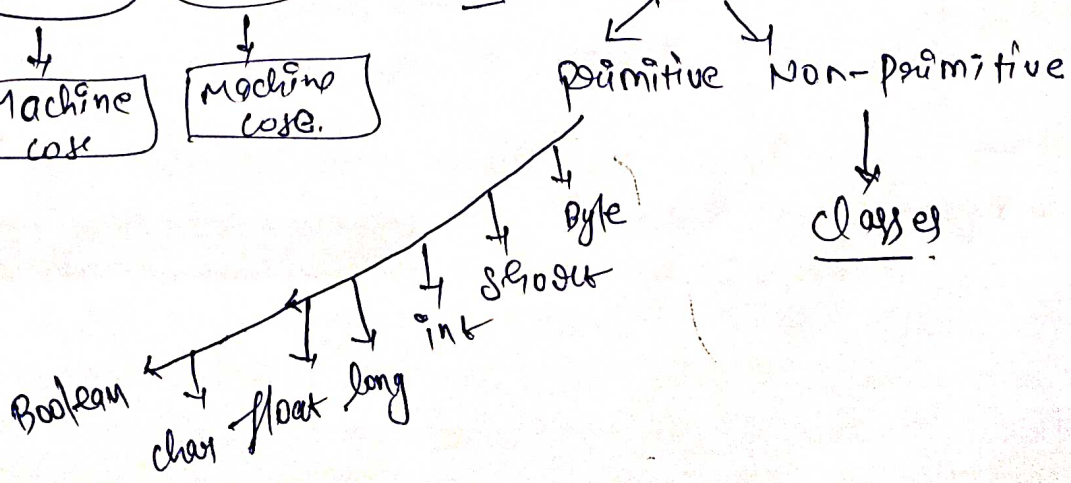
③ Why Java is platform independent:-



④ Java Tokens

- → keywords
- → Identifiers
- → constants
- → Strings
- → Special Symbols
- → operators

⑤ data types



Byte → 1 Byte → -128 to 127
Short → 2 Byte → -32768 to 32767
int → 4 Bytes → -2147483648 to 2147483647
long → 8 bytes
float → 4 bytes → $1.4e^{-045}$ to $3.4e^{+300}$
double → 8 bytes
char → 2 bytes → 0 to 65536
boolean → Not Defined → true or false / 1 bit.

⑥ Type casting:-

widening	Narrowing
Byte	double
↓	↓
Short	float
↓	↓
int	long
↓	↓
long	int
↓	↓
float	Short
↓	↓
double	byte.

⑦ operators:-
→ Arithmetic
→ Relational
→ Bit wise
→ Logical
→ Assignment
→ conditional / Ternary
→ Instance of operator.

⑧ conditional statements:-
→ if
→ if else
→ if elseif
→ switch case

⑨ Loops:-
→ while
→ for
→ do-while
→ continue
→ break
→ nested loop.

⑩ Maths functions:-
→ sin → cos → random().

⑪ class:- → Derived Datatype
→ combination of data members & functions. & constructors.
→ template for an object

⑫ Object:- → instance of a class
→ has a state & behaviour.

⑬ Class

① class is a blueprint or Template.
② logical ~~entity~~ framework
③ Declares once
④ Doesn't allocate memory.
⑤ class can exist without its objects
⑥ group of similar objects

Object

① Instance of a class.
physical reality
can be created many times.
allocate memory
But object can't exist without a class
object can be created by many times.

④ Constructor:-

→ Special method used to initialize objects.
→ constructor called when object of class is created
→ initializes immediately
→ same name as the class
→ JVM first allocates memory for variables & then execute the constructor to initialize instance variable
→ JVM calls it automatically
→ no return type
→ called at run time

⑤ Types of constructor:-

- Default → automatically Defined by JVM
- No-argument → Having no arguments.
- parameterized → it contains some parameters
- copy constructor → create new object using existing object.

⑥ Advantage of copy constructor:-

→ easier to use
→ no need for any type casting
→ complete control over object creation
→ pass by reference
→ Allocate appropriate memory to object

⑦

	Constructors	Method()
• Naming	Same as class name.	can be anything
Return type	Not have any return type even void.	Must have, at least void return type
call	→ Invoked implicitly	→ Invoked explicitly
Purpose	→ To initialize an object.	→ To execute the code
Inheritance	cannot be inherited by subclass	Can be inherited by subclass

(8) Destructor:-

→ opposite of constructor
→ Destroy the resources occupied by object
→ last function
→ In Java there is a garbage collector automatically called
→ there is no concept of destructor in Java
→ working:-
  → object created it occupy space in heap
  → if object is no longer used by thread then it is eligible for garbage collection
  → when garbage collector destroys an object the JRE calls finalize() method to close connections such as Database & network

(9) this:-

→ this is a reference variable refers the current object
→ invoked current object methods.
→ " " class constructor
→ can be passed as a parameter to constructor.
→ can be used to return the current object from the Method.
→ reference the current object.
→ used to Differentiate b/w local & instance variable

eg:
```
class A{ int a; // instance variable
  A(int a){ // local variable.
    this.a = a;
  }
  void show(){
    sout(a);
  }
  public static void main(String--){
    A o = new A(100);
    o.show();
  }
}
```

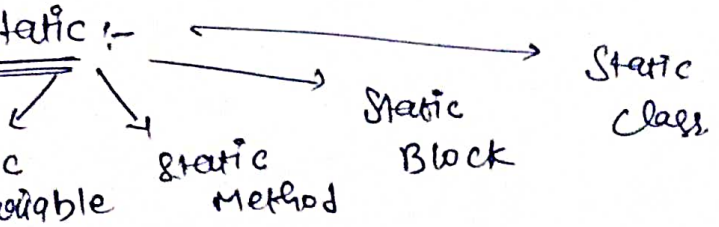
→ used to call default constructor of same class

```
class A{
  A(){
    Sout
  }
  A(int a){
    this();
    sout
  }
}
```

(call first)

→ allowed to call parameterized cons. of same class

(10) Static :-

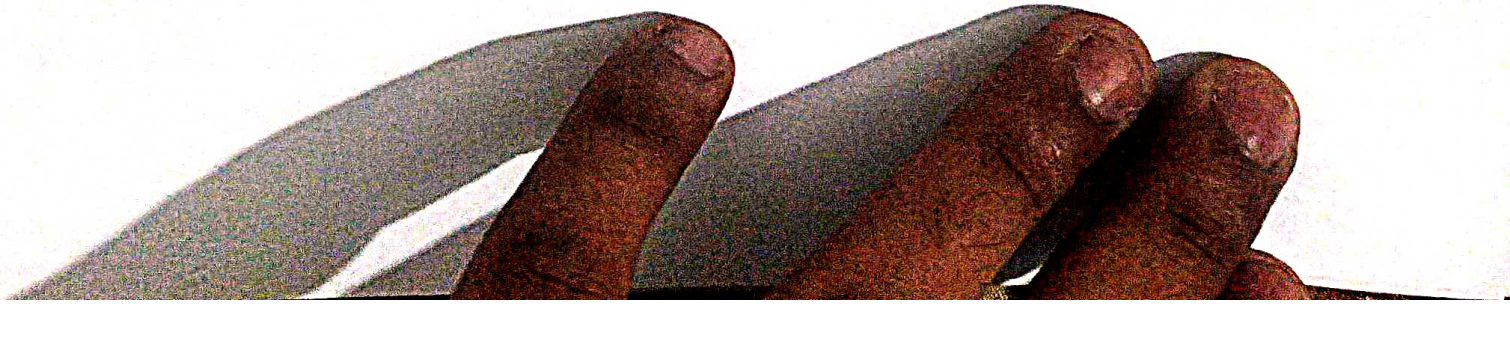


→ used for memory management
→ gets memory only once in the class & executes at two times of loading .class file.
→ it makes program memory efficient
→ we can access them directly using class name
→ Static method can't be overridden
→ abstract " " " Static.
→ Static method cannot use this & super keywords.
→ Class have multiple Static blocks.

Syntax:- class A{
Static {
}
}

→ we cannot use non-Static variables in Static methods.

Eg:- Class StaticDemo{
Static int a=4;
Static int b;
Static void display(int x);
Sout(x);
}
Static {
Sout
}
main
}



(11) Nested Class:-

→ Class within another class.
→ Nested class will access Private & public members But other classes only access public member.

→ Types of Nested class
  ↙ ↘
Static Non-Static

(12) Import keyword:-

→ used to import built in and User defined packages into your java code
→ all classes are imported by *.
→ Types of Import ↗ Static
                  ↘ Non-Static.

(13) pillars of OOPS:-

Abstraction
Inheritance Poly Morphism Encapsulation

Inheritance:-

→ The mechanism of a class to derive properties & character-stics from another class is called inheritance
→ Most important feature of OOPS.
→ reuse the properties of another class
→ Base class. --
→ Derived class.
→ is also known as Is a relationship.
→ using extends keyword we can implement it.

* Advantages:-

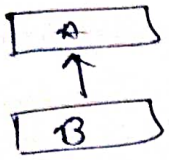
→ promotes reusability
→ generate more Dominant objects.
→ Avoid duplicity & Data redundancy
→

Properties of Inheritance:

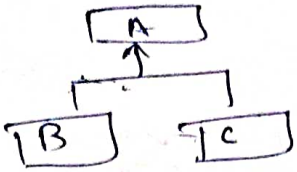
→ ~~A Sub inher~~ Super class can't inherit method of Sub class.
→ Sub class can't access private members of Base class A.
→ " access attributes & methods of ~~the~~ Super class

Types of Inheritance:-

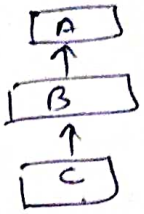
Single inheritance



Hierarchical



Multilevel



Multiple & Hybrid cannot used But we can use it with Interfaces.

Super keyword:-

→ refers the object of Super class
→ it is used when we want to call the super class variable, method & constructor through Sub class object.
→ when ever the Super class & Sub class variable & method name Both are same then it can be used only.
→ To avoid the confusion b/w Super class and Sub class variable and methods that have same name we should use Super keyword.

Super → variable
Super → method
Super → constructor

Syntax:

```
class A{
    A(){
    }
}
class B extends A{
    B(){
        super();
    }
}
```

eg:- class A {

int a=10;

}

class B extends A {

int a=20;

void show() {

System.out.println(super.a);

}

}

class Test {

main

B x = new B();

x.show();

}

point a=10

→ for methods.

Super.show();

→ for constructor

Super();

→ Most common application is remove ambiguity b/w members of superclass & subclass.

Access control :- restrict the scope of class, methods, constructor & uses to control access.

eg:- phone

Default :- it has no keyword.

→ The data members, class or methods having Default access modifier are accessible only within the same Package.

Package one;

class codeera {

void msg()

{

}

}

}

Package two;

import one.*;

class codeera2 {

main

codeera c = new codeera();

c.msg();

}

}

→ (gud eve)

Private → keyword is used private.
→ accessible only within the class
→ only visible in a enclosing classes.

Protected :- ~~new~~ prior
→ access in same package or subclasses in different package
→ apply only when inheritance is involved.

Public ~~—~~
→ accessible every where.

	Same class	Same package	Sub class	universal
private	✓			
Default	✓	✓		
Protected	✓	✓	✓	
Public	✓	✓	✓	✓

Polymorphism :- (one name many forms).
→ ability to take objects to many forms.
→ implements using the concept of overloading & overriding

Advantages :-
→ Single variable can be used to store multiple data types.
→ Easy to debug the codes
→ increases reusability

→ Implementation of Poly. :-

Method overloading

Method overriding

Method overloading → when two or more method have same name but differ. parameters,
→ Return type is alone not sufficient.

Method overriding :- If Subclass has the same method as declared in the parent class it is called

overloading	overriding
① Method with same name diff. Signature	Method with same name same Signature.
② Inheritance & Method hiding not involved here	Here Subclass extends the Superclass method
③ Performed within the class	It occurs in two classes with IS-A Relationship
④ Known at compile time	known as runtime.

final keyword.

→ Used for restrictions.
→ final can be
→ variable ⇒ you cannot change the value
→ Method ⇒ " " override it
→ class ⇒ " " extend it

Encapsulation → the action of enclosing something
→ Bundling of Data with the methods.
→ Wrapping up of data & functions into a single unit

Advantages :-
→ protect an object from unwanted access
→ reduce implementation errors
→ protection of data from accidental corruption
→ Makes easy to understand

Abstraction :- (Information hiding)

- Represents only essential & hides unnecessary info.
- → represents simplified view.
- → only shows data which is relevant to user
- → Implemented using abstract class

Advantage :-
- → reduce program complexity

# Abstraction	Encapsulation
(1) removing/ withdrawing Something unnecessary	act of Binding the Data & Secure it
(2) Applies at designing stage	Apply at implementation stage
(3) eg: Interface & Abstract class	eg: Access modifiers
(4) purpose/ Reduce code complexity	Purpose :- Data protection

Implementation of Abstraction :-

(1) Abstract class :- A class which contains the abstract keyword in its declaration is known as abstract class.

- → may or may not contain abstract method i.e, method without body
- → But if a class has at least one abstract method then the class must be declared abstract.
- → If a class declares abstract cannot be instantiated
- → To use abstract class, we have to inherit it to another class & provide implementations.
- → there is no object of abstract class
- → Cannot declare abstract constructor & static method

(ii) Interface :-

(1) An interface is similar to an abstract class with the following exceptions
   → All method defined in an interface are abstract
   → Doesn't contain any logical implementation
   → Cannot contain instance variables.

(2) Declare by interface keyword

(3) More abstract than abstract classes

(4) Implements using implements keywords

* Just like a class which contains only abstract Method
* Methods are by default public & abstract
* variables are " " public & static & final
* Methods must be overriden inside the implementing classes
* Deals with client & developer.
* one interface can be inherit another by use of keyword extends.

class Implements Interface
Interface Interfaces extends

Abstract	Interface
(1) Does not support multiple Inheritance	Supports multiple inheritance
(2) have abstract & non-abstract method.	have only abstract Method
(3) can have final, non-final Static & non-static variables	only static & final variables
(4) abstract class can extends another class & implement multiple interface.	can extend another Java interface only
(5) Class member like private & protected	Members are public By Default