# CORE JAVA

CORE JAVA:

* Methods: Methods are nothing but a block of statement or set of statement which will be executed whenever, it is called or invoked.

→ Why Method?

=X= We design for re-usability

=# General Syntax:

Access level, Modifiers, return type, Method type (Args)

- Public
- Protected
- Default
- Private

- Static
- Non-Static

- int
- float
- double
- Char
- boolean
- Java type
- or derived type
- void

- any identifier

- i/p Not mandatory

→ Returntype specifies what type of o/p that method returns after processing.

→ Return type and return value should match.

→ Return type can be any of the 8 primitive data types or Javatype (derived type) or void.

→ Method Name is any identifier, Any valid identifier we can give for MethodName.

→ If a method wants to add two number then it should be capable of receiving two number that is decided by i/p.

→ Program execution happens in RAM (Random Access Memory)

→ RAM memory will be divided into Stack Memory

& Heap memory.

→ Stack Memory is for execution and Heap memory is for Storage.

→ ~~(first JVM will be loaded on the stack to initiate execution process)~~ //

→ JVM will invoke the Class loader

→ Classloader will load all the static members of the class into static pool in the Heap memory.

→ Static pool will be named as the ClassName

→ JVM will check the availability of Main method in the static pool.

→ If Main Method is available it will take it and it will load it will load it on the stack for execution

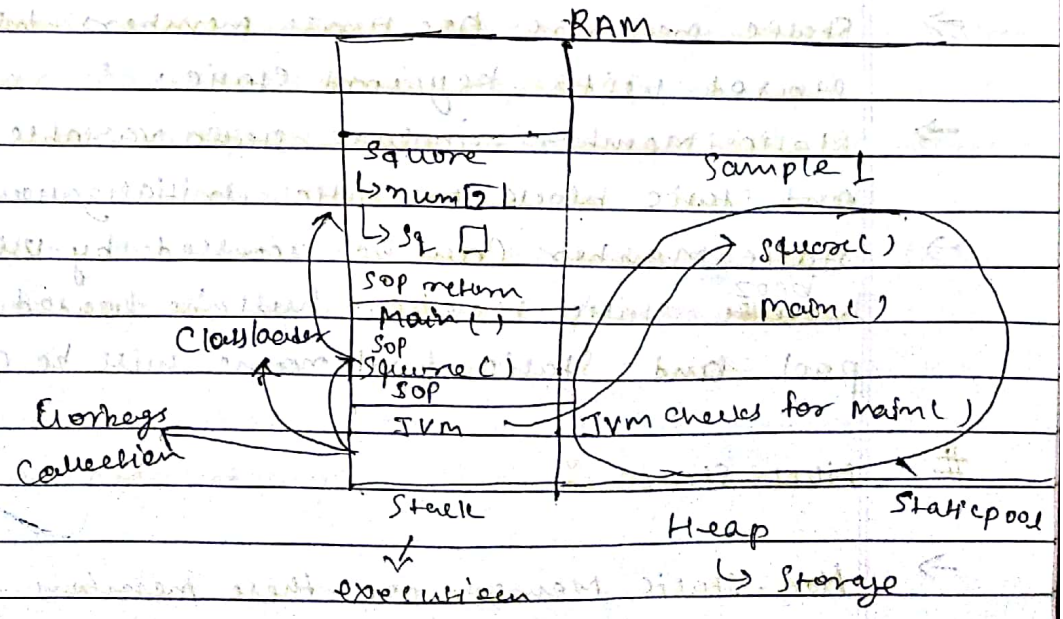
→ main() method will continue its execution and finally it will return the control both the JVM.

→ main() Method will be erased from the stack

→ JVM will Call garbage Collector (deamon thread) lowest priority thread.

→ garbage Collector will Clean (erase) the Content from Heap memory.

→ // (and finally JVM will be removed from memory)



→ Any method cannot be called outside main() only Constructor can be called

→ The result of comparision is either true or false.

a = b → Assignment operator

a == b → Comparision

⇒ ternary operator ?

(Condition) ? exp1 : exp2 optimised

// return (num1 != num2) ? 2*(num1 + num2) : (num1 + num2);

* Class :

→ Class is nothing but collection of fields and behaviour

→ field → variable attributes, data members behaviour → Methods, function, member function

→ whatever we write inside the class definition block are member of a class.

→ There are 2 types of members
- Static members
- Non-Static Members

→ Static members are those members which are declared with keyword Static.

→ Static members include static variable, static methods and static block or static initialization Block (SIB).

→ Static Member can be accessed by using ClassName, because (bcoz) static member will be loaded on to static pool and static pool name will be ClassName.

# NON-Static :

→ Non-Static Member are - these members which are not

declared using keyword static.

→ Non static Member includes non-static variable, non-static Methods, Constructors and Non-static block (or IIB instance initialization Block).

→ To access Non-static members we should create the object of a class or instance of class or we should instance of class or we should instanciate a class.

** default executable method = main()

ClassName referencevariable = new ClassName();

→ Here new is an operator and its operation is to the new address space or object in the heap memory randomly

→ the Name() will load all non-static members of that specific class into the new address space created by the new operator.

→ Reference variable will hold the object address

eg- Sample ss = new Sample();

→ Class loader will create the static pool only once in one execution cycle. So,

→ Static members will be having only one copy.

→ Non-static members can have many copies.

→ We can have multiple copies of Non-static member we can create multiple objects for a class, so we can have multiple copies of Non-static members.

→ Whenever we need single copy we go for static members eg: CompanyName, SchoolName..

→ If we need multiple copies we go for non-static member - eg- Employeeid, Studentid.

=> Object is an entity which will have its own state and behaviour.

Variable

Local variable

→ variable declared inside a method or block or in the method parameter.

→ It's scope is only inside the method or block in which it is developed

→ It cannot be accessed outside the method or block in which it is developed

→ we cannot categorise local variable into local or global

Global variable

→ variable which are declared outside of any method or block but within the class

→ global variable can be accessed from all parts of the program

→ global variable can be categorised into static and non-static.

Note :
→ Non-static variable are called as instance variable
→ static variable are called as class variable.
→ Method argument will be local
→ local variable are created on stack when method is invoked.
→ once method execution is over local variable is erased along with method and control returns to main method.
→ if we don't initialise any global variable (static or Nonstatic), it will be storing default value.

int = 0, double = 0.0, rr = null, ch = space, boolean = false

→ local variable if not initialised, will not be having default value.

→ So, before using local variable, we should initialise otherwise (CTE).

→ Before using any variable, it should have some value.

⇒ Non static variable of the class can be accessed directly from the static or non-static context (Method or blocks)

→ static variable cannot be accessed from static-context (Method or block).

* Constant variable:

→ final variable value cannot be changed but it can used any number of times.

→ If we donot have to re-initialise the value of a variable, then we go for final variable eg. PIe(π) = 3.14.

RULE:

→ We cannot just declare a final variable (global) (static Non-static), we have to initialize it at the time of declaration (CTE)

⇒ What is final?

→ final is a keyword, it is basically used to create constant variable.

→ If you want to declear constant variable in Java, we will declare that variable with final keyword.

→ final is a keyword, final variable value can't be changed or overwritten.

→ Both global and local variable can be final.

Rule :→ global final variable should be initialized at the time of declaration itself (Non static final can be initialized inside constructor).

→ Local variable can be declared once & initialized later (anytime).

## METHOD OVERLOADING

→ Developing Multiple method with same Name by changing the argument list is called as Method overloading.

→ Changing the Argument list means
(I) Changing the Number of Arguments
(II) Changing the data types of the Arguments
(III) Changing the position of the arguments.

→ If there is a common task to be performed with the variation in the i/p, then we should go for Method Overloading.

e.g: If you want to write a program which add number and also 3 number then we should develop two methods 1) first one should receive 2 i/p.
2) 2nd one should receive 3 i/p.

→ Instead of designing these 2 methods with different name, we can develop the 2 method with same name, which is called as Method Overloading.

1) We can achieve consistency in method Names which are designed for common purpose.
2) We can achieve efficiency in program readability
3) We can remember method names easily.

4). We can achive compile time polymorphism.

→ We can overload both static & non static methods

→ If the method names are same then there should be same variations in the arguement and there is no restrictions on other parts of the methods while overloading.

Q. Can we overload the main method ?

Ans yes, like any other method main() method can also be overloaded, but the programm execution starts from main() method which is having arguement as (String[] args) and other restrictions of main() Method should be called (invoked) explicitly.

* Constructor Chaining : It is the process of calling one constructor from another constructor with respect to current object. It can be done in two ways. 1) Within same class 2) From base class can be done → this() keyword → super() keyword.

⇒ Constructor

⇒ Constructor is a special method, hence it is used to Construct / Creat an object.

→ Whenever we are creating object for a class by using new operator, Constructor will be executed.

→ Constructor will have same name as class Name.

→ Constructor will not have any return type explicitly.

→ Constructor will not return any value.

→ Constructor will always be non-static, it is also called as non-static initializer, hence is used to initize the object.

→ if a programmer doesnot write or develop a Constructor for a class then that will be

default Constructor.

There are four steps in object creation.

① New operator will create a new memory or Address space in heap.

② All the Non-Static members of the class will be loaded with its initial value.

③ Constructor body will be executed on the stack like a method.

④ Address of the object will be ~~executed~~ stored in the referenced variable.

⇒ There are two types of Constructor;

① parametrised or Arguminated Constructor.

② No Args Constructor or Constructor without parameter.

| | Method | Constructor |
|---|---|---|
| ① | Methods Can have any Name i.e it Can have ClassName as well | ① Constructor Name should always be ClassName. |
| ② | Methods should have return type | ② Constructor should never have return type explicitly |
| ③ | Methods Can be Static or Non-Static. | ③ Constructor should always be non-Static |
| ④ | Method may or may not return value based on return type | ④ Constructor Can never return value. |
| ⑤ | There will be no default method | ⑤ there will be default Constructor if the programmer has not developed. |

⇒ Why Constructor?

→ Constructor is necessary to Construct the object.

Q. Why programmer should develop the constructor if Java provides a default Constructor?

→ To Construct the object in our own way, programmer should develop the Constructor.

(1) May be to initialize the instance variable at the time of object creation itself. (Using Parameterized Const)

(2) To execute Certain Statement at the time of object creation eg: forefor deiver class of selenium.

→ If the programmer is developing a constructor for a class then, there will not be default Constructor.

→ If you want to create an object for a class you should utilize the variable Constructor of the class.

* Constructor overloading:

→ Developing multiple Constructor for a class with a variation in datatype, no. of argument and position of argument is called Constructor overloading.

** Advantage of constructor overloading

→ We will get multiple options to create an Objects for a Class.

* this() Statement

→ this() calling method Statement is used to call one Constructor form another Constructor in case of Constructor overloading.

→ If you want to utilise the initialization code written in the other constructor of the same class then we can use this( ) calling statement.

→ this(Parameter) will call the current class constructor with matching parameter
e.g. this(90) will call the current class integer constructor.

⇒ We can call constructor by its name at the time of object creation with new operator eg. new A()

→ We can call constructor through this( ) calling statement.

→ Generally for one object creation only one constructor will be executed, if you want to execute other constructor of the same class for one object creation then we should use this( ) calling statement.

Rule ① this( ) calling statement can be developed inside constructor (it cannot be developed in any other method apart from constructor.

② this( ) calling statement should always be first statement inside a constructor.

→ We can develop multiple this( ) calling statement inside a single constructor (max one → 1 or 0)

* this keyword :

→ This keyword is a default reference variable in Java.

→ If you want to access the current instance member then we should use this keywords

→ this will be pointing to the current instance member,

→ this will be pointing to the current object

→ always we keep variable name same for readability purpose.

this.i = i

## Business Class and main Class :

① → In one java file we can develop multiple class.

→ The class in which we develop business logics is called as Business Class.

→ The class in which we will write main method is called as main class.

② If there are multiple classes in a java file then file name should be main classname.

③ whenever we are compiling any java program which is associated with multiple class then as a result of compilation, multiple class (.) dot class file will be generated based on class Name.

④ If you want to access, the static member of other class, then we should use that ClassName.

e.g. thatClassName.variableName.

⑤ If you want to access, the NON static member of other class, then we should create that class object.

⑥ whenever we are running any java programm which is associated with multiple classes, then multiple static pools will be created based on the classname.

→ All the static member of each class will be loaded on their respective static pool.

Q. Can we create or develop multiple main classes in our Java file?

Ans: Yes, we can develop multiple main classes in one Java file, in this case Java file name can be any one of them and program execution starts from dot class file that we are running.

## Inheritance

Q.1 What is an Object?

Ans: Object is an entity which will have its own state and behaviour

→ Any object oriented programming language will have four principle:

1. Inheritance
2. Polymorphism
3. Encapsulation
4. Abstraction

⇒ Inheritance: Inheritance is one of the principle of the object oriented programming which represents "Is-A" relationship.

→ Inheritance means acquiring the features or properties of one class into another class (w.r.t Class)

→ The class from which we are getting the features is called as parent class / Base Class or super class

→ The class which have acquired the features is called as child or derived Class or Sub class.

→ Through inheritance we can achieve extensibility i.e We can achieve Code re-usability
Code maintainability.
Code optimisation.

→ To represents Inheritance b/w class we are using extends keywords.

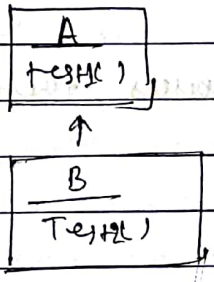
→ Inheritance is only for Non-Static members

→ Order of deciding of Non-Static Member

=> Inheritance is many types Mainly four

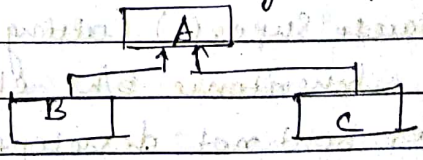
(1) Single level Inheritance
(2) Multi level Inheritance
(3) Hybrid Inheritance
(4) Multiple Inheritance //

=> 1) Single level Inheritance : One class having only one super class is called a single level inheritance.

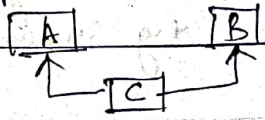



=> 2) Multilevel Inheritance : One Sub class inheriting from another Subclass is called as multi level inheritance.


=> 3 Hybrid Inheritance : lot of Sub classes having Common Super class is called as Hybrid Inheritance




=> 4 Multiple Inheritance : One class having Multiple immediate Super class is called as Multiple Inheritance.



In Java multiple inheritance is not possible through classes.

* Super Calling Statement :

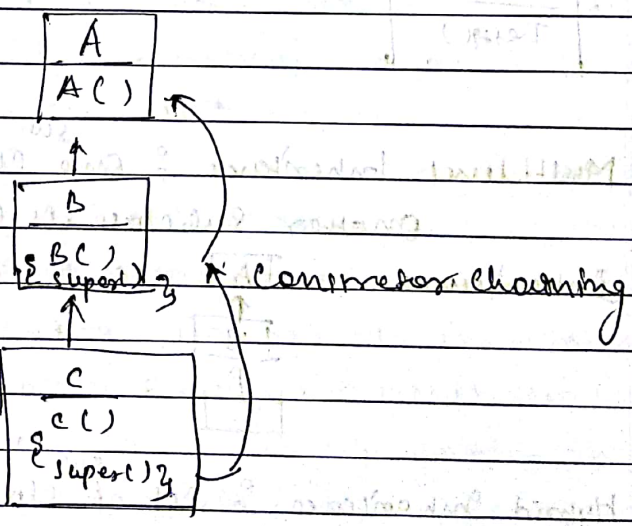
(1) super() calling statement is used to call super class constructor in case of inheritance.

(2) super() calling statement will call the immidate super class constructor.

(3) Through super() calling statement, we can achive constructor chaining.

(4) Subclass constructor calling it's super class constructor is called as constructor chaining.

Rule: 5 If there is inheritance b/w classes, then constructor chaining is mandatory.




* Default super() Calling Statement : If there is inheritance b/w classes and, if the programmer had not developed super() calling statement inside a constructor then compiler will develop a default super calling statement.

Eg super()

→ Default Constructor with default super() Calling Statement for inheritance.

→ Super(parameter) will call the super class Constructor with the matching parameter
e.g. Super(90) will call the super class integer Constructor.

* Rule of Super()

→ Super() Calling Statement should be the first Statement inside Constructor.

→ We Can write Maximum one super() Calling Statement inside a Constructor.

→ We Cannot develop both this() and super() Calling Statement inside the same Constructor because both should be the first Statement inside the Constructor.

→ if there is inheritance b/w classes and the programmer has not developed either this() or super Calling Statement then we have default super() Statement.

⇒ Similarities b/w this() and super()

(1) Both are used to call Constructors
(2) Both Can be developed inside a Constructor only
(3) Both should be the first Statement inside Constructor

⇒ Difference b/w this() & super()

| this() | super() |
|---|---|
| (1) this is used to call Current Class Constructor | (1) super() is used to call super class Constructor. |
| (2) this() is used in case of Constructor overloading | (2) super() is used in case of Inheritance. |

| | |
|---|---|
| (3) there will be no default this() statement | (3) there will be default super() calling statement |
| (4) through this() calling statement we can use the initialization code of other constructor of the same class | (4) Using super() calling statement we can Archive constructor Chaining. |

## Object class :

→ Object is the supermost class for any class in java.

→ object is a built in class in java.

→ object class stands at the top most level in inheritance hierarchy.

→ There are lot of non-static method which is already developed in object class.

e.g: finalize(), clone(), ~~hashcode~~ hashcode(), equals(), toString(), ~~to String()~~, wait(), notify(), notifyAll(), getClass().

→ Any class in java will inherit all the non-static method of object class.

→ Any object in java will have all these non-static member of object class.

NOTE: Any class (is a subclass to the object class) means any programmer developed class or ~~any~~ another built-in class e.g - String class.

→ Q. why java doesnot support multiple inheritance through class?

→ multiple inheritance means one class having

Multiple immediate Super Class.

→ If there is inheritance b/w classes, there should be Constructor Chaining.

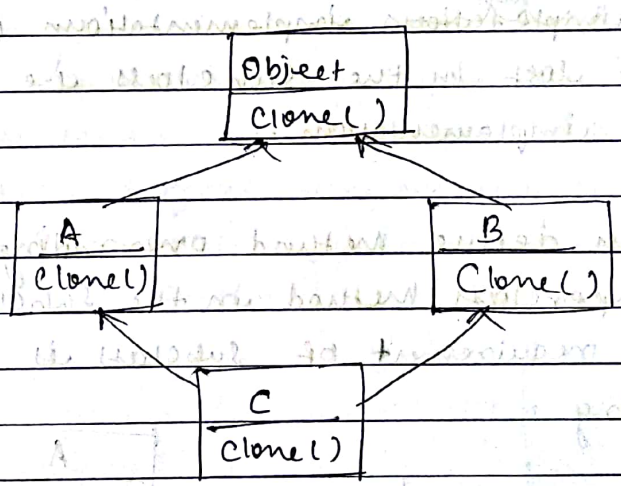
→ Constructor Chaining can be achieved through super() Calling Statement.

→ In this case we may have to develop multiple super() Calling Statements inside a single Constructor which is against the rule of Java because super() Statement should be the first Statement.

(11) Diamond problem/path issue.

→ In case of multiple inheritance multiple paths will be established from one class to Object Class (another class).

→ Generally, Object features will be inherited to one class in one single path.




In multiple inheritance, it (feature) of Object Class) will be inherited to the same class multiple times which is not possible.

→ This is called as Diamond problem/or multiple path issue.

# Method Overridding

→ Changing the implementation of Super class method in sub class according to the need of the subclass is called as method overridding.

→ Through method overridding, we can achive Run-time Polymorphism.

→ To perform method overridding, these things are Mandatory
1. Inheritance
2. Non-static Method
3. Method Signature (Method declaration part) should be exactly same

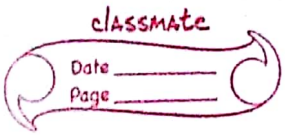
→ When we perform method overridding, super class new implementation of that method will be lost in the subclass i.e we will get the latest implementation.

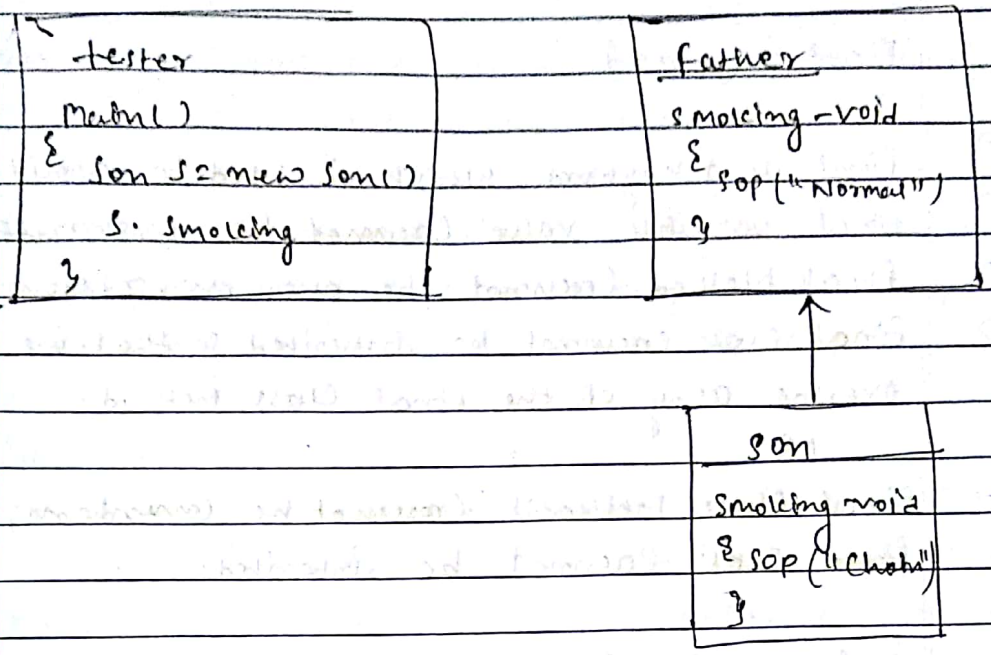
→ We can define method overridding as re-defining the superclass method in the subclass according to the requirement of subclass is called as method overridding.

```
tester
Main()
{ B b1 = new B();
  b1.test
```

```
A
test() - void
{ sop }
```

```
B
test() - void
{ sop }
```




* Super keyword :→

→ Super keyword is used to access super class non-static member in case of inheritance.

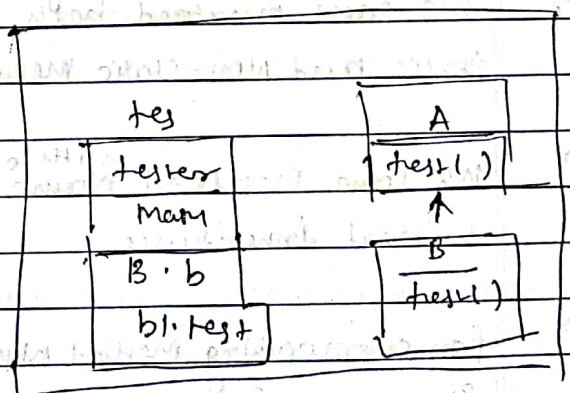
→ Effective Usage of super keyword is in case of Method overriding.

→ Due to method overriding, super class implementation of that method will be masked with the subclass.

→ To get the masked implementation, then we can use super keyword.

e.g:

Class A




Another example :