CHANDAN MUKHERJEE

MTech (IT), BE (Computer Science)
SCJP (Java), Oracle (SQL) GLOBAL CERTIFIED
Microsoft Certified Innovative Educator
Corporate Trainer
chan.muk@gmail.com

JAVA OOP

Object Oriented Features

- Encapsulation and Data Hiding
- Polymorphism
- Inheritance
- Abstraction
- Modularity
- Dynamic Initialization
- Message Passing
- Class
- Object

CONCEPT OF CLASS AND OBJECT

HOUSE – have any physical existence?

Java Class & Objects

Class

Person

Data Members

unique_id
name
age
city
gender

Methods

eat()
study()
sleep()
play()

name- John
age- 35
city- Delhi
gender- male

name- Dessy
age- 20
city- Pune
gender- female

Breed: Bulldog
Size: large
Colour: light gray
Age: 5 years

Dog1Object

Breed: Beagle
Size: large
Colour: orange
Age: 6 years

Dog2Object

Breed: German Shepherd
Size: large
Colour: white & orange
Age: 6 years

Dog3Object

Dog

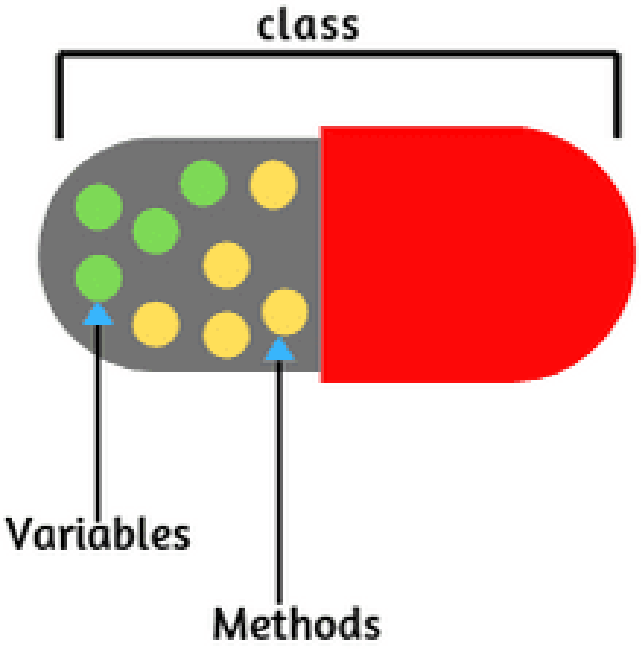
Fields
Breed
Size
Colour
Age

Methods
Eat()
Run()
Sleep()
Name()

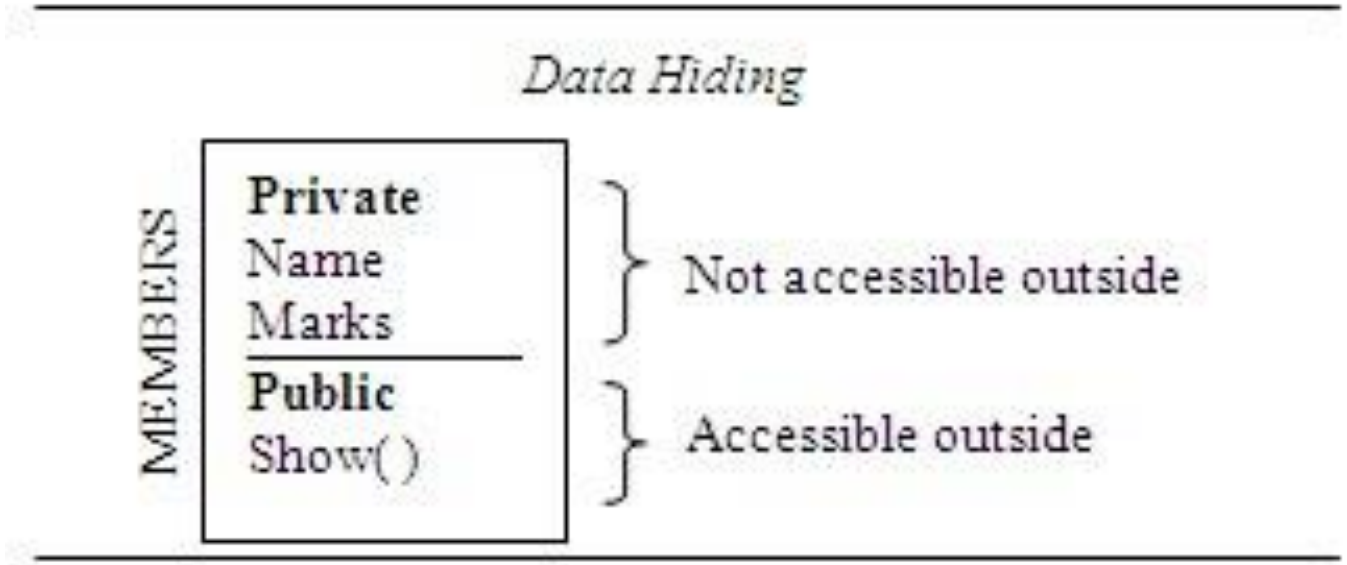
- PROGRAM

ENCAPSULATION

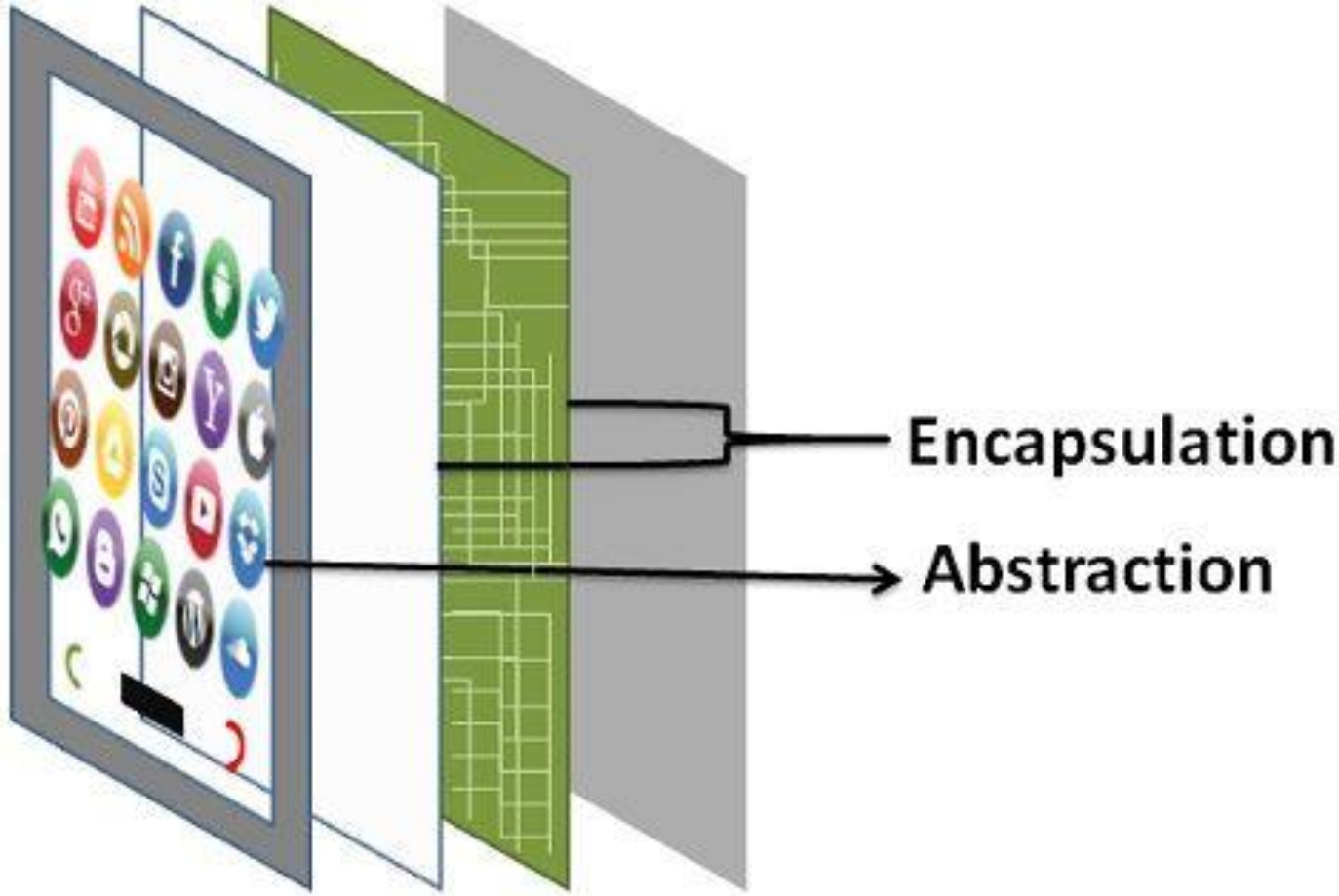
```
class
{

   data members
          +
   methods (behavior)

 }
```



DATA HIDING


Data Hiding
MEMBERS
Private
Name
Marks
Public
Show( )
Not accessible outside
Accessible outside

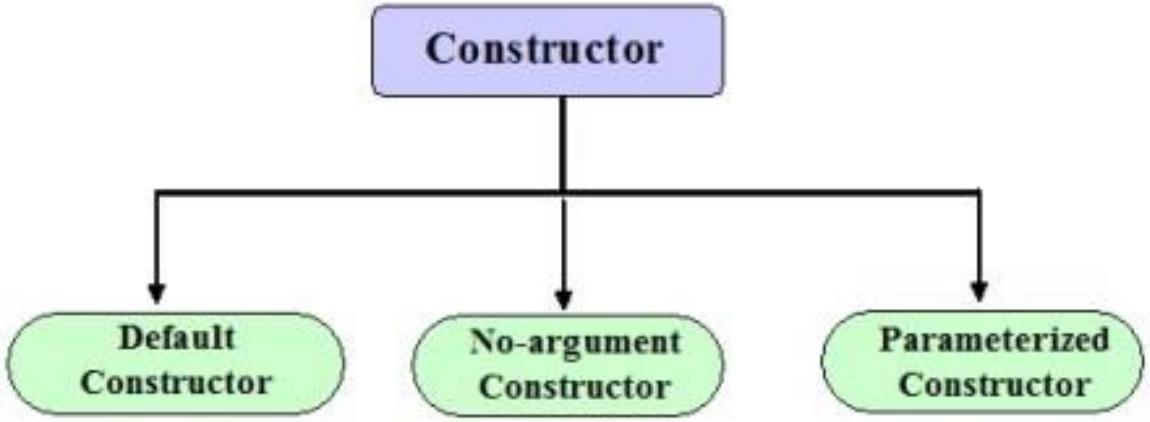
ABSTRACTION



- PROGRAM

CONSTRUCTOR

- CLASS NAME & METHOD NAME SHOULD BE SAME
- NO RETURN TYPE
- IF WE ADD RETURN TYPE THEN IT WILL NOT CONSIDER AS A CONSTRUCTOR IT WILL ACT AS NORMAL METHOD.
- IT ONLY IMPLICITLY EXECUTED WHEN ANY OBJECT IS CREATED.
- IT'S JOB IS TO INITIALIZE THE VARIABLES BY 0 OR ANY USER DEFINED VALUE



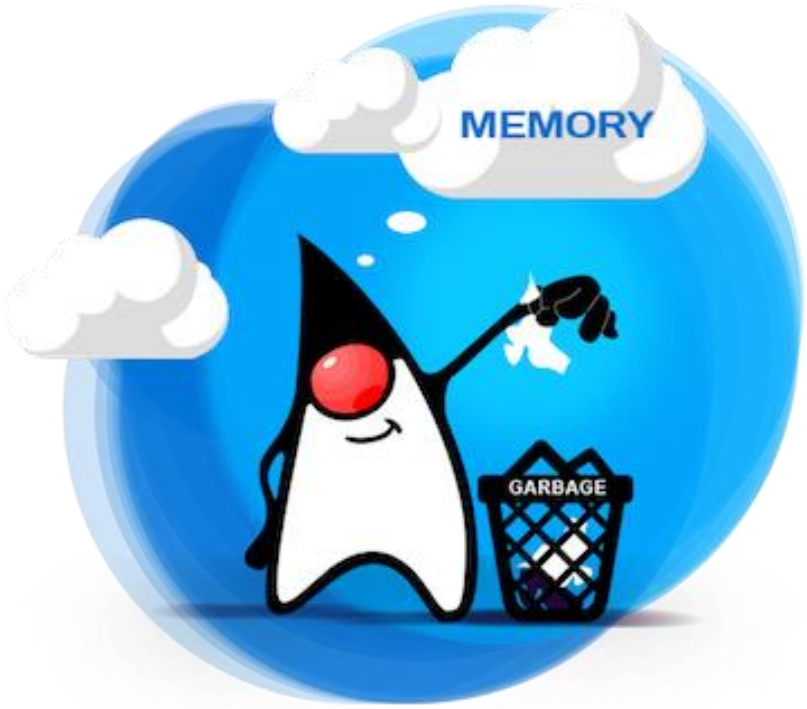
CONSTRUCTOR

- TWO TYPES :- **SYSTEM DEFINED (DEFAULT)** AND **USER DEFINED**
- SYSTEM DEFINED CONSTRUCTOR ALWAYS NO PARAMETER/ARGUMENT
- USER DEFINED CONSTRUCTOR WITH PARAMETER/ARGUMENT OR WITHOUT PARAMETER/ARGUMENT
- IF USER DEFINED CONSTRUCTOR IS PRESENT THEN SYSTEM DEFINED CONSTRUCTOR WILL NEVER BE EXECUTED.
- SYSTEM DEFINED CONSTRUCTOR (NO ARG) IS ADDED BY COMPILER
- MULTIPLE CONSTRUCTOR CAN BE PRESENT WITHIN A SINGLE CLASS

- PROGRAM

	Constructor	Method
Name	Constructor's name must be same as the name of the class.	Method's name can be anything.
Return Type	Constructor doesn't have a return type.	Method must have a return type.
Call	Constructor is invoked implicitly by the system.	Method is invoked by the programmer.
Main Job	Constructor can be used to initialize an object.	Method consists of Java code to be executed.
Overload	Constructor can be Overload.	Method also can be overload.

Garbage Collector



TO CALL GARBAGE COLLECTOR
System.gc()

Phase in Garbage Collection

Marking Phase
(List of live objects created)

↓

Relocating Phase
(References for list of live ojects updated)

↓

Compacting Phase
(Dead objects released & live objects compacted & moved)

Garbage Collector

- Garbage collection is the process which reclaims the memory allocated by objects which are unreferenced .
- Since objects are stored in the heap memory it can be considered as a mechanism to reclaim the heap memory
- Garbage collector is a JVM daemon thread which perform the task of garbage Collection
- An object is a candidate for garbage collection if and only if there is no reference to it.
- Any object becoming unreferenced does not mean that it will be immediately garbage collected , it only means that the object is candidate for getting collected in the next garbage collector cycle.
- JVM runs the garbage collector based on several predefined algorithms mostly when it is in need of memory

- PROGRAM

Finalize Method

- Before an object is garbage collected, the runtime system invokes the objects ***finalize()*** method.
- The finalize() can be used to perform clean up activities such as release system file resources (or) close sockets (or) close database connections etc.
- The finalize() method is declared in the ***java.lang.Object*** class,

Example

```
protected void finalize(){
//close resource }
```

Note : It is important to understand that finalize() is only invoked prior to garbage collection. It is not invoked immediately when an object goes out-of-scope. So programmers should provide other means of releasing system resources used by the object. It must not rely on finalize() for normal program operation.

- PROGRAM

METHOD OVERLOADING

FETURES OF OVERLODING:

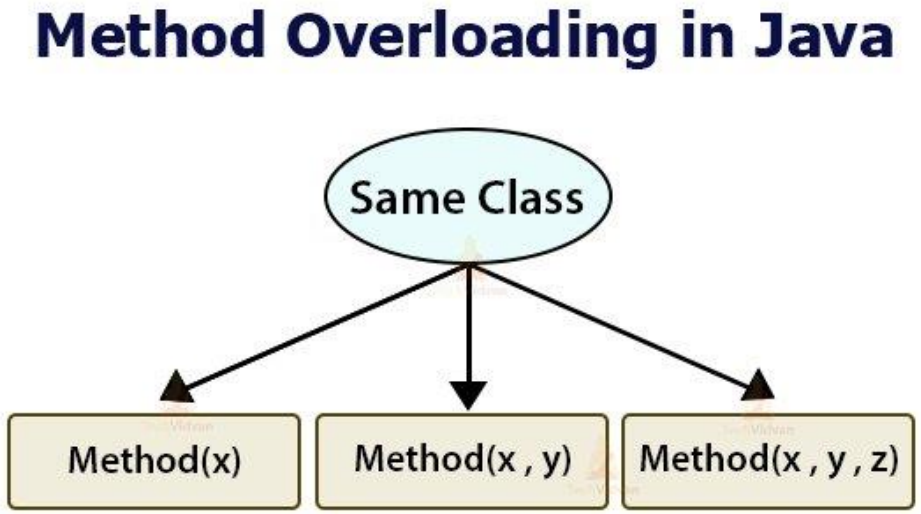
SAME METHOD NAME,

DIFFERENT PARAMETER LIST.

IF NO OF PARAMETER IS SAME THEN DATA TYPE SHOULD BE DIFFERENT OR PARAMETER SEQUENCE IS DIFFERENT

IT DOES NOT DEPENDS ON RETURN TYPE & ACCESS SPECIFIRE.

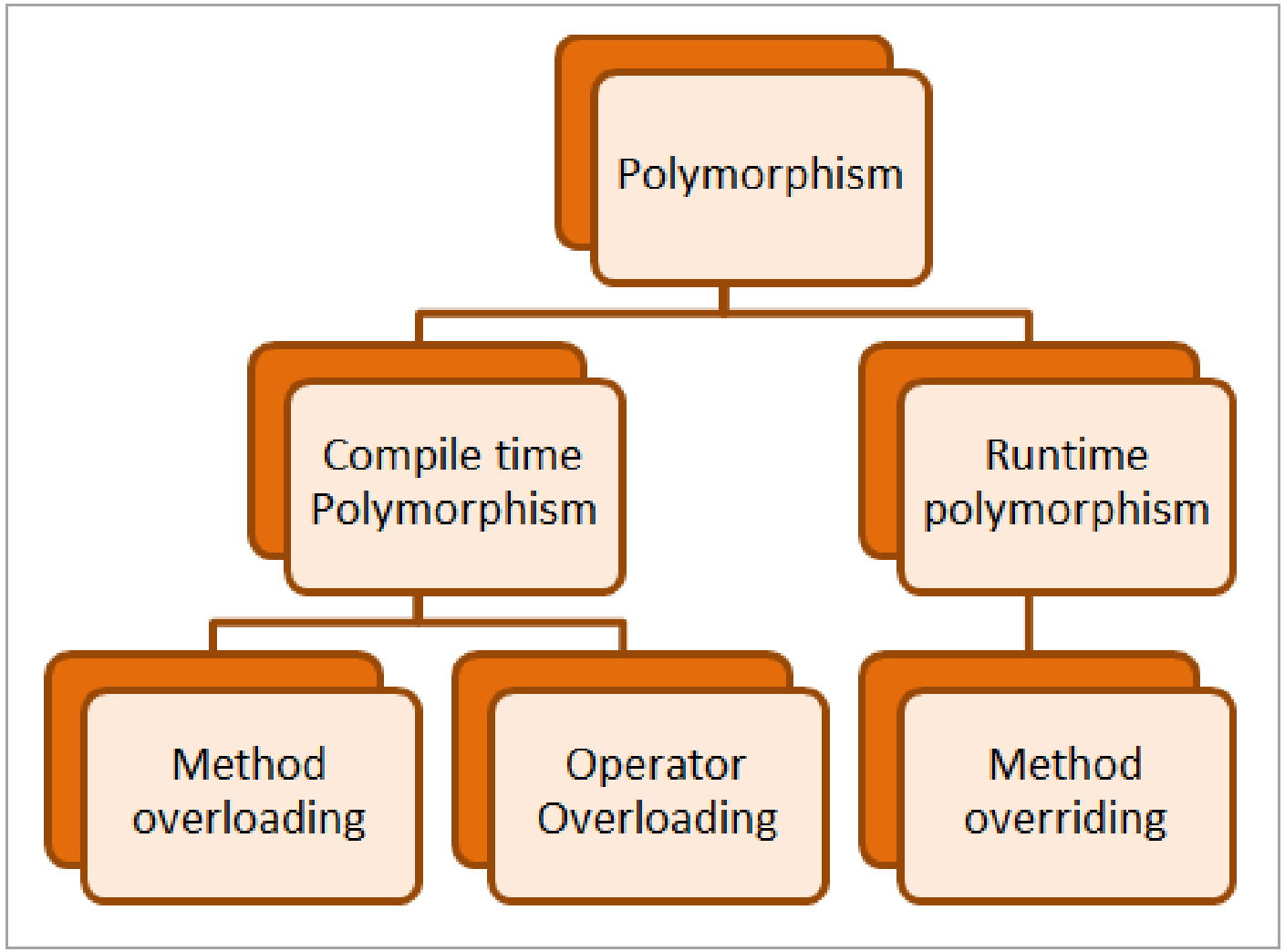
ALL METHONDS PRESENT WITHIN SAME CLASS OR PARENT CHILD RELATIONSHIP, THEN WE CALLED IT AS OVERLOADING.



POLYMORPHISM

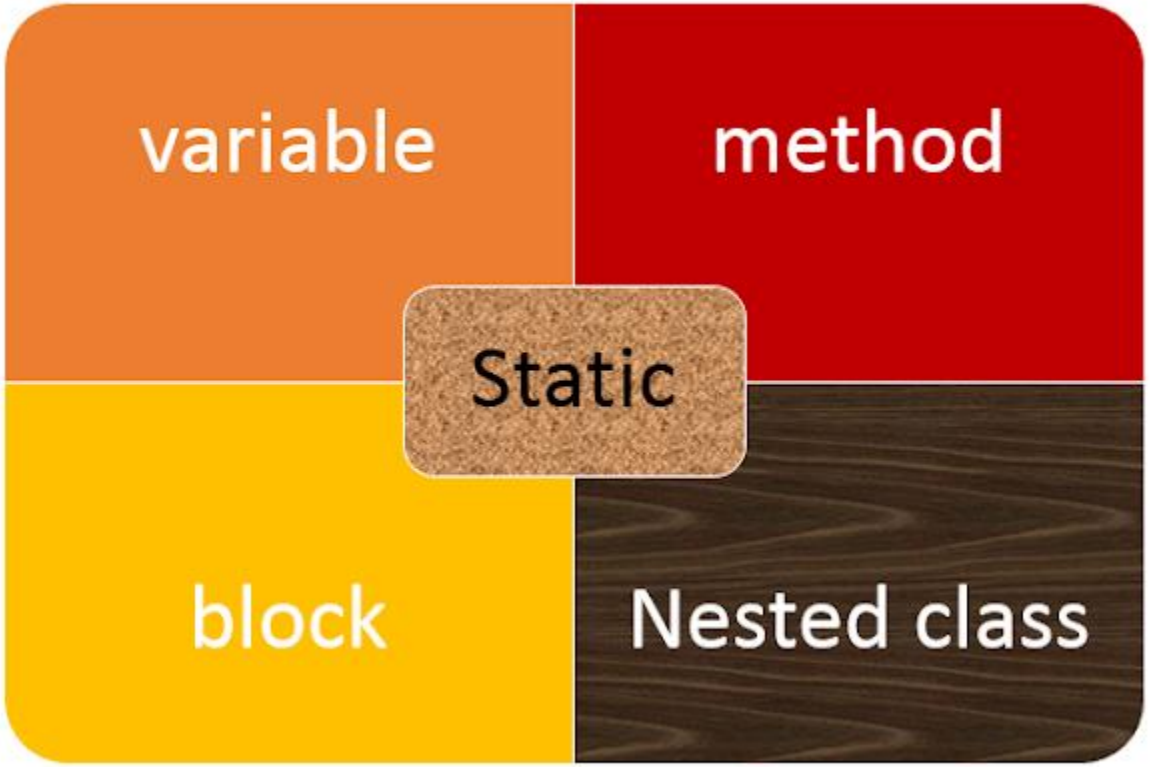
Polymorphism

POLYMORPHISM TYPE



- PROGRAM

STATIC KEYWORD



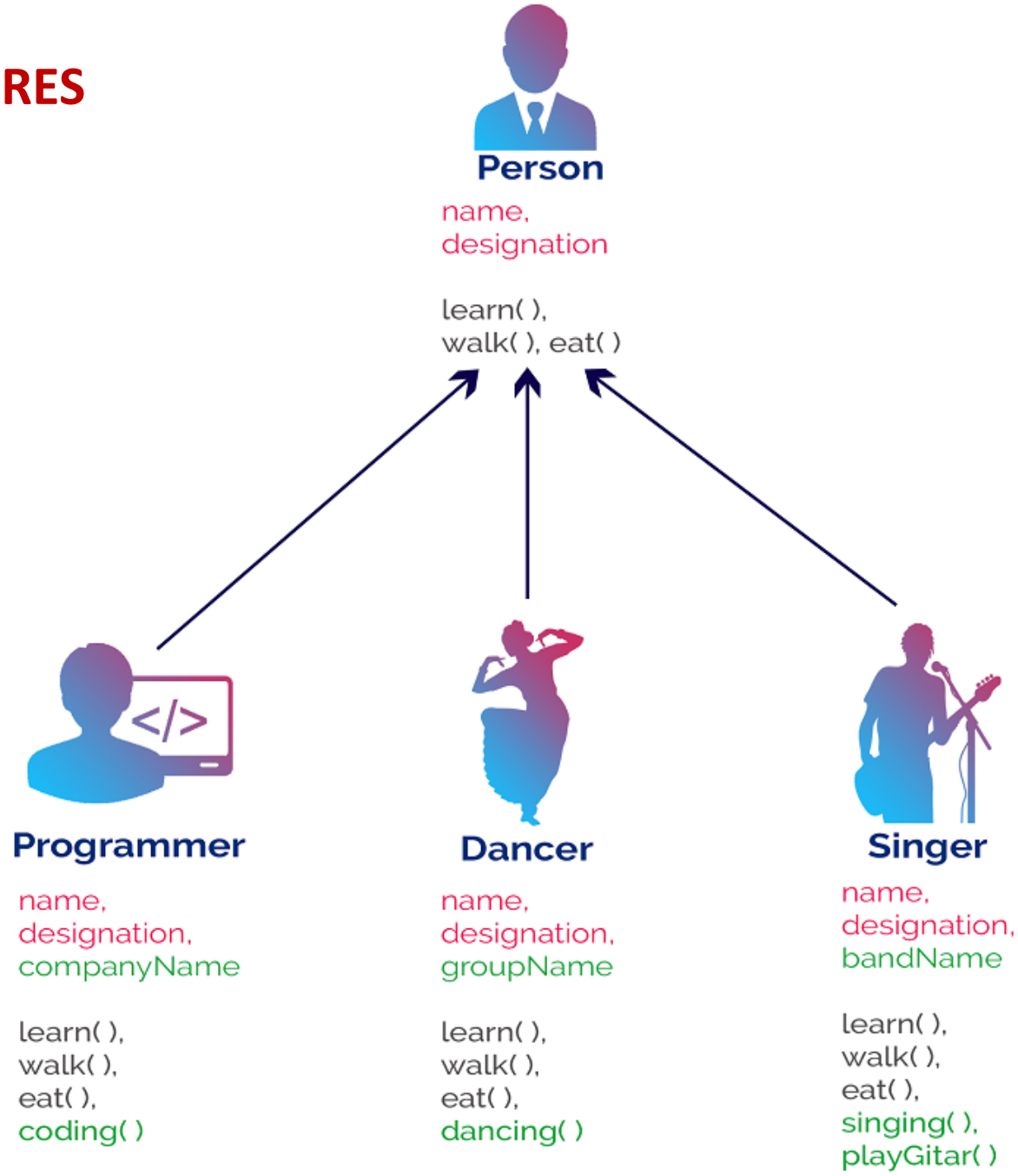
STATIC KEYWORD FEATURE

- STATIC IS KEYWORD THAT CAN BE WRITTEN BEFORE VARIABLE, METHOD & INNER CLASS.
- STATIC KEYWORD CAN NOT BE WRITTEN BEFORE OUTER CLASS.
- ONLY A SINGLE MEMORY IS ALLOCATED FOR A SINGLE STATIC VARIABLE. ALL OBJECTS OF THAT CLASS SHARE THAT MEMORY.
- STATIC VARIABLE MEMORY IS ALLOCATED WHEN THE CLASS IS LOADED INTO MEMORY BY BOOTSTRAP LOADER.
- TO CALL A STATIC METHOD, WE DON'T NEED TO CREATE ANY OBJECT. WE CAN CALL IT BY CLASSNAME.METHODNAME() [A.F1()]
- FROM A STATIC METHOD, WITHOUT CREATING OBJECT WE CAN DIRECTLY CALL ANOTHER STATIC METHOD ONLY.
- STATIC METHOD CAN ONLY USE STATIC VARIABLES.

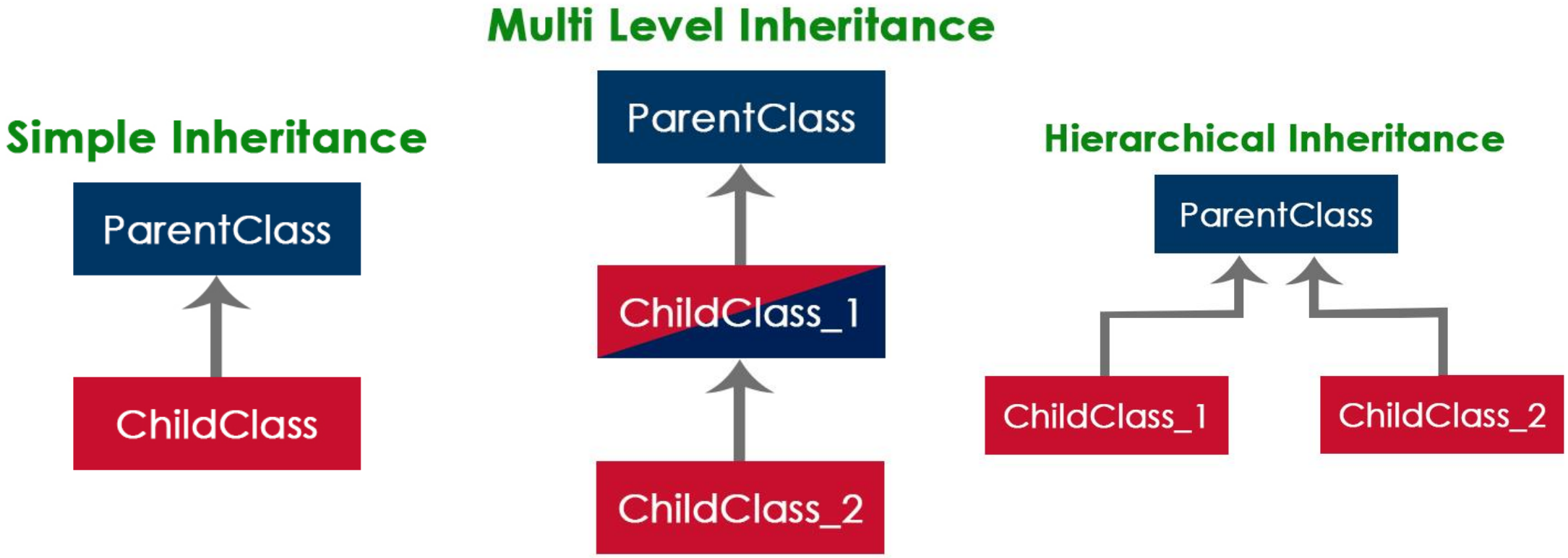
- PROGRAM

INHERITANCE

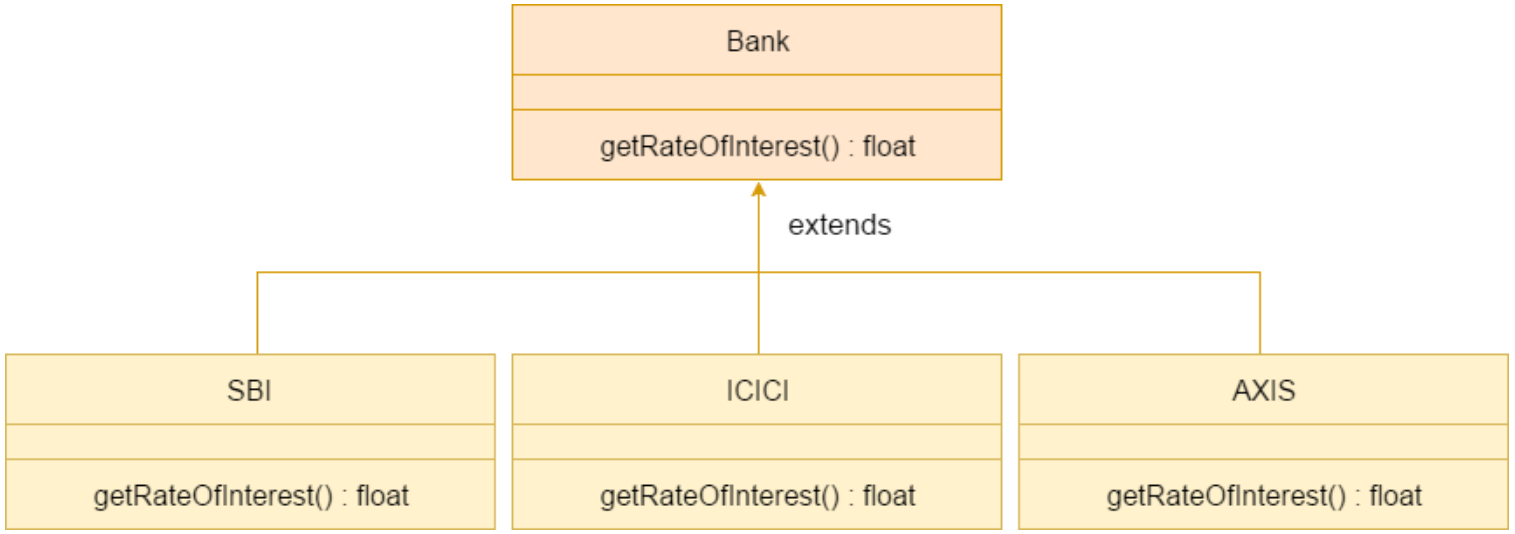
INHERIT FEATURES



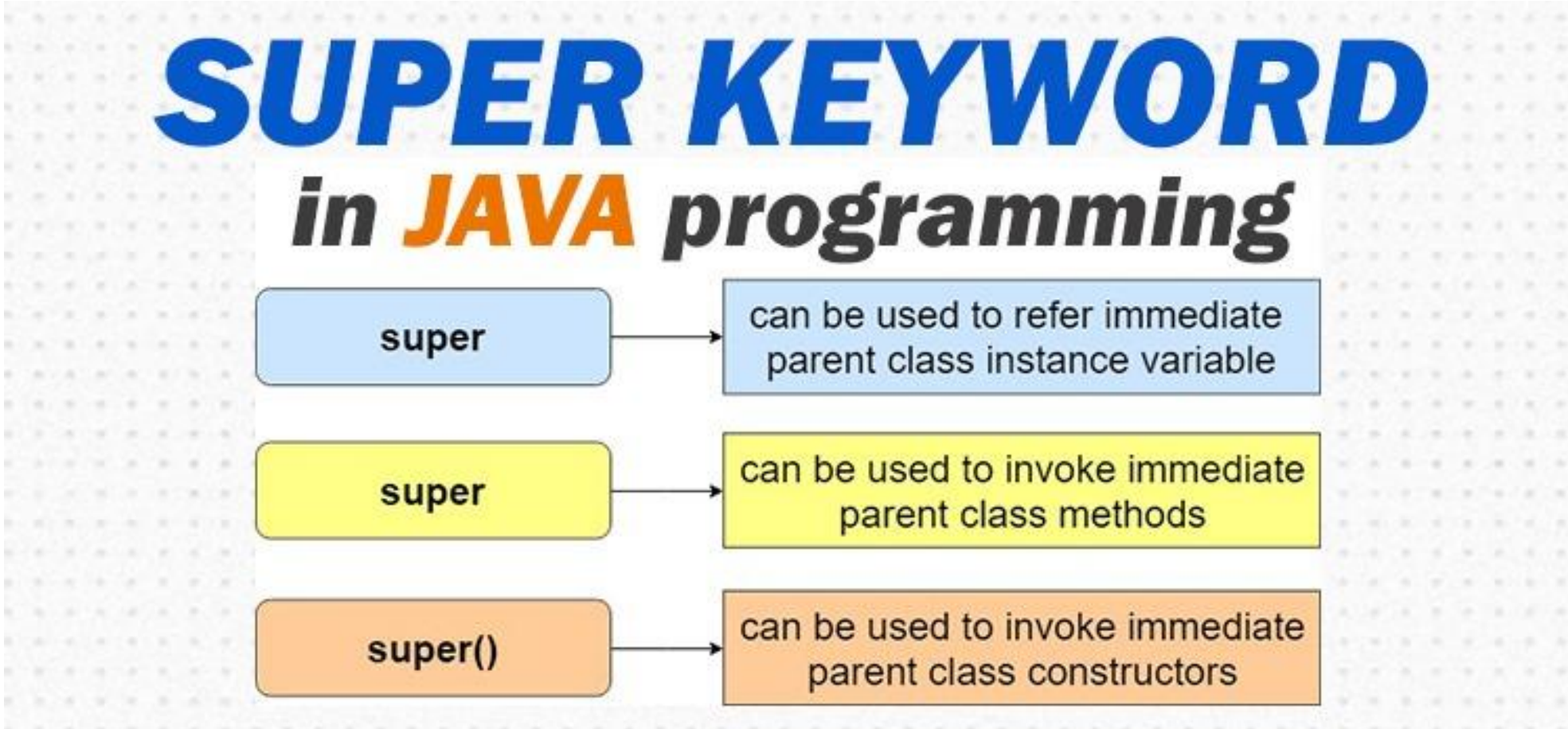
INHERITANCE TYPE FOR JAVA CLASS



OVERRIDING



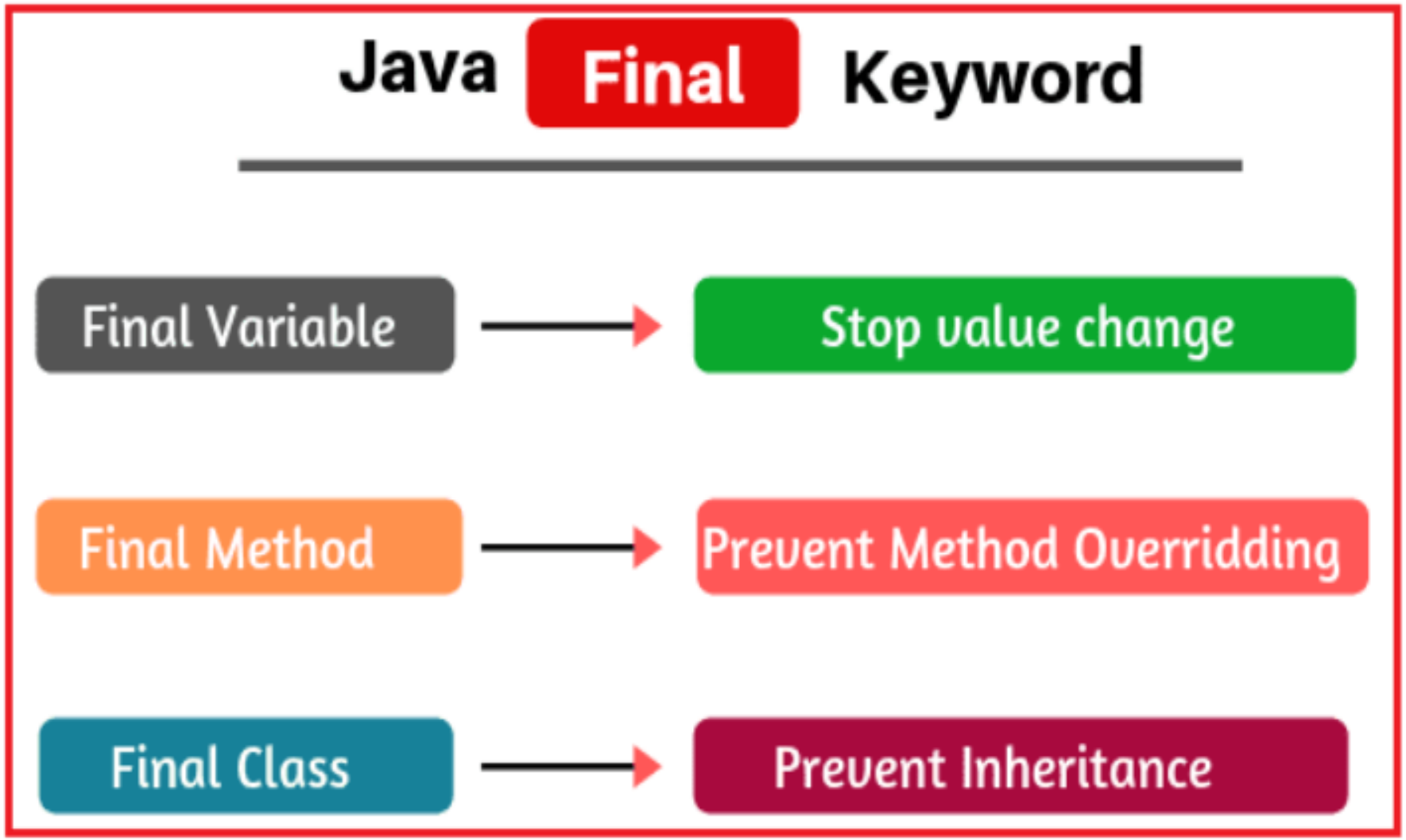
- ✓ Same method name.
- ✓ Same parameter list.
- ✓ Same return type.
- ✓ Same access specifire or widening access specifire.
- ✓ One method is in parent class & another method is in child class.


SUPER KEYWORD
in JAVA programming
super
can be used to refer immediate parent class instance variable
super
can be used to invoke immediate parent class methods
super()
can be used to invoke immediate parent class constructors

- PROGRAM

OVERLOADING vs OVERRIDING

Method overloading	Method overriding
It is possible only in same class.	It is possible only in derived classes.
Static methods can be overloaded	The method must be a non-virtual or static method for overriding.
Also known as static binding or early binding.	Also known as dynamic binding or late binding.
Used to implement compile time polymorphism	Used to implement run time polymorphism
It has same method name in same class with different signatures	Derived class has same method name with same signature as of base/parent class.
It helps to extend functionalities	It helps us to overwrite or change the existing functionalities.


Java Final Keyword
Final Variable
Stop value change
Final Method
Prevent Method Overridding
Final Class
Prevent Inheritance

JAVA FINAL VARIABLE NAMING CONVENTION

Constant Variable - Constant variable names should be written in upper characters separated by underscores.

- Create a base class called Vehicle that stores number of wheels and speed. Create the following derived class:
- Car that inherits Vehicle and also stores number of passengers.
- Truck that inherits Vehicle and also stores the load limit.
- Write a main () function to create objects of these classes and display all the information about Car and Truck. Also, compare the speed of the two vehicles.