

# Object Oriented Programming in Java

## Interview Questions

- Q1) (a) What are the differences between Procedural Programming and Object oriented programming languages?
- (b) What are the drawbacks of procedural programming language?
- (c) What are the needs/advantages of object oriented programming language?

## Procedural Programming

- Focuses on functions & logic!
- less reliable & secure
  - ↳ No access modifiers
- less reusability & difficulty in maintaining the codebase
- Example: → C Programming language

Note: → Javascript is functional programming language.

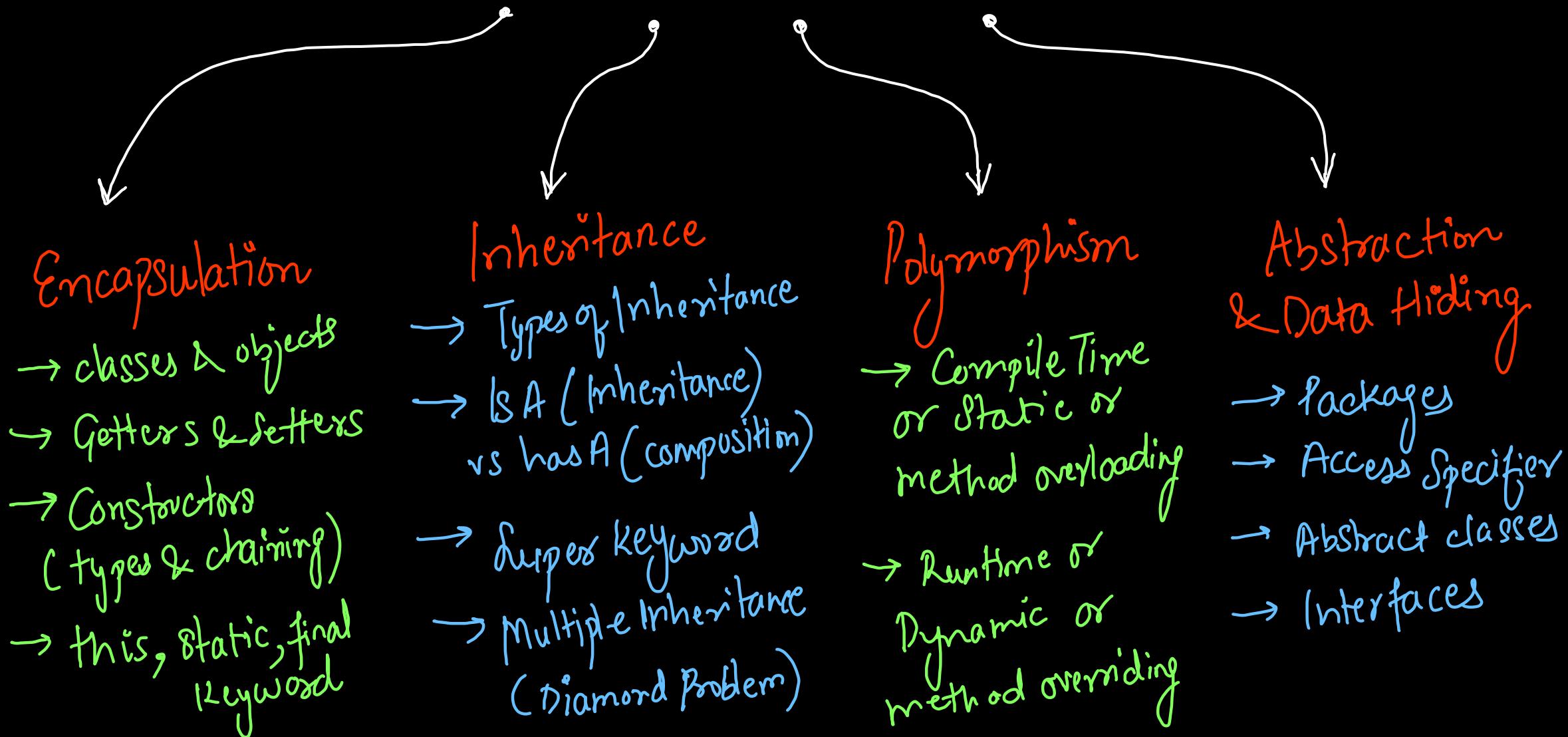
## Object oriented Programming

- Focuses on data & objects!
  - ↳ Data members & member functions
- More reliable & secure
  - ↳ Data hiding & Encapsulation
- More reusability & easy maintainability
  - ↳ Due to Inheritance & Polymorphism
- Example: → C++, Java, Python, C#

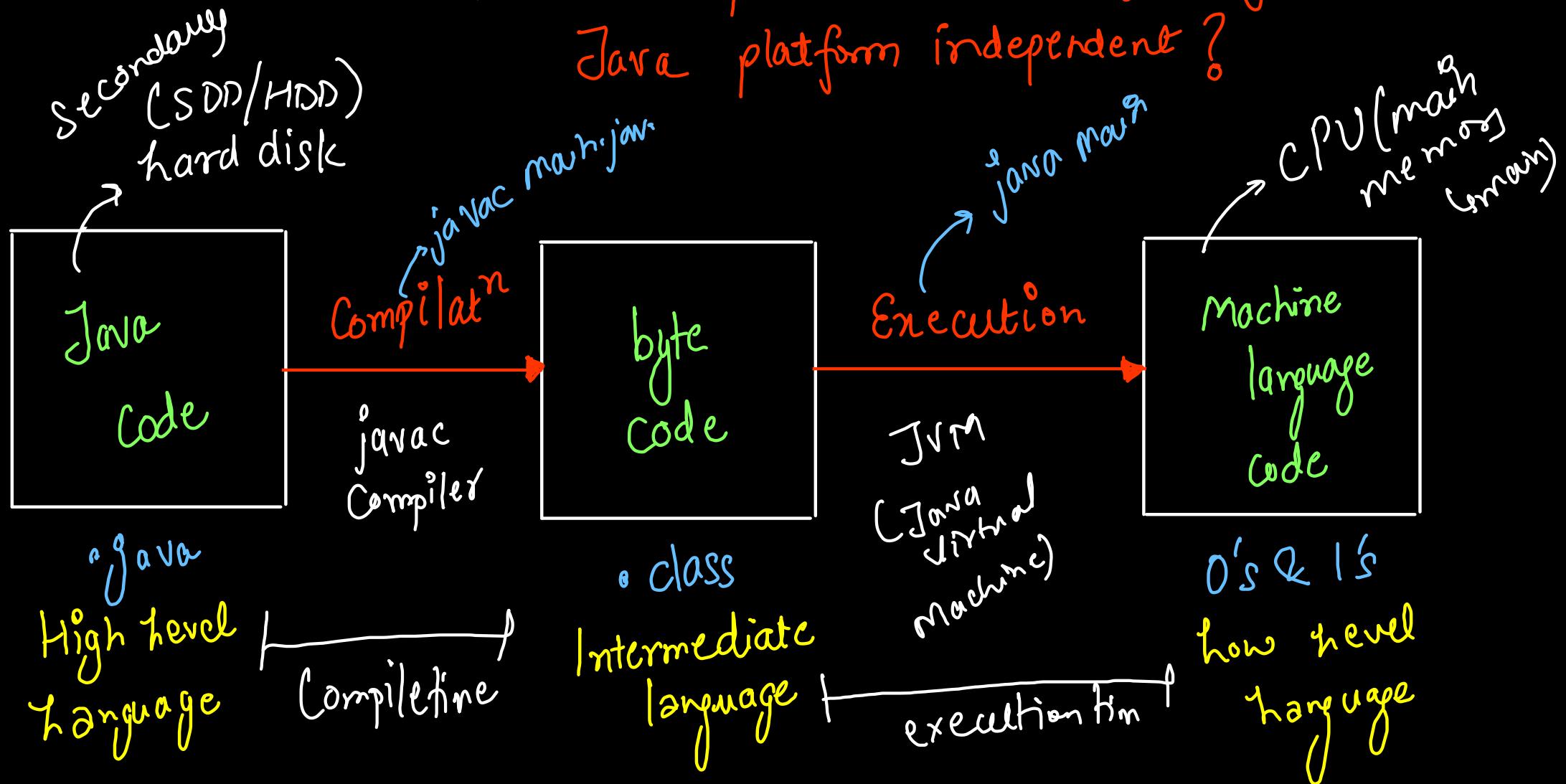
↳ higher order functions

# Need of Object Oriented Programming

## PILLARS OF OOPS



~~(Q) Explain Software Development Process in Java. What are various components in Java? Why is Java platform independent?~~



# # Java Architecture

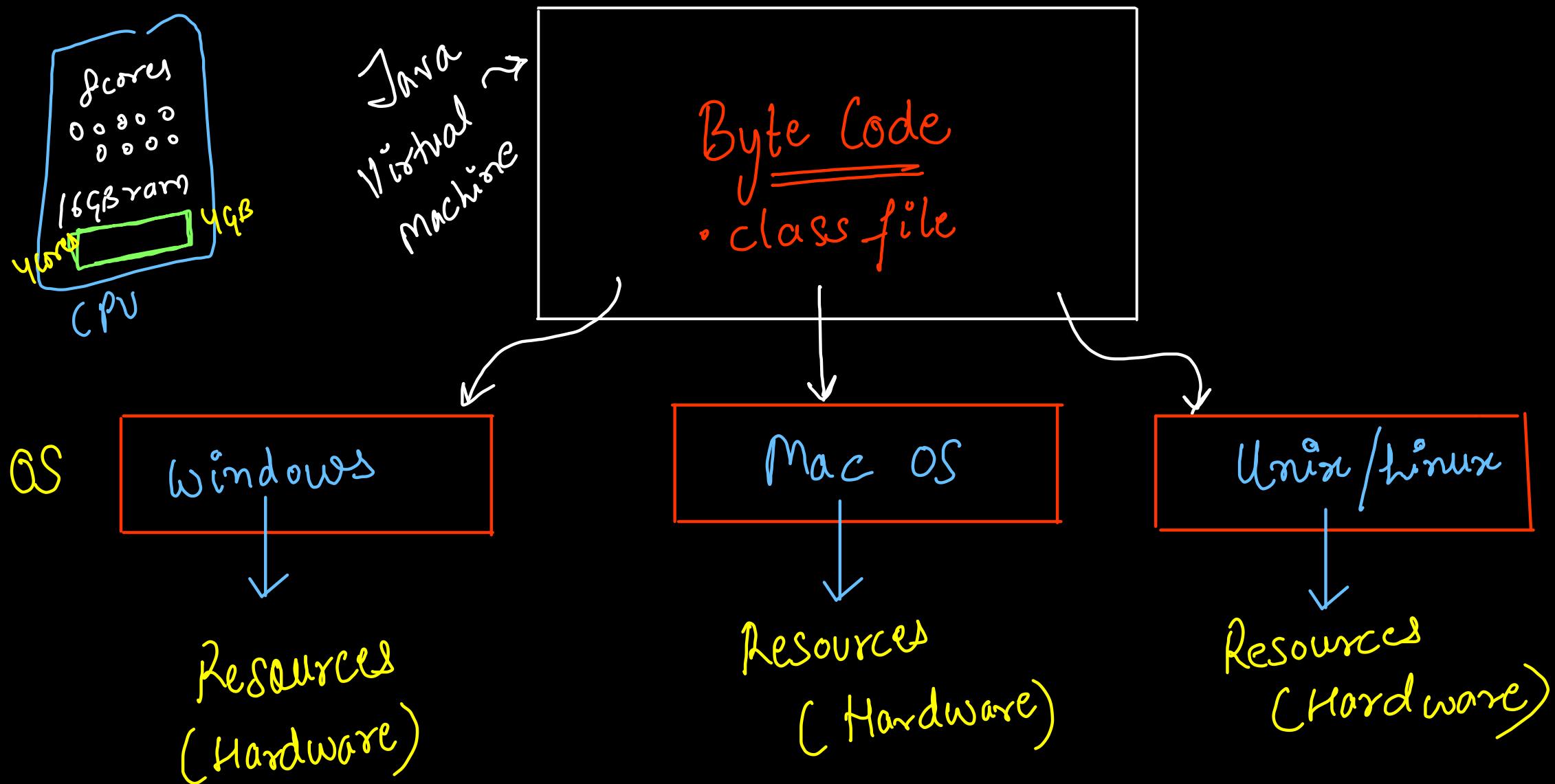
Oracle

Java Development Kit (JDK) = JRE + Development Tools  
{ Interpreter, Class Loader, Compiler (javac) }  
linker,

Java Runtime Environment (JRE)  
= JVM + library classes

Java Virtual Machine (JVM)

# # Java Is Platform Independent



\* Games  
\* desktop apps

C++ → 2 faster → Java

.cpp source code

compile

machine  
dependent

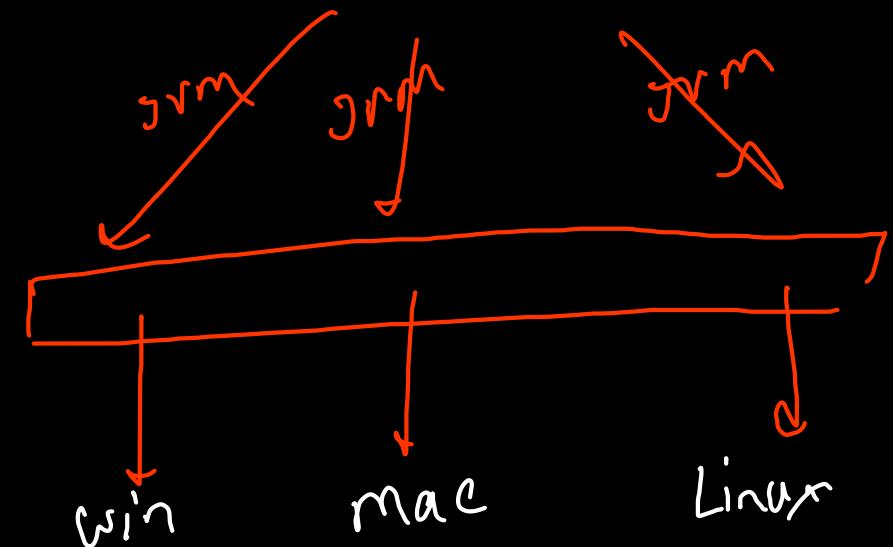
exe executable file

faster

run on operating syst

.java source code

.class byte code



Q) What are differences between Program & Process? What do you mean by process memory layout? Explain the steps in process creation/ program execution?

(secondary storage)  
Hard disk      # Program vs Process

# Program → Code / Set of Instructions (.java)

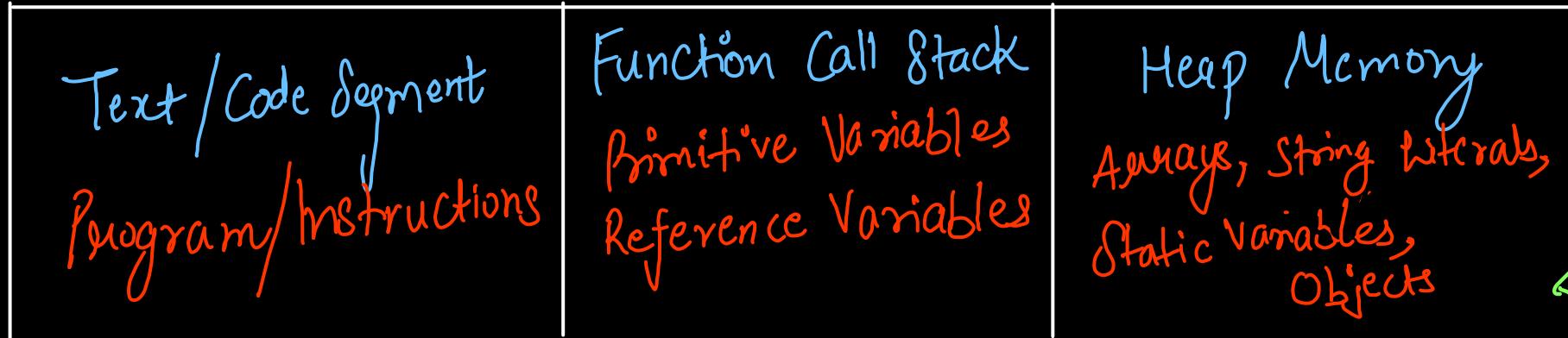
# Program under execution is known as process

# Process = Program (Code) + Stack & Heap Memory  
+ Other resources

(CPU, disk, Network, I/O file, etc)

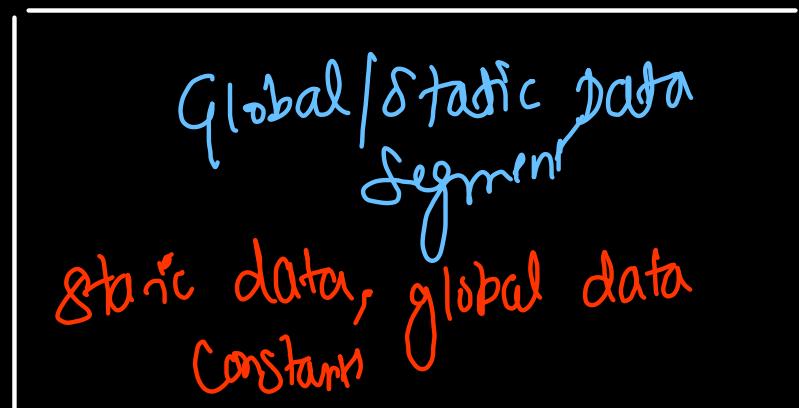
RAM

## # Process Memory layout (Random Access memory) main memory

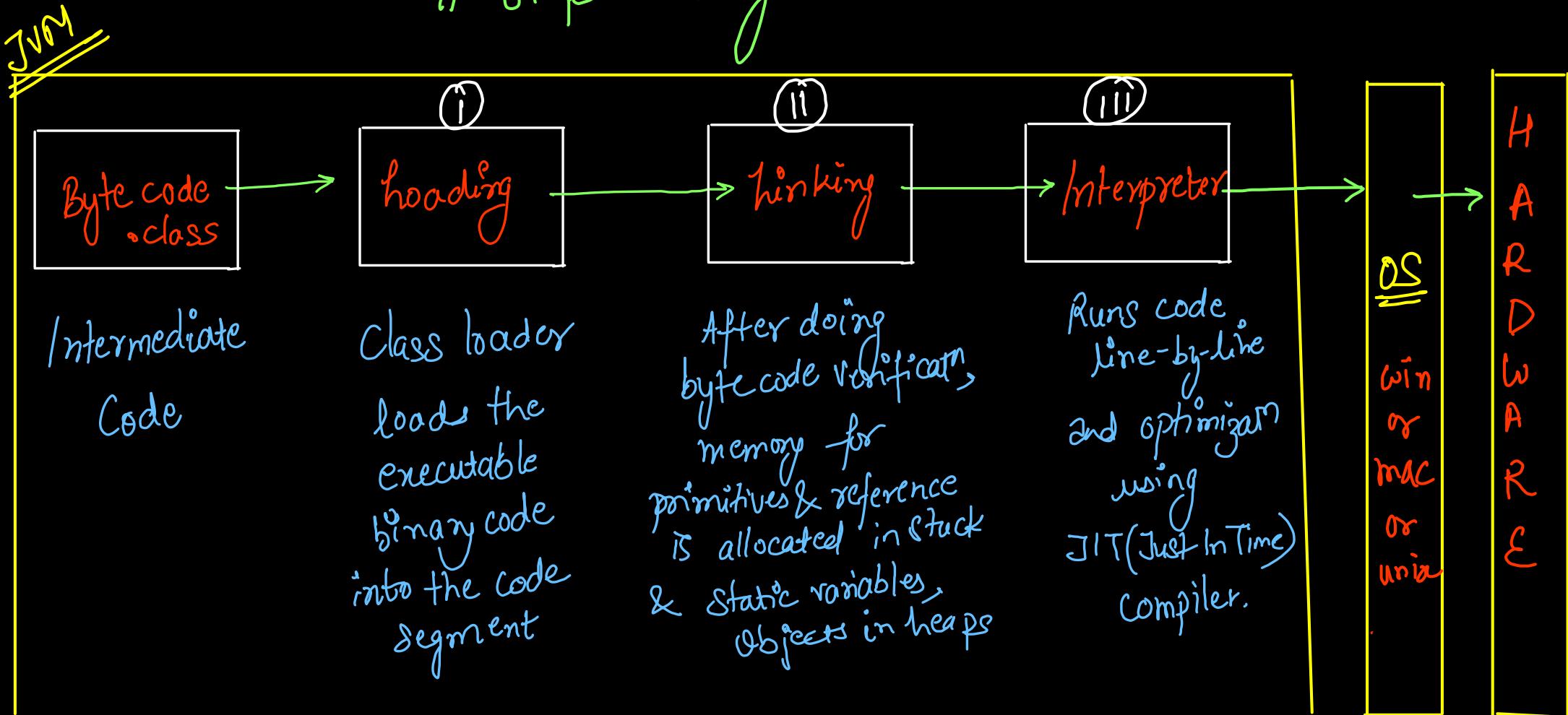


function scope (int, char, boolean, short, byte, long, float, double)  
lifetime = fn

scope = program  
(global)  
lifetime = program



# # 8 Steps in Program Execution



Q) How are classes different from objects in Java?

Explain with the help of real world example.

Class	vs	Object
• logical Entity		• Physical / Real-world entity
• do not occupy stack or heap memory space		• Occupies both stack & heap memory space
• Blueprint for an Object		• Instance of a class
Data Members (Properties)	↓	Reference Variable in Stack
Member functions (Behavior)	↓	Actual Object in Heap

```
class Movie{  
    int duration;  
    String name;  
    double ratings;  
    String genre;  
}
```

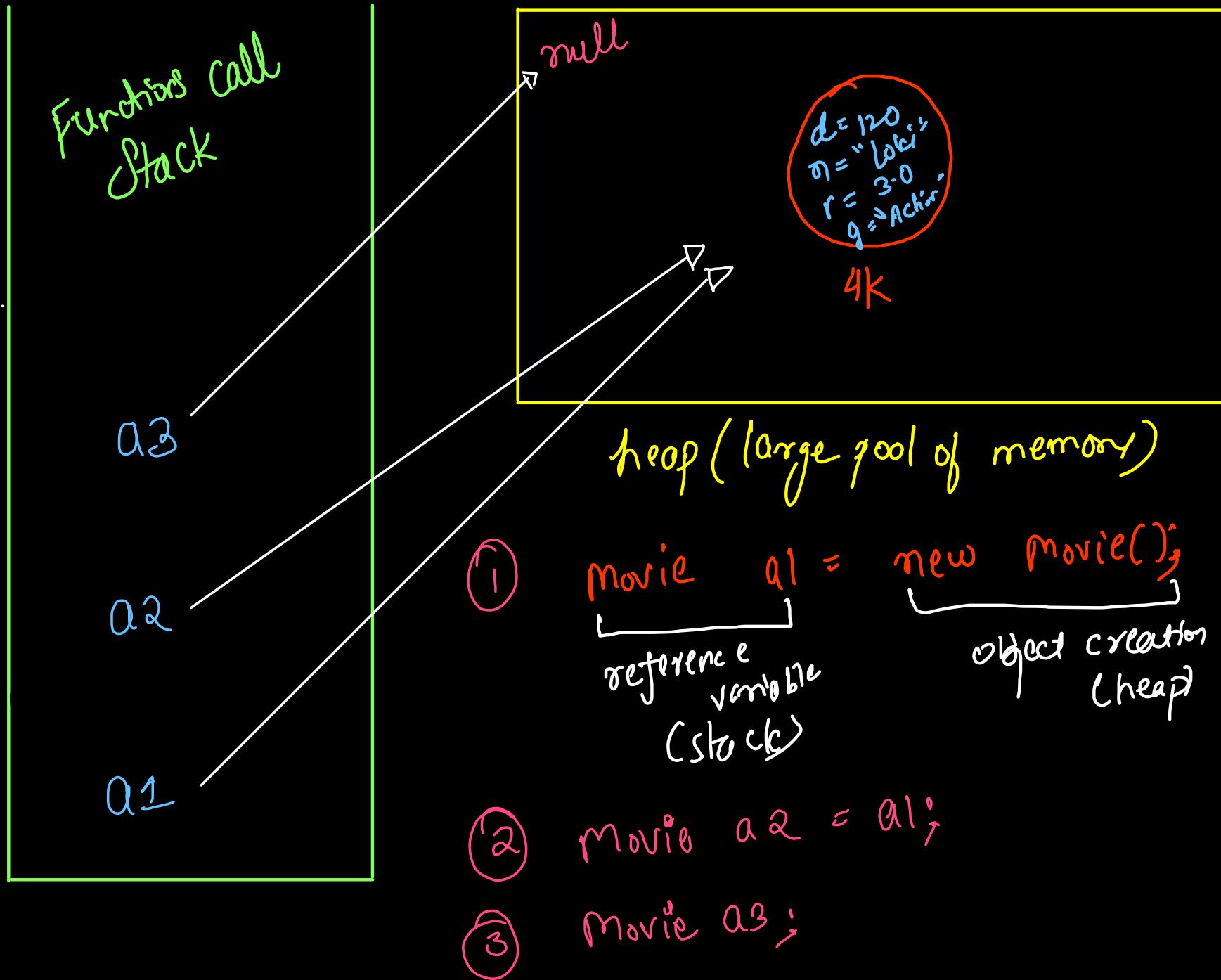
Class → Text Segment  
(Class Header)

No memory allocated

```
Movie avengers1 = new Movie();  
avengers1.duration = 120;  
avengers1.name = "Avengers Loki";  
avengers1.ratings = 4.0;  
avengers1.genre = "Action";  
  
// System.out.println(avengers1);  
System.out.println(avengers1.name + " " +  
                    avengers1.duration + " " +  
                    avengers1.genre + " " +  
                    avengers1.ratings);
```

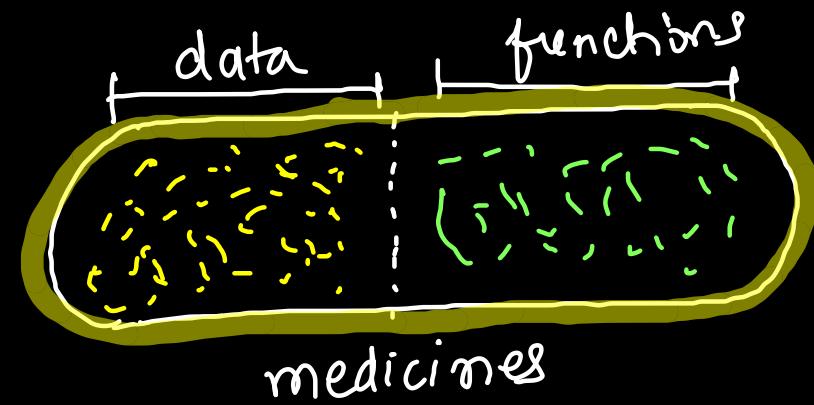
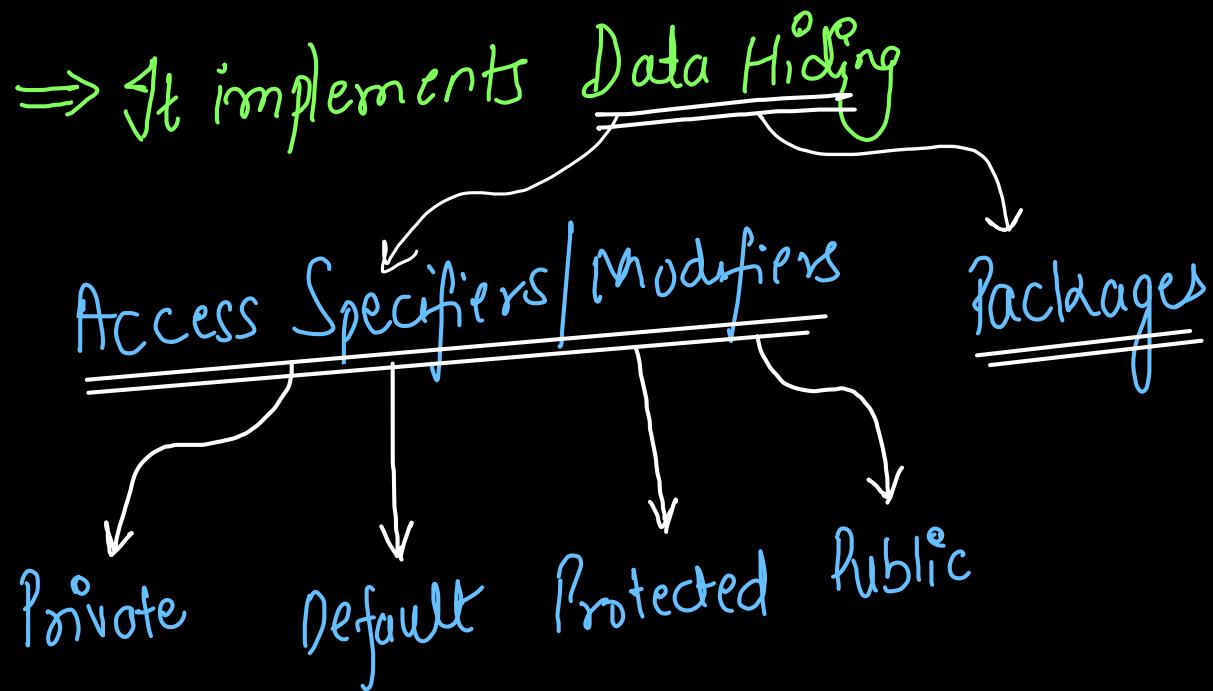
Object  
Memory Allocation  
Reference Variable → Stack  
Actual Instance → Heap

Avengers Loki 120 Action 4.0



Q) What is ENCAPSULATION? Explain it with the help of real world example!

"Wrapping together the data members and member functions into a single entity (known as class)"



```
public static void main(String[] args) {  
    int[] durations = {120, 150, 140, 200, 130};  
  
    String[] name = {"Avengers Loki",  
                    "Avengers Ultron",  
                    "Avengers Infinity War",  
                    "Avengers Endgame",  
                    "Thor Love and Thunder"};  
  
    double[] ratings = {4.0, 3.0, 4.9, 5.0, 2.0};  
  
    String[] genre = {"Action", "Comedy", "Thriller", "Scifi", "Romance"};  
  
    System.out.println(name[4] + " " +  
                       durations[4] + " " +  
                       genre[4] + " " +  
                       ratings[4]);  
}
```

## Code Without Encapsulation (Without Classes & Objects)

Finished in 136 ms

Thor Love and Thunder 130 Romance 2.0

```
class Movie{  
    private int duration;  
    private String name;  
    private double rating;  
    private String genre;
```

→ Data Members (Properties) ⇒ Private

```
public void setDuration(int newDuration){ duration = newDuration; }  
public void setName(String newName){ name = newName; }  
public void setRating(double newRating){ rating = newRating; }  
public void setGenre(String newGenre){ genre = newGenre; }
```

```
public int getDuration(){ return duration; }  
public String getName(){ return name; }  
public double getRating(){ return rating; }  
public String getGenre(){ return genre; }
```

}getters

setters

Member Functions  
(Behavior)  
(Manipulators)

```
Movie avengers1 = new Movie();  
avengers1.setDuration(120);  
avengers1.setName("Avengers Loki");  
avengers1.setRating(4.0);  
avengers1.setGenre("Action");  
  
System.out.println(avengers1.getName() + " "  
    + avengers1.getDuration() + " " +  
    avengers1.getGenre() + " " +  
    avengers1.getRating());
```

↓  
Public

Q (A) What are **CONSTRUCTORS** in Java? Explain with a coding example.

- special function → same name as the classname
- provide default values to the properties.
- called as soon as object is created
- for an object, constructor is only called once
- no "explicit" return type (But there is an implicit return type)

- It is a **special function** which gets called as soon as the object is allocated **heap memory**.
- It has **same name** as **classname**.
- It has **no explicit return type**.
- Implicitly, return type of constructor is "**this**" (reference of the class current object)
- Constructor cannot be **static/abstract/final**.  
However, it can be **private/protected/default**.

```
class Movie{  
    public Movie(){  
        duration = 100;  
        name = "Untitled";  
        rating = 0.0;  
        genre = "Unclassified";  
    }  
  
    private int duration;  
    private String name;  
    private double rating;  
    private String genre;  
  
    public void setDuration(int newDuration){ duration = newDuration; }  
    public void setName(String newName){ name = newName; }  
    public void setRating(double newRating){ rating = newRating; }  
    public void setGenre(String newGenre){ genre = newGenre; }  
  
    public int getDuration(){ return duration; }  
    public String getName(){ return name; }  
    public double getRating(){ return rating; }  
    public String getGenre(){ return genre; }  
}
```

object(this)  
Movie a1 = new Movie();  
Constructor call  
Reference

Q(B) What are the types of constructors in java?  
Write implementation of all of them.

### Types of Constructors

- Default Implicit Constructor → no parameter, no body
- Default Explicit Constructor → no parameters
- Parameterized Constructor → 1 or more than 1 parameters
- Copy constructor → parameter of reference of same class' object

## (A) Default Implicit Constructor

```
public Movie(){}
```

## (B) Default Explicit Constructor

```
public Movie(){  
    duration = 100;  
    name = "Untitled";  
    rating = 0.0;  
    genre = "Unclassified";  
}
```

## (C) Parameterized Constructor

```
public Movie(int duration, String name,  
            double rating, String genre){  
  
    setDuration(duration);  
    setName(name);  
    setRating(rating);  
    setGenre(genre);  
}
```

## (D) Copy Constructor

```
public Movie(Movie other){  
    setDuration(other.getDuration());  
    setName(other.getName());  
    setRating(other.getRating());  
    setGenre(other.getGenre());  
}
```

Q) What is constructor overloading? What are the rules for overloading of two methods/constructors?

\* → Function name should be same

- Two constructors are said to be overloaded if atleast one condition is satisfied :→

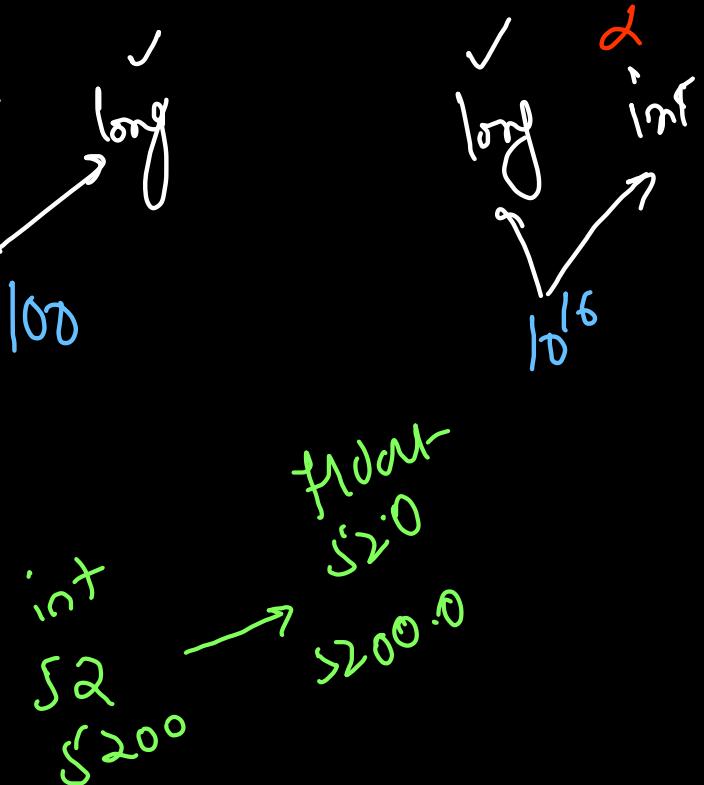
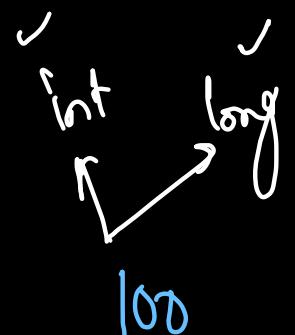
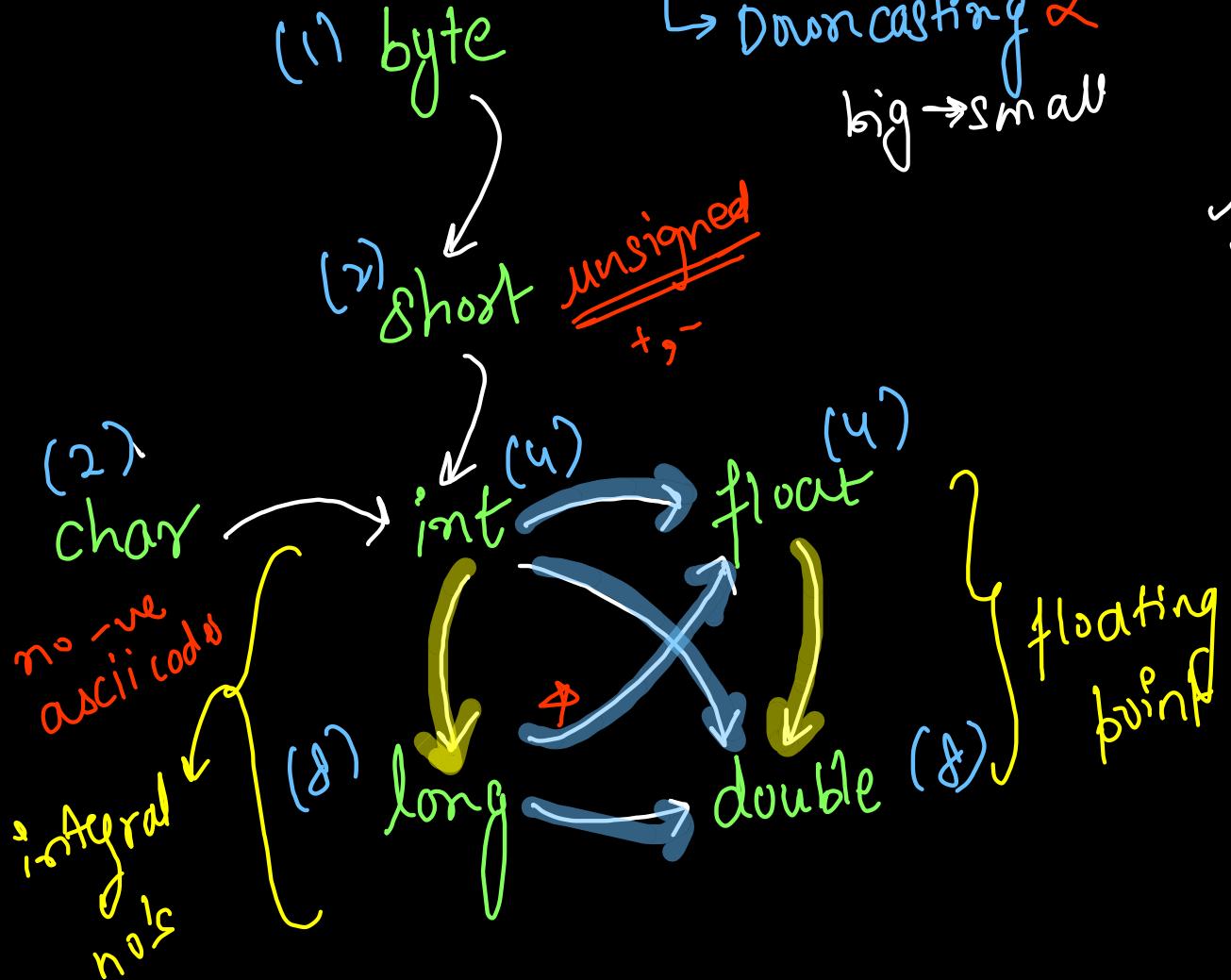
→ Number of arguments are different

→ Types of arguments are different

→ Order of arguments are different

\* → Change in return type "does not" make functions/methods overloaded.

# Type Promotion in Java  $\Rightarrow$  to resolve ambiguity and match function call to a particular function definition



To put it another way, the JLS distinguishes between a loss of ***magnitude*** and a loss of ***precision***.

`int` to `byte` for example is a (potential) loss of magnitude because you can't store 500 in a `byte`.

`long` to `float` is a potential loss of precision but not magnitude because the value range for floats is larger than that for longs.

So the rule is:

- **Loss of magnitude:** explicit cast required;
- **Loss of precision:** no cast required.

```
class Movie {  
    int duration;  
    String name, genre;  
    double rating;  
  
    public Movie(int duration) {  
        this.duration = duration;  
    }  
}
```

```
class Driver {  
    Run | Debug  
    public static void main(String[] args) {  
        // NO EXACT MATCH FOUND: Movie(char)  
        // Char Type Promoted to Integer (Upcasting - IMPLICIT)  
  
        Movie avengers1 = new Movie(duration: 'A');  
        System.out.println(avengers1.duration);  
  
        // COMPILATION ERROR: Long Demoted to Integer  
        // (Downcasting - IMPLICITLY NOT POSSIBLE)  
        // Movie avengers2 = new Movie(180l);  
        // System.out.println(avengers2.duration);  
  
        // NO EXACT MATCH FOUND: Movie(long)  
        // Long Type Demoted to Integer (Downcasting - EXPLICIT)  
  
        Movie avengers2 = new Movie((int) 180l);  
        System.out.println(avengers2.duration);  
    }  
}
```

- architaggarwal@Archits-MacBook-Air Java 00PS % javac 00PS\_Codes/5.TypePromotion.java
  - architaggarwal@Archits-MacBook-Air Java 00PS % java 00PS\_Codes.Driver
- 65  
180

Q) Give the corrected output for the following code out of the given options.

Code :-

```
public static void swap(Movie a1, Movie a2){  
    Movie a3 = a1;  
    a1 = a2;  
    a2 = a3;  
}
```

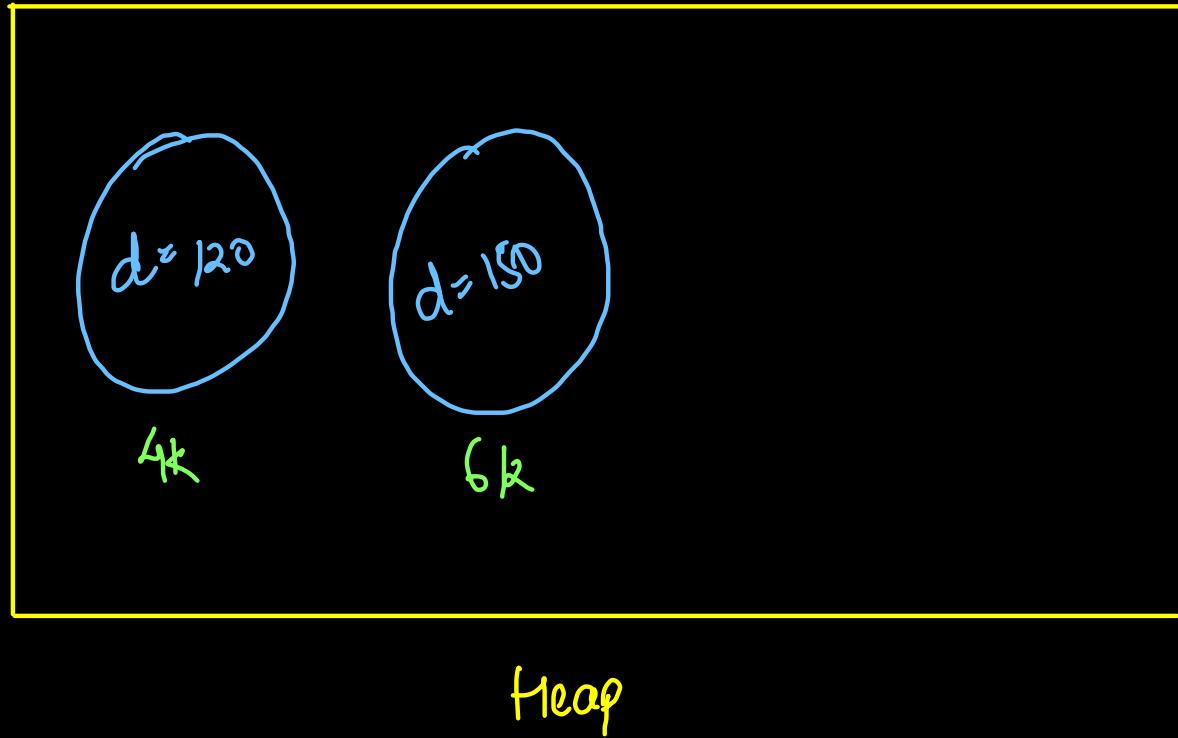
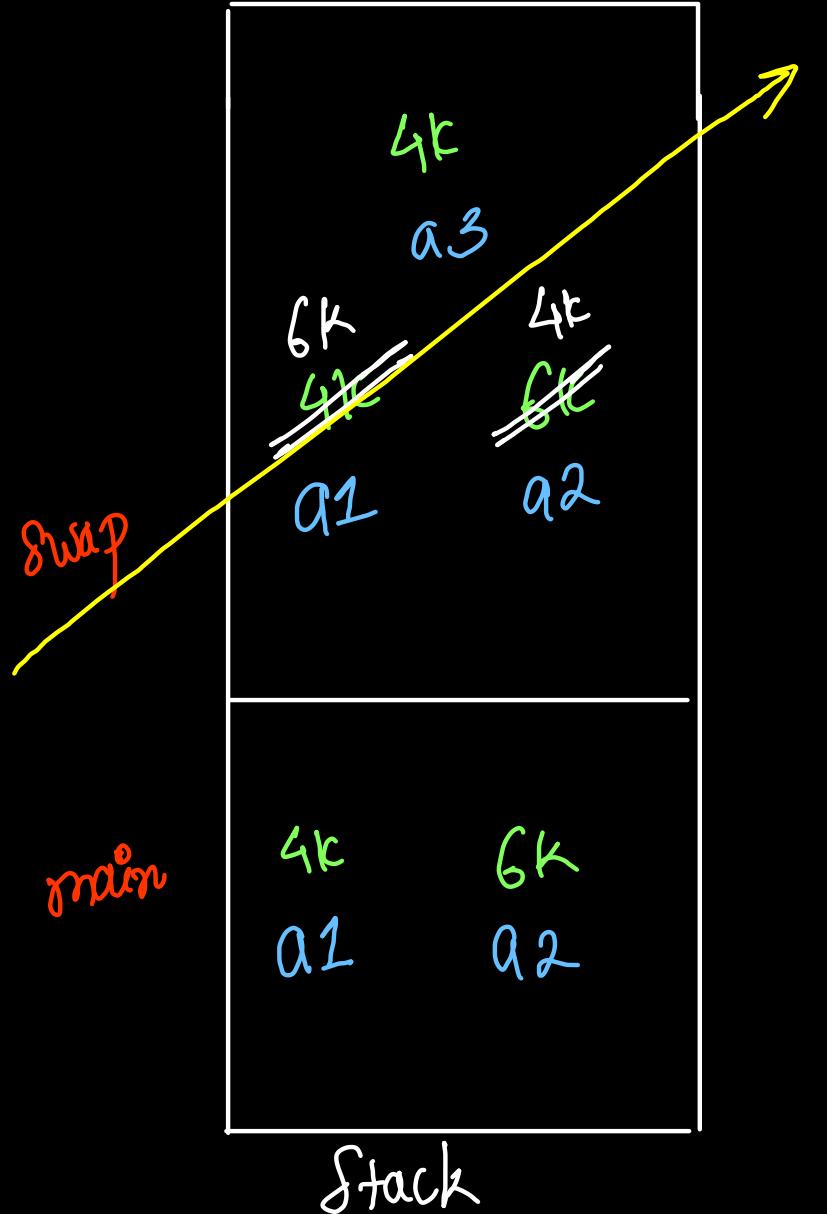
```
Movie a1 = new Movie();  
a1.setDuration(120);  
System.out.println(a1.getDuration());  
  
Movie a2 = new Movie();  
a2.setDuration(150);  
System.out.println(a2.getDuration());  
  
swap(a1, a2);  
  
System.out.println(a1.getDuration());  
System.out.println(a2.getDuration());
```

Options :-

- (a) 120, 150, 150, 120
- (b) 120, 150, 120, 150
- (c) 120, 150, 120, 120
- (d) 120, 150, 150, 150

java is always  
pass by value!  
↓  
stack changes  
does persist  
not  $\equiv$

Swap Game - ①



Q) Give the corrected output for the following code out of  
the given options.

Code :-

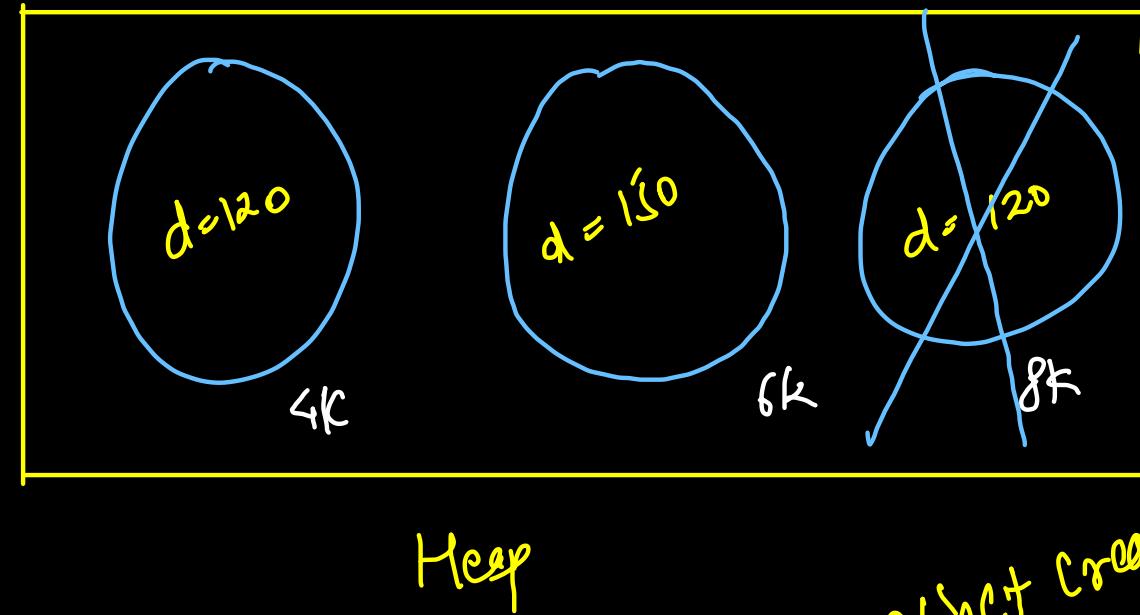
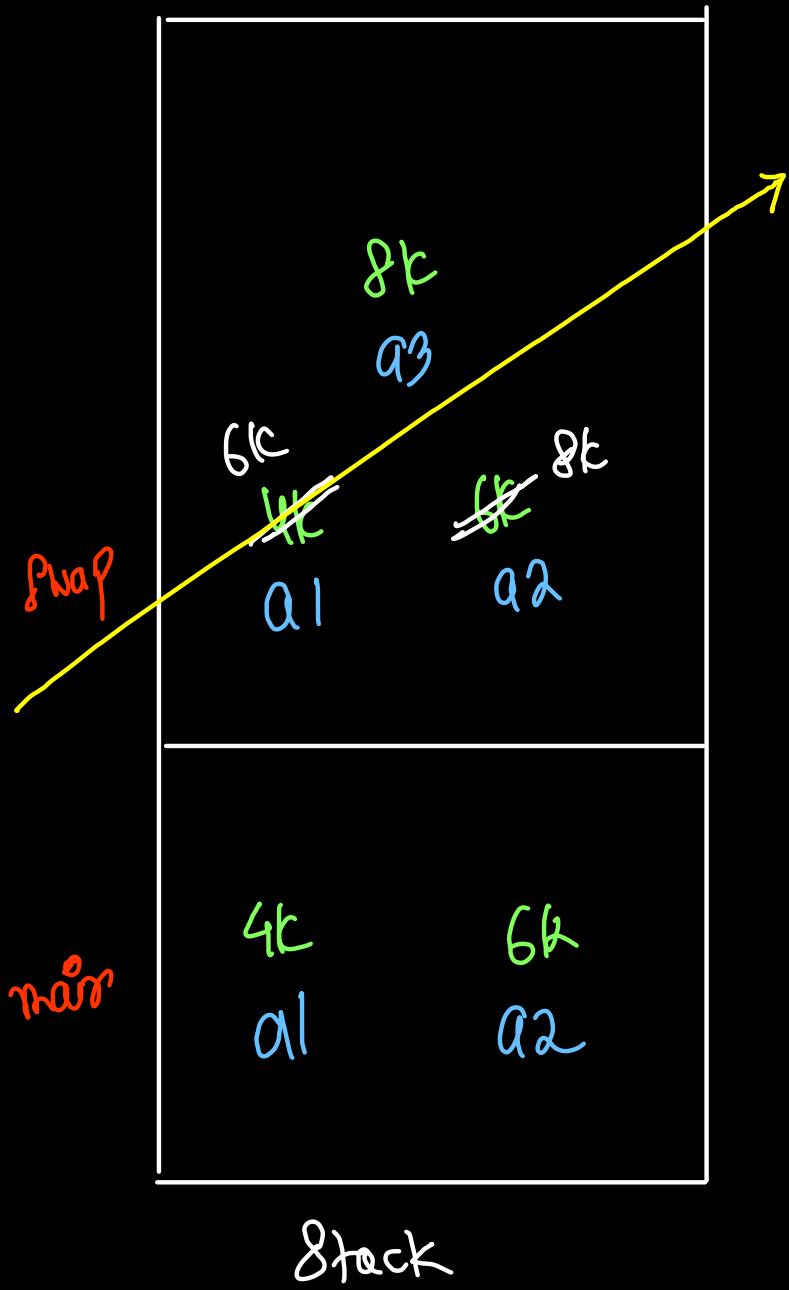
```
public static void swap(Movie a1, Movie a2){  
    Movie a3 = new Movie();  
    a3.setDuration(a1.getDuration());  
  
    a1 = a2;  
    a2 = a3;  
}
```

```
Movie a1 = new Movie();  
a1.setDuration(120);  
System.out.println(a1.getDuration());  
  
Movie a2 = new Movie();  
a2.setDuration(150);  
System.out.println(a2.getDuration());  
  
swap(a1, a2);  
  
System.out.println(a1.getDuration());  
System.out.println(a2.getDuration());
```

Options :-

- (a) 120, 150, 150, 120
- (b) 120, 150, 120, 150
- (c) 120, 150, 120, 120
- (d) 120, 150, 150, 150

Swap Game - 2



Object creation  
↳ dump  
↳ garbage collection

Q) Give the corrected output for the following code out of  
the given options.

Code →

```
public static void swap(Movie a1, Movie a2){  
    Movie a3 = a1;  
  
    a1.setDuration(a2.getDuration());  
    a2.setDuration(a3.getDuration());  
}
```

```
Movie a1 = new Movie();  
a1.setDuration(120);  
System.out.println(a1.getDuration());  
  
Movie a2 = new Movie();  
a2.setDuration(150);  
System.out.println(a2.getDuration());  
  
swap(a1, a2);  
  
System.out.println(a1.getDuration());  
System.out.println(a2.getDuration());
```

Options →

- (a) 120, 150, 150, 120
- (b) 120, 150, 120, 150
- (c) 120, 150, 120, 120
- (d) 120, 150, 150, 150

Swap Game → 3

Stack

$a_3 = 4k$

$a_1 = 4k$

$a_2 = 6k$

$a_1 = 4k$

$a_2 = 6k$

Swap

Heap

$d = 125$

4k

$d = 150$

6k

Heap changes will persist.

main

Q) Give the corrected output for the following code out of  
the given options.

Code :-

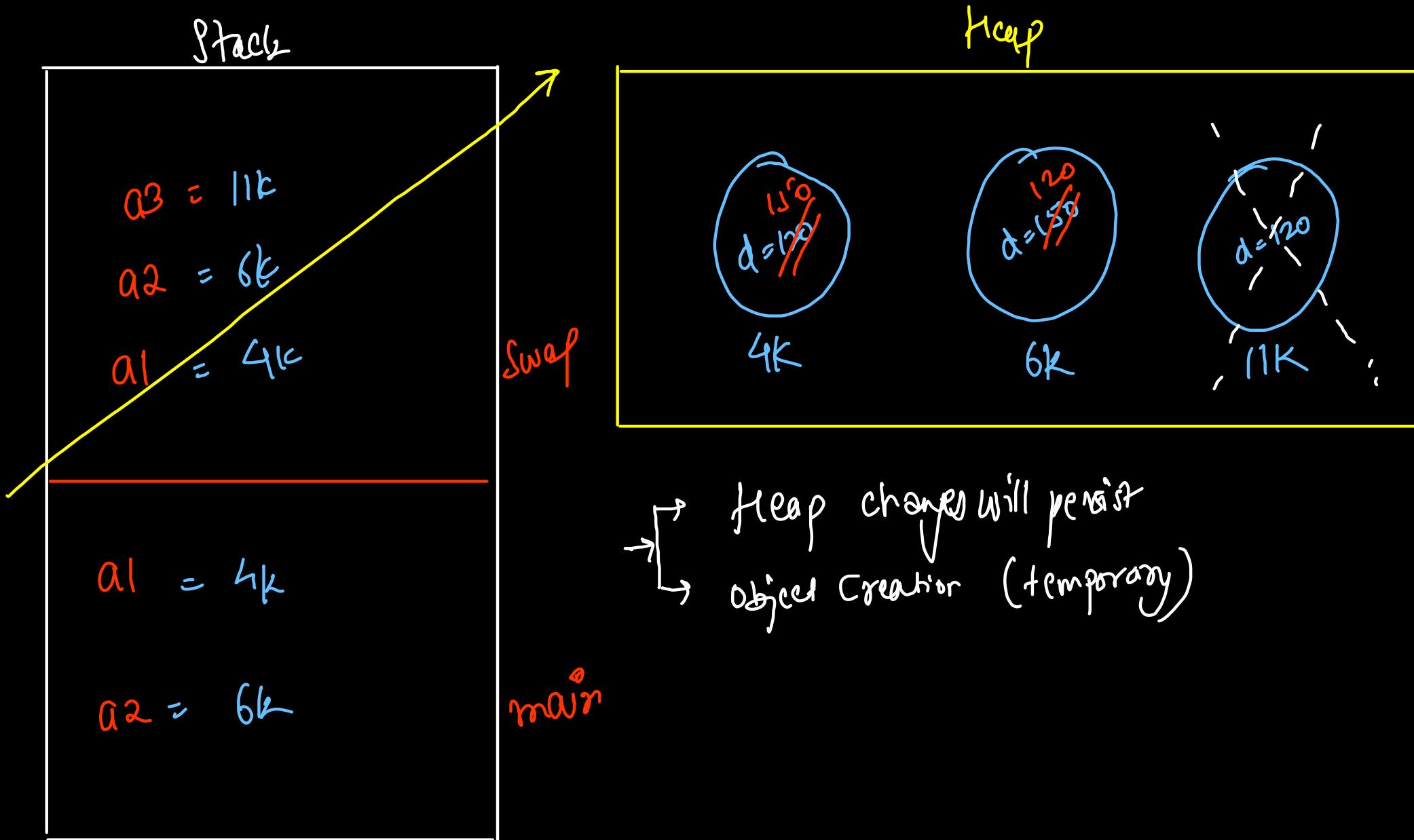
```
public static void swap(Movie a1, Movie a2){  
    Movie a3 = new Movie();  
    a3.setDuration(a1.getDuration());  
  
    a1.setDuration(a2.getDuration());  
    a2.setDuration(a3.getDuration());  
}
```

```
Movie a1 = new Movie();  
a1.setDuration(120);  
System.out.println(a1.getDuration());  
  
Movie a2 = new Movie();  
a2.setDuration(150);  
System.out.println(a2.getDuration());  
  
swap(a1, a2);  
  
System.out.println(a1.getDuration());  
System.out.println(a2.getDuration());
```

Options :-

- (a) 120, 150, 150, 120
- (b) 120, 150, 120, 150
- (c) 120, 150, 120, 120
- (d) 120, 150, 150, 150

Swap Games - 4



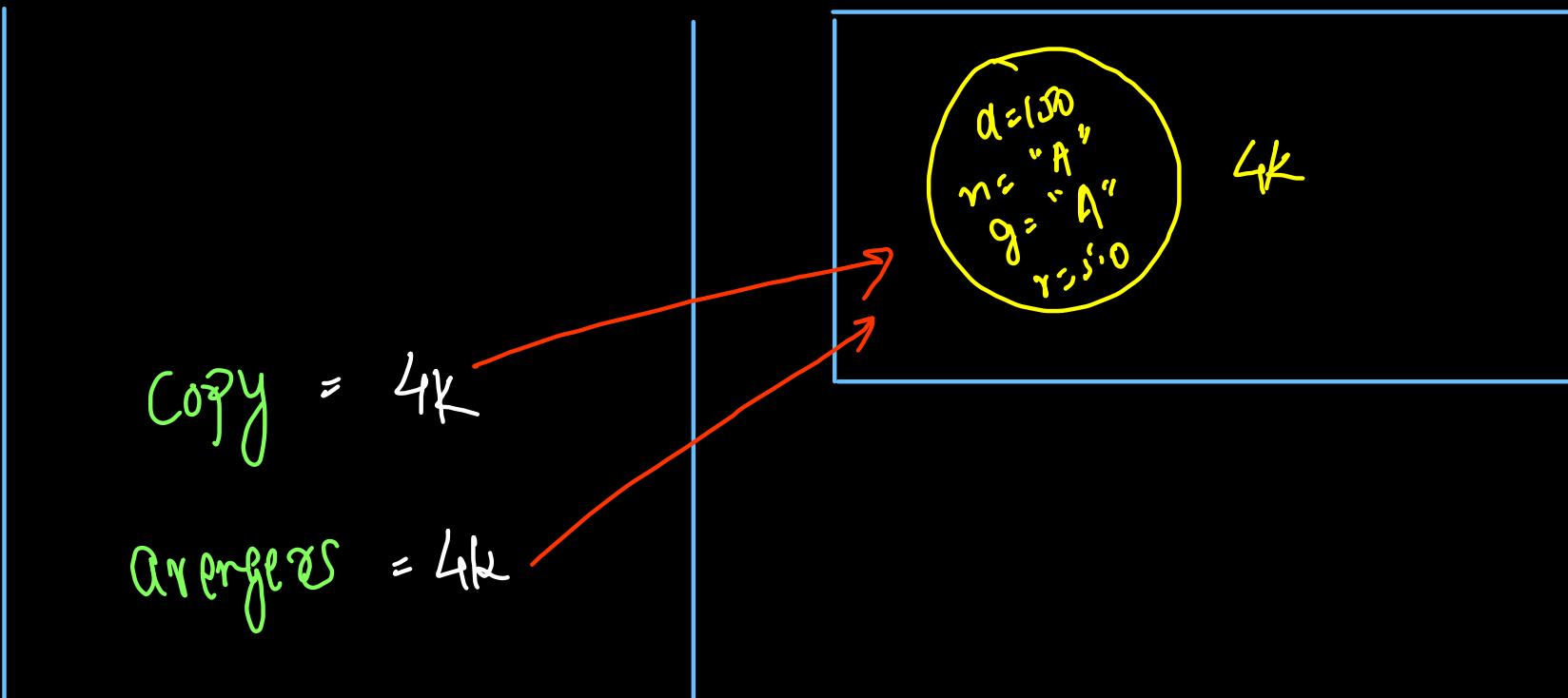
→ Heap changes will persist  
Object Creation (temporary)

Q) What is the difference between shallow copy and deep copy?

- Shallow Copy :-
- It stores the references of object to the original memory address.
  - Changes made in the cloned new object are reflected in the original old object and vice-versa.
  - It is just copy of reference variables, and not actual object creation is taking place
  - Shallow copy is faster than deep copy!

```
Movie avengers = new Movie();
System.out.println(avengers.duration + " " + avengers.genre
    + " " + avengers.name + " " + avengers.ratings);
```

```
Movie copy = avengers;
System.out.println(copy.duration + " " + copy.genre
    + " " + copy.name + " " + copy.ratings);
```



```
class Movie{
    int duration = 150;
    String genre = "Action", name = "Avengers";
    double ratings = 5.0;
}

public class Main {
    public static void main(String[] args) {
        Movie avengers = new Movie();
        System.out.println(avengers.duration + " " + avengers.genre
                           + " " + avengers.name + " " + avengers.ratings);

        Movie copy = avengers;
        System.out.println(copy.duration + " " + copy.genre
                           + " " + copy.name + " " + copy.ratings);

        copy.duration = 180;
        System.out.println(avengers.duration + " " + avengers.genre
                           + " " + avengers.name + " " + avengers.ratings);
        System.out.println(copy.duration + " " + copy.genre
                           + " " + copy.name + " " + copy.ratings);
    }
}
```

- Deep Copy: →
- It creates a cloned object that is new data members (properties) & copies values in them.
  - Changes in cloned new object are not reflected in changes of original object and vice-versa.
  - It is actual copy of properties (stored in heap) and not just reference variables (in stack)
  - Deep copy is slower than shallow copy.

```
Movie(Movie other){  
    duration = other.duration;  
    genre = other.genre;  
    name = other.name;  
    ratings = other.ratings;  
}  
  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Movie avengers = new Movie();  
        System.out.println(avengers.duration + " " + avengers.genre  
                           + " " + avengers.name + " " + avengers.ratings);  
  
        Movie deepCopy = new Movie(avengers);  
        System.out.println(deepCopy.duration + " " + deepCopy.genre  
                           + " " + deepCopy.name + " " + deepCopy.ratings);  
  
        avengers.duration = 180;  
        System.out.println(avengers.duration + " " + avengers.genre  
                           + " " + avengers.name + " " + avengers.ratings);  
        System.out.println(deepCopy.duration + " " + deepCopy.genre  
                           + " " + deepCopy.name + " " + deepCopy.ratings);  
    }  
}
```

deepCopy = 6k  
avengers = 4k

~~d=150~~ 180  
d=150 4k

```
class Movie {  
    int duration;  
    String name, genre;  
    double rating;  
    ArrayList<String> languages;  
  
    public Movie(String name, int duration, double rating, String genre) {  
        this.name = name;  
        this.duration = duration;  
        this.genre = genre;  
        this.rating = rating;  
        this.languages = new ArrayList<>();  
    }  
  
    // PARTIAL DEEP COPY CONSTRUCTOR  
    public Movie(Movie other) {  
        this.name = other.name;  
        this.duration = other.duration;  
        this.genre = other.genre;  
        this.rating = other.rating;  
        this.languages = other.languages;  
    }  
}
```

```
● architaggarwal@Archits-MacBook-Air Java 0OPS % javac OOPS_Codes/6.1.ShallowCopy.java  
● architaggarwal@Archits-MacBook-Air Java 0OPS % java OOPS_Codes.Driver  
Avengers Endgame 180 4.5 SuperHero  
[English]  
[English, Tamil, Malayalam]  
[English, Tamil, Malayalam]  
Avengers Endgame 180 4.5 SuperHero  
[English, Tamil, Malayalam]  
[English, Tamil, Malayalam, Hindi, Telugu]  
[English, Tamil, Malayalam, Hindi, Telugu]  
○ architaggarwal@Archits-MacBook-Air Java 0OPS %
```

```
class Driver {  
    Run | Debug  
    public static void main(String[] args) {  
  
        Movie avengers = new Movie(name: "Avengers Endgame", duration: 180,  
        rating: 4.5, genre: "SuperHero");  
        avengers.languages.add(e: "English");  
  
        System.out.println(avengers.name + " " + avengers.duration  
            + " " + avengers.rating + " " + avengers.genre);  
        System.out.println(avengers.languages);  
  
        // SHALLOW COPY  
  
        Movie shallowCopy = avengers;  
  
        avengers.languages.add(e: "Tamil");  
        shallowCopy.languages.add(e: "Malayanam");  
  
        System.out.println(shallowCopy.languages);  
        System.out.println(avengers.languages);  
  
        // PARTIAL DEEP COPY  
  
        Movie partialDeep = new Movie(avengers);  
        System.out.println(partialDeep.name + " " + partialDeep.duration  
            + " " + partialDeep.rating + " " + partialDeep.genre);  
        System.out.println(partialDeep.languages);  
  
        avengers.languages.add(e: "Hindi");  
        partialDeep.languages.add(e: "Telugu");  
  
        System.out.println(partialDeep.languages);  
        System.out.println(avengers.languages);  
    }
```

## # Plain Old Java Object (POJO) class Features

### Data Members & Member Functions in class

- Properties should be private!
- Getters & setters should be public!
- Function  Method ~ Functions inside classes associated with an object are known as methods!
  - Always create a default explicit constructor
  - to implement "Data Hiding"

Q) What are differences between methods & functions in Java?

A method is a function or procedure in object oriented programming.

- Function inside a class which is called /invoked or associated with objects of that class is known as a method.
- Functions can be written inside as well as outside classes  
ie. functions do not need a class to be created.
- Functions outside the class can only access local / primitive data like parameters whereas methods can access private data members of a class / object directly!

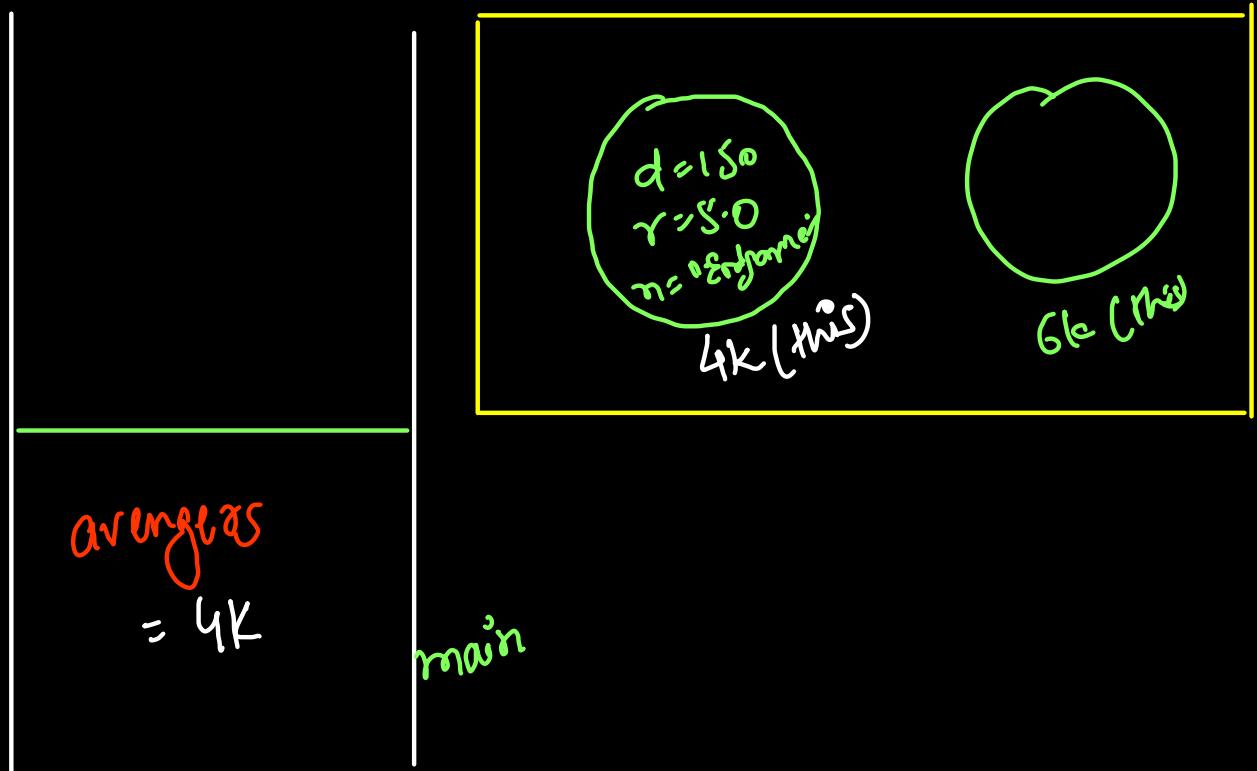
a) What is this keyword? What are it's applications?

THIS Keyword → self referential pointer (reference variable pointing to current object)

APPLICATIONS ⇒

- 1) → Refer data members from within the class
- 2) → Refer member functions from within the class
- 3) → Used in Constructor chaining (invoke own constructor)
  - chaining should be the first & single call only!
- 4) → Pass the current object as a parameter to a method/constructor
- 5) → Return the current object from a method/constructor
  - Return type of a constructor

```
class Movie {  
    int duration;  
    double ratings;  
    String name;  
    String genre;  
  
    public Movie() {  
    }  
    // IMPLICIT RETURN TYPE: THIS  
    // (MOVIE REFERENCE TYPE)  
    public Movie(int newDuration, double newRatings,  
                String newName, String newGenre) {  
        duration = newDuration;  
        ratings = newRatings;  
        name = newName;  
        genre = newGenre;  
    }  
}
```



```
public static void main(String[] args) {  
    Movie avengers = new Movie(newDuration: 150, newRatings: 5.0, newName: "Avengers Endgame",  
    newGenre: "Thriller");  
    System.out.println(avengers.duration + " " + avengers.name  
                      + " " + avengers.ratings + " " + avengers.genre);  
}
```

```
class Movie {
    private int duration;

    // Application 1: Constructor Return type
    // Application 2: Invoking Member Function
    public Movie(int duration) {
        this.setDuration(duration);
    }

    public int getDuration() {
        return duration;
    }

    // Application 3: Access Data Member Properties
    public void setDuration(int duration) {
        this.duration = duration;
    }

    // Application 4: Pass Current Object as Parameter
    public void display() {
        Driver.displayDurationOutside(this);
    }

    // Application 5: Return Current Object
    public Movie join(Movie other) {
        this.duration += other.duration;
        return this;
    }
}
```

```
class Driver {
    public static void displayDurationOutside(Movie obj) {
        System.out.println("Movie Duration = " + obj.getDuration());
    }

    Run | Debug
    public static void main(String[] args) {
        Movie avengers1 = new Movie(duration: 120);
        avengers1.display();

        Movie avengers2 = new Movie(duration: 150);
        avengers1.join(avengers2);
        avengers1.display();
    }
}
```

- architaggarwal@Archits-MacBook-Air System Design % javac OOPS\_Codes/08.ThisKeyword.java
- architaggarwal@Archits-MacBook-Air System Design % java OOPS\_Codes.Driver

```
Movie Duration = 120
Movie Duration = 270
○ architaggarwal@Archits-MacBook-Air System Design %
```

Q) Are there default parameterized constructors or functions in Java?  
What do you understand by Constructor Chaining?

```
class Cuboid{  
    int length;  
    int breadth;  
    int height;  
  
    Cuboid(){  
        // this.length = 1;  
        // this.breadth = 1;  
        // this.height = 1;  
        this(1);  
    }  
  
    Cuboid(int side){  
        // this.length = side;  
        // this.breadth = side;  
        // this.height = side;  
        this(side, side, side);  
    }  
  
    Cuboid(int length, int breadth){  
        // this.length = length;  
        // this.breadth = breadth;  
        // this.height = 1;  
        this(length, breadth, 1);  
    }  
  
    Cuboid(int length, int breadth, int height){  
        this.length = length;  
        this.breadth = breadth;  
        this.height = height;  
    }  
}
```

## Constructor Chaining

One constructor calling another constructor  
using `this()` keyword!

```
public static void main(String[] args){  
    Cuboid obj1 = new Cuboid();  
    Cuboid obj2 = new Cuboid(5);  
    Cuboid obj3 = new Cuboid(5, 10);  
    Cuboid obj4 = new Cuboid(5, 10, 15);  
  
    System.out.println(obj1.length + " " + obj1.breadth + " " + obj1.height);  
    System.out.println(obj2.length + " " + obj2.breadth + " " + obj2.height);  
    System.out.println(obj3.length + " " + obj3.breadth + " " + obj3.height);  
    System.out.println(obj4.length + " " + obj4.breadth + " " + obj4.height);  
}
```

```
Finished in 109 ms  
1 1 1  
5 5 5  
5 10 1  
5 10 15
```

⇒ Constructor chaining call  
should be the first statement  
in the calling constructor.

```
class Cuboid {  
    int length, breadth, height;  
  
    Cuboid() {  
        this.length = 1;  
        this.breadth = 1;  
        this.height = 1;  
    }  
  
    Cuboid(int side) {  
        this.length = side;  
        this.breadth = side;  
        this.height = side;  
    }  
  
    Cuboid(int length, int breadth) {  
        this.length = length;  
        this.breadth = breadth;  
        this.height = 1;  
    }  
  
    Cuboid(int length, int breadth, int height) {  
        this.length = length;  
        this.breadth = breadth;  
        this.height = height;  
    }  
}
```

```
public class Main {  
    Run | Debug  
    public static void main(String[] args) {  
        Cuboid obj1 = new Cuboid();  
        System.out.println(obj1.length + " " + obj1.breadth + " " + obj1.height);  
  
        Cuboid obj2 = new Cuboid(side: 5);  
        System.out.println(obj2.length + " " + obj2.breadth + " " + obj2.height);  
  
        Cuboid obj3 = new Cuboid(length: 10, breadth: 15);  
        System.out.println(obj3.length + " " + obj3.breadth + " " + obj3.height);  
  
        Cuboid obj4 = new Cuboid(length: 10, breadth: 15, height: 20);  
        System.out.println(obj4.length + " " + obj4.breadth + " " + obj4.height);  
    }  
}
```

Code without constructor chaining  
Code redundancy!

```
class Cuboid {  
    int length, breadth, height;  
  
    Cuboid() {  
        this(side: 1);  
    }  
  
    Cuboid(int side) {  
        this(side, side, side);  
    }  
  
    Cuboid(int length, int breadth) {  
        this(length, breadth, height: 1);  
    }  
  
    Cuboid(int length, int breadth, int height) {  
        this.length = length;  
        this.breadth = breadth;  
        this.height = height;  
    }  
}
```

```
class Driver {  
    Run | Debug  
    public static void main(String[] args) {  
        Cuboid obj1 = new Cuboid();  
        Cuboid obj2 = new Cuboid(side: 5);  
        Cuboid obj3 = new Cuboid(length: 5, breadth: 10);  
        Cuboid obj4 = new Cuboid(length: 5, height: 10, breadth: 15);  
  
        System.out.println(obj1.length + " " + obj1.breadth + " " + obj1.height);  
        System.out.println(obj2.length + " " + obj2.breadth + " " + obj2.height);  
        System.out.println(obj3.length + " " + obj3.breadth + " " + obj3.height);  
        System.out.println(obj4.length + " " + obj4.breadth + " " + obj4.height);  
    }  
}
```

- architaggarwal@Archits-MacBook-Air 00PS\_Codes % javac 09.ConstructorChaining.java
- architaggarwal@Archits-MacBook-Air 00PS\_Codes % java Driver

```
1 1 1  
5 5 5  
5 1 10  
5 15 10
```

Q) Is Java a PURE object oriented programming language? What are wrapper classes? What is auto boxing and unboxing?

int, boolean, char,  
long, float, double,  
short, byte

Every entity should be class or object.  
and it should follow all  
object oriented programming  
principles.

predefined (primitive)  
or userdefined

Answer : No

Solution  $\Rightarrow$  Wrapper classes

Auto boxing & Unboxing

int $\rightarrow$ Integer	float $\rightarrow$ Float
boolean $\rightarrow$ Boolean	double $\rightarrow$ Double
char $\rightarrow$ Character	short $\rightarrow$ Short
long $\rightarrow$ Long	byte $\rightarrow$ Byte

```
public static void main(String[] args) {  
    // Primitive Stack Variable  
    int a = 5;  
    System.out.print(a + " ");  
  
    // Integer Wrapper Class  
    Integer aa = 10; // Autoboxing  
    System.out.print(aa + " ");  
  
    a = aa; // Unboxing  
    System.out.print(a + " ");  
  
    // Actual Object Creation, Getter and Setter  
    Integer b = new Integer(value: 20);  
    System.out.print(b.toString() + " ");  
  
    a = b.intValue();  
    System.out.print(a + " ");  
}
```

```
● architaggarwal@Archits-MacBook-Air 0OPS_Codes % javac 10.Wrap  
perClasses.java  
10.WrapperClasses.java:17: warning: [removal] Integer(int) in  
Integer has been deprecated and marked for removal  
    Integer b = new Integer(20);  
               ^  
1 warning  
● architaggarwal@Archits-MacBook-Air 0OPS_Codes % java Driver  
5 10 10 20 20 %
```

```
public static void functionsAndConst() {
    Integer a = 10;

    // Integer to String and String to Integer
    String b = a.toString();
    System.out.println(a + " Integer to String : " + b);

    String c = "256";
    Integer d = Integer.parseInt(c);
    System.out.println(c + " in Integer : " + d);

    // Various Number Conversions
    System.out.println(a + " Decimal to Binary : " + Integer.toBinaryString(a));
    System.out.println(a + " Decimal to Hexadecimal : " + Integer.toHexString(a));
    System.out.println(a + " Decimal to Octal : " + Integer.toOctalString(a));

    System.out.println("Integer MAXIMUM Range : " + Integer.MAX_VALUE);
    System.out.println("Integer MINIMUM Range : " + Integer.MIN_VALUE);

    // Various Other Functions
    System.out.println(a.compareTo(d));
    System.out.println(a.equals(d));
    System.out.println(Integer.max(a, d));
    System.out.println(Integer.min(a, d));
}
```

- ```
● architaggarwal@Archits-MacBook-Air 0OPS_Codes % java Driver
10 Integer to String : 10
256 in Integer : 256
10 Decimal to Binary : 1010
10 Decimal to Hexadecimal : a
10 Decimal to Octal : 12
Integer MAXIMUM Range : 2147483647
Integer MINIMUM Range : -2147483648
-1
false
256
10
```

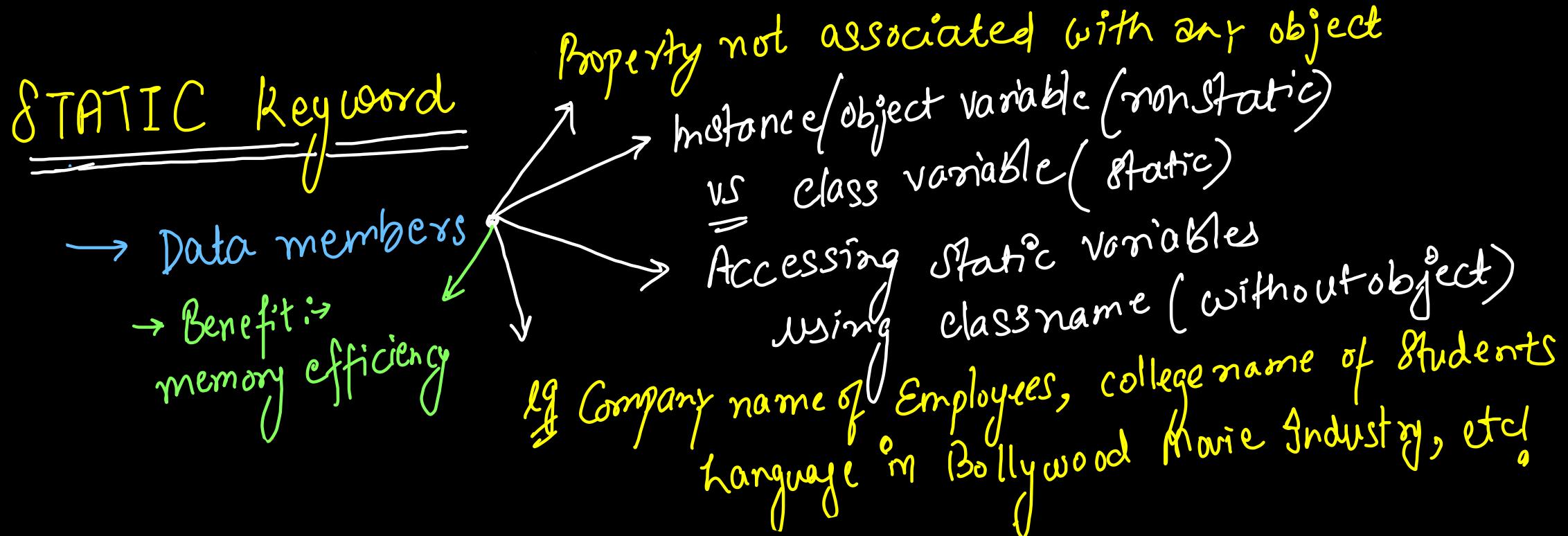
```
class MyInteger {  
    private int data;  
  
    public MyInteger(int data) {  
        this.data = data;  
    }  
  
    public int getData() {  
        return data;  
    }  
  
    public void setData(int data) {  
        this.data = data;  
    }  
}
```

```
MyInteger val1 = new MyInteger(data: 50);  
System.out.println(val1);  
System.out.println(val1.getData());  
val1.setData(data: 100);  
System.out.println(val1.getData());
```

```
-----  
MyInteger@ea30797  
50  
100
```

```
Integer val = new Integer(value: 5);  
Integer val2 = 5; // autoboxing: Integer -> int  
System.out.println(val2);  
  
int val3 = val2; // Unboxing: int -> Integer  
System.out.println(val3);
```

Q) What do you mean by static keyword? What are static variables or class variables in Java?



```
class Movie {  
    // Non Static or Instance Variables  
    int duration;  
    String name;  
    double rating;  
  
    // Static or Class Variable  
    static String language = "English";  
  
    public Movie(String name, int duration, double rating) {  
        this.name = name;  
        this.duration = duration;  
        this.rating = rating;  
    }  
}
```

```
public static void main(String[] args) {  
    Movie a1 = new Movie(name: "Avengers Infinity War", duration: 130, rating: 4.8);  
    Movie a2 = new Movie(name: "Avengers End Game", duration: 200, rating: 4.9);  
    Movie a3 = new Movie(name: "Avengers Secret Wars", duration: 170, rating: 4.6);  
    Movie a4 = new Movie(name: "Avengers Kang Dynasty", duration: 220, rating: 5.0);  
  
    // Non Static Variables  
    System.out.println(a1.name + " " + a1.duration + " " + a1.rating);  
    System.out.println(a2.name + " " + a2.duration + " " + a2.rating);  
    System.out.println(a3.name + " " + a3.duration + " " + a3.rating);  
    System.out.println(a4.name + " " + a4.duration + " " + a4.rating);  
  
    // Static Variable or Class Variable  
    System.out.print(a1.language + " ");  
    System.out.print(a2.language + " ");  
    System.out.print(a3.language + " ");  
    System.out.print(a4.language + " ");  
  
    // Accessing Using Class Name  
    // (Static Properties Even Exist Without Single Object)  
    System.out.print(Movie.language + " ");  
}
```

- architaggarwal@Archits-MacBook-Air 0OPS\_Codes % javac 11.StaticKeyword.java
- architaggarwal@Archits-MacBook-Air 0OPS\_Codes % java Driver  
Avengers Infinity War 130 4.8  
Avengers End Game 200 4.9  
Avengers Secret Wars 170 4.6  
Avengers Kang Dynasty 220 5.0  
English English English English %

Interview  
Ques

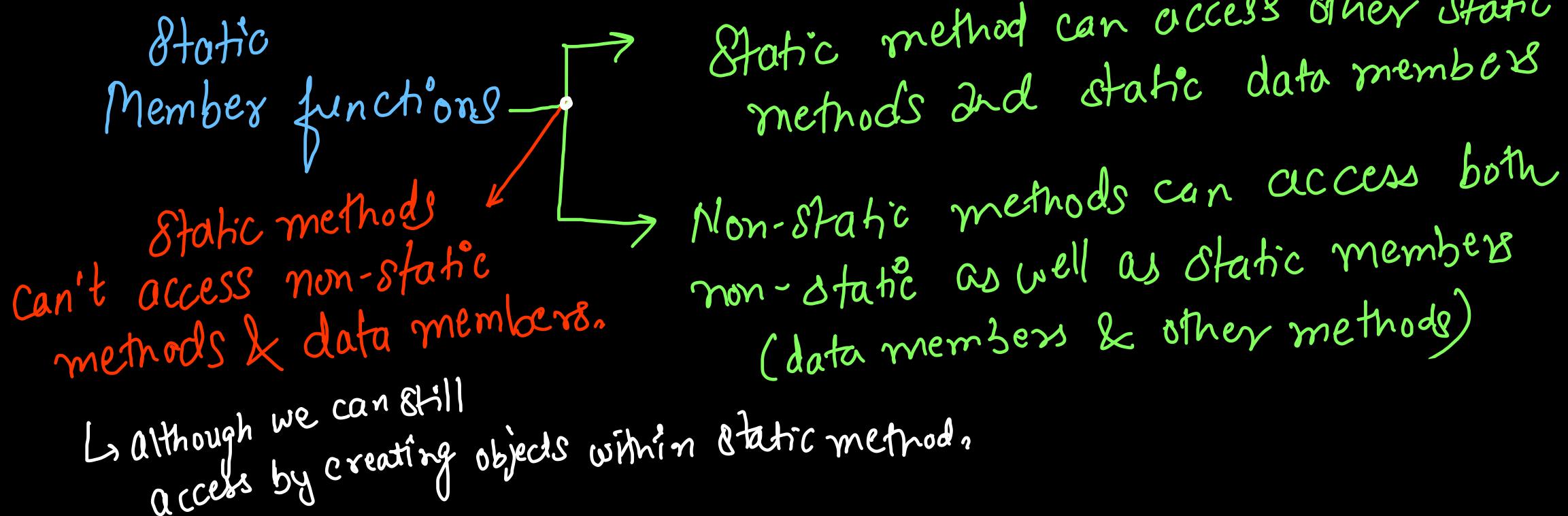
Q) Why are static variables considered evil?

- It represents global scope (accessible using class name & from all objects of class)
- lifetime of variable is very long (entire program)
- less Object-Oriented way and reduce reusability
  - class loading init
  - do not require object creation
  - not object specific
- Not thread safe and difficult to serialize
  - (multiple threads will see same instance/copy of static variables which can lead to inconsistency/race condition during concurrent/parallel access).

```
public static void staticEvil() {  
    // 1. Program LifeTime  
    // 2. Global Scope  
    // 3. Less Object Oriented (Do Not Require Object Creation)  
    System.out.println(Movie.language);  
  
    Movie a1 = new Movie(name: "Avengers Infinity War", duration: 130, rating: 4.8);  
    Movie a2 = new Movie(name: "Avengers End Game", duration: 200, rating: 4.9);  
    Movie a3 = new Movie(name: "Avengers Secret Wars", duration: 170, rating: 4.6);  
    Movie a4 = new Movie(name: "Avengers Kang Dynasty", duration: 220, rating: 5.0);  
  
    System.out.println(a1.language + " " + a2.language + " " + a3.language + " " + a4.language);  
  
    // 4. Same Copy Shared to Every Object  
    a1.language = "Hindi";  
    System.out.println(a1.language + " " + a2.language + " " + a3.language + " " + a4.language);  
  
    // 5. Not Thread Safe and Non-Serializable  
}
```

- architaggarwal@Archits-MacBook-Air 00PS\_Codes % javac 11.StaticKeyword.java
- architaggarwal@Archits-MacBook-Air 00PS\_Codes % java Driver  
English  
English English English English  
Hindi Hindi Hindi Hindi

Q) What are static member functions in Java? What are rules for accessing static or non-static data?



## Movie Class

```
public void nonStaticFun() {  
    // Access Non Static Members (Properties & Other Functions) : Allowed  
    System.out.println(" Name : " + this.name + " Duration : "  
        + this.duration + " Rating : " + this.rating);  
  
    // Access Static Member (Properties & Other Functions) : Allowed  
    System.out.println("Language : " + Movie.language);  
    Movie.staticFun();  
}  
  
public static void staticFun() {  
    // This Keyword is Not There Inside Static Functions  
    // System.out.println(this);  
  
    // Access Non Static Members (Properties & Other Functions) : Not Allowed  
    // System.out.println(" Name : " + this.name + " Duration : " + this.duration +  
    // " Rating : " + this.rating);  
    // nonStaticFun();  
  
    // Access Static Member (Properties & Other Functions) : Allowed  
    System.out.println("Language : " + Movie.language);  
  
    // Access Non Static Members Using Object Creation  
    Movie obj = new Movie(name: "Avengers", duration: 150, rating: 4.5);  
    System.out.println(obj.name + " " + obj.duration + " " + obj.rating);  
}
```

## Output

```
● architaggarwal@Archits-MacBook-Air 0OPS_Codes % javac 11.StaticKeyword.java  
● architaggarwal@Archits-MacBook-Air 0OPS_Codes % java Driver  
Language : English  
Avengers 150 4.5  
Name : Avengers Infinity War Duration : 130 Rating : 4.8  
Language : English  
Language : English  
Avengers 150 4.5
```

## Driver Code

```
// No Object Required for Calling Static Functions  
Movie.staticFun();  
  
Movie a1 = new Movie(name: "Avengers Infinity War", duration: 130, rating: 4.8);  
// Object Required to Call Non Static Member Function  
a1.nonStaticFun();
```

## # Non static methods

```
public void setLanguage(String newLanguage){ language = newLanguage; }
public String getLanguage(){ return language; }
```

(1) Using Objects → Variables & Functrn access : Allowed

```
System.out.println(avengers1.getLanguage());
avengers1.setLanguage("English");
System.out.println(avengers1.getLanguage());
```

(2) Classname → Not Allowed  
(without object creation)

```
System.out.println(Movie.getLanguage());
Movie.setLanguage("Tamil");
System.out.println(Movie.getLanguage());
```

You require objects/  
instances in order  
to access non  
static members

## # STATIC METHODS

```
private static String language = "Hindi";
public static void setLanguage(String newLanguage){ language = newLanguage; }
public static String getLanguage(){ return language; }
```

1) Using Objects/Instances → Allowed ✓

```
System.out.println(avengers1.getLanguage());
avengers1.setLanguage("English");
System.out.println(avengers1.getLanguage());
```

2) Using Classname (without object creation) → Allowed ✓

```
System.out.println(Movie.getLanguage());
Movie.setLanguage("Tamil");
System.out.println(Movie.getLanguage());
```

You do not require  
object creation to access  
static members.

Interview Ques

Q) Why is main() method static in Java?

- JVM can call it without object creation
- It is the first method to undergo execution
- JVM can call main() method directly using classname

Note :→ It should also be always →

- ① public
- ② void return type
- ③ String[] args

"public static void main(String[] args)"

Default Exception Handler

exit  
int main  
↓ -1 or +1  
normal  
terminated  
abnormal  
terminated)

Interview Question Q) Do we have "this" inside a "static" method?

A) No, since static methods are not associated with objects and this keyword is associated with a particular instance/object, hence we do not have "this" access inside "static" method.

Interview Question Q) Can constructors be static?

A) No, since constructors implicitly return the current object reference ("this"), they cannot be static.

Q) What do you understand by static nested classes in Java?

static

Nested class → To create objects (instantiate)  
inner class objects without instantiating  
outer class

- Outer class can access inner class data members
  - Inner class can only access outer class' static members
  - Inner class object creation requires:- OuterClassName.innerClassName
- Note: If main() method & inner class are in the same outer class,  
we can omit outer class name

# Outer class (Movie)

```
class Movie {  
    // Non Static or Instance Variables (Outer Class)  
    int duration;  
    String name;  
    double rating;  
  
    // Static or Class Variable (Outer Class)  
    static String language = "English";  
  
    public Movie(String name, int duration, double rating) {  
        this.name = name;  
        this.duration = duration;  
        this.rating = rating;  
    }  
  
    public void outerNonStaticFun() {...}  
    public static void outerStaticFun() {...}  
    public static class Theater {...}  
}
```

```
public void outerNonStaticFun() {  
    // Access Outer Non Static Members (Properties & Other Functions) : Allowed  
    System.out.println(" Name : " + this.name + " Duration : "  
        + this.duration + " Rating : " + this.rating);  
  
    // Access Outer Static Member (Properties & Other Functions) : Allowed  
    System.out.println("Language : " + Movie.language);  
  
    // Access Static Inner Class Static Members : Allowed  
    System.out.println(Theater.screenType);  
  
    // Access Static Inner Class Non Static Members  
    // Without Inner Class Object: Not Allowed  
    // System.out.println(this.chain + " " + this.noOfSeats);  
  
    // With Inner Class Object: Allowed  
    Theater inner = new Theater(chain: "PVR", noOfSeats: 100);  
    System.out.println(inner.chain + " " + inner.noOfSeats);  
  
    System.out.println(x: "-----");  
}
```

```
public static void outerStaticFun() {  
    // Access Outer Non Static Members (Properties & Other Functions) : Not Allowed  
    // System.out.println(" Name : " + this.name + " Duration : " + this.duration +  
    // " Rating : " + this.rating);  
    // outerNonStaticFun();  
  
    // Access Outer Static Member (Properties & Other Functions) : Allowed  
    System.out.println("Language : " + Movie.language);  
  
    // Access Static Inner Class Static Members : Allowed  
    System.out.println(Theater.screenType);  
  
    // Access Static Inner Class Non Static Members  
    // Without Inner Class Object: Not Allowed  
    // System.out.println(this.chain + " " + this.noOfSeats);  
  
    // With Inner Class Object: Allowed  
    Theater inner = new Theater(chain: "PVR", noOfSeats: 100);  
    System.out.println(inner.chain + " " + inner.noOfSeats);  
  
    System.out.println(x: "-----");  
}
```

# Inner class (Theater)

```
public static class Theater {  
    // Non Static or Instance Variables (Inner Class)  
    String chain;  
    int noOfSeats;  
  
    // Static or Class Variable (Inner Class)  
    static String screenType = "2D";  
  
    public Theater(String chain, int noOfSeats) {  
        this.chain = chain;  
        this.noOfSeats = noOfSeats;  
    }  
  
    public void innerNonStaticFun() {  
        // Access Outer Non Static Members : Not Allowed  
        // System.out.println(" Name : " + this.name + " Duration : "  
        // + this.duration + " Rating : " + this.rating);  
  
        // Access Outer Static Member (Properties & Other Functions) : Allowed  
        System.out.println("Language : " + Movie.language);  
  
        // Access Static Inner Class Static Members : Allowed  
        System.out.println(Theater.screenType);  
  
        // Access Static Inner Class Non Static Members: Allowed  
        System.out.println(this.chain + " " + this.noOfSeats);  
  
        System.out.println(x: "-----");  
    }  
}
```

```
public static void innerStaticFun() {  
    // Access Outer Non Static Members (Properties & Other Functions) : Not Allowed  
    // System.out.println(" Name : " + this.name + " Duration : " + this.duration +  
    // " Rating : " + this.rating);  
    // outerNonStaticFun();  
  
    // Access Outer Static Member (Properties & Other Functions) : Allowed  
    System.out.println("Language : " + Movie.language);  
  
    // Access Static Inner Class Static Members : Allowed  
    System.out.println(Theater.screenType);  
  
    // Access Static Inner Class Non Static Members  
    // Without Inner Class Object: Not Allowed  
    // System.out.println(this.chain + " " + this.noOfSeats);  
  
    // With Inner Class Object: Allowed  
    Theater inner = new Theater(chain: "PVR", noOfSeats: 100);  
    System.out.println(inner.chain + " " + inner.noOfSeats);  
  
    System.out.println(x: "-----");  
}
```

# Output

## Driver Code

```
class Driver {  
    Run | Debug  
    public static void main(String[] args) {  
        Movie.outerStaticFun();  
        Movie.Theater.innerStaticFun();  
  
        Movie outer = new Movie(name: "Avengers", duration: 180, rating: 4.5);  
        outer.outerNonStaticFun();  
  
        Movie.Theater inner = new Movie.Theater(chain: "PVR", noOfSeats: 100);  
        inner.innerNonStaticFun();  
    }  
}
```

Language : English

2D

PVR 100

Language : English

2D

PVR 100

Name : Avengers Duration : 180 Rating : 4.5

Language : English

2D

PVR 100

Language : English

2D

PVR 100

Q) What do you understand by "static block" in Java?

→ static block

- ↳ used to initialize static variables
- ↳ executed before main() execution
- ↳ at time of class loading

Q) Can you run a java program only using static block, ie. without main function?

No, in newer Java versions, main() is mandatory to run a java program. If there is no entry point, java code will not run even if there is a static block.

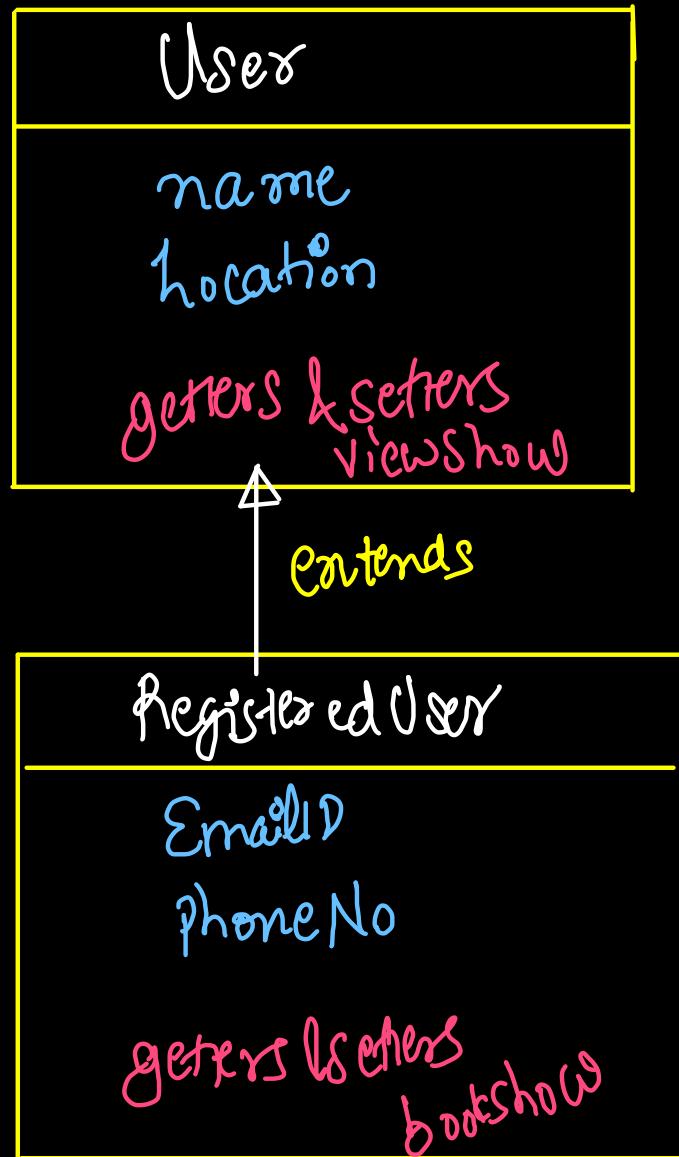
```
class Movie {  
    // Non Static or Instance Variables  
    int duration;  
    String name;  
    double rating;  
  
    // Static or Class Variable  
    static String language;  
  
    public Movie(String name, int duration, double rating) {  
        System.out.println(x: "Constructor - Object Creation");  
        this.name = name;  
        this.duration = duration;  
        this.rating = rating;  
        this.language = "Hindi";  
    }  
  
    // Static Block (Runs Before Main Method Execution: During Loading & Linking)  
    static {  
        System.out.println(x: "Movie Static Block Executed");  
  
        // Non Static Member Initialized  
        language = "English";  
  
        System.out.println(language + " OR " + Movie.language);  
    }  
}
```

```
class Driver {  
    static int x;  
    static {  
        System.out.println(x: "Driver Static block executed");  
        x = 100;  
        System.out.println(x);  
    }  
  
    Run | Debug  
    public static void main(String[] args) {  
        System.out.println(x: "Main Method Started");  
  
        // main method must be there otherwise static block will also not run,  
        // Although it will run after static block of its class  
  
        System.out.println(x);  
        System.out.println(Movie.language);  
  
        Movie obj = new Movie(name: "Avengers", duration: 180, rating: 4.5);  
        System.out.println(obj.language);  
    }  
}
```

- architaggarwal@Archits-MacBook-Air 0OPS Codes % javac 11.3.StaticBlock.java
- architaggarwal@Archits-MacBook-Air 0OPS Codes % java Driver  
Driver Static block executed  
100  
Main Method Started  
100  
Movie Static Block Executed  
English OR English  
English  
Constructor - Object Creation  
Hindi

Q) What do you mean by INHERITANCE in Java? What is super/  
parent class & child/sub class? Give some real life examples.

- Way in which two classes can be related with each other. (is A relationship)
- super class or parent class is being inherited by child class
- super class or parent class properties & behavior of parent class are  
acquired by child class directly or indirectly.
- Hence, it improves reusability, readability, extensibility  
and maintainability of code.
- Along with reusing parent properties, we can add new  
properties & methods in child class.
- Inheritance is implemented to achieve generalization, to  
implement run-time polymorphism



Parent class / Super class  
Non-Registered  
Non-Authenticated  
Anonymous

Child class / Sub class  
Registered User  
Authenticated User

```
class User {  
    String name;  
    String location;  
  
    public User(String name, String location) {  
        this.name = name;  
        this.location = location;  
    }  
  
    public User() {  
        this.name = "Anonymous";  
        this.location = "India";  
    }  
  
    public void viewShow() {  
        System.out.println("I can view listing shows on app");  
    }  
  
    class RegisteredUser extends User {  
        String emailId;  
        long phoneNo;  
  
        public RegisteredUser() {  
            this.emailId = "registeredUser@gmail.com";  
            this.phoneNo = 9319117888l;  
        }  
  
        public RegisteredUser(String emailId, long phoneNo) {  
            this.emailId = emailId;  
            this.phoneNo = phoneNo;  
        }  
  
        public void bookShow() {  
            System.out.println("I can book the shows on app");  
        }  
    }  
}
```

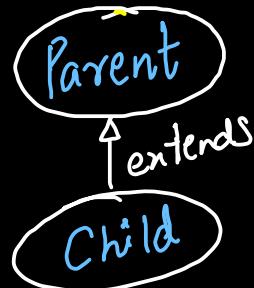
```
public class Solution {  
    Run | Debug  
    public static void main(String[] args) {  
        User obj1 = new User();  
        obj1.name = "Archit Aggarwal";  
        obj1.location = "Delhi";  
        System.out.println(obj1.name + " " + obj1.location);  
        obj1.viewShow();  
  
        // Compilation Error  
        // System.out.println(obj1.phoneNo);  
        // System.out.println(obj1.emailId);  
        // obj1.bookShow();  
  
        RegisteredUser obj2 = new RegisteredUser();  
        obj2.name = "Shahrukh Khan";  
        obj2.location = "Mumbai";  
        obj2.phoneNo = 9319117889l;  
        obj2.emailId = "archit.aggarwal023@gmail.com";  
  
        System.out.println(obj2.name + " " + obj2.location);  
        System.out.println(obj2.phoneNo + " " + obj2.emailId);  
  
        obj2.viewShow();  
        obj2.bookShow();  
    }  
}
```

```
● architaggarwal@Archits-MacBook-Air System Design % javac Solution.java  
● architaggarwal@Archits-MacBook-Air System Design % java Solution  
Archit Aggarwal Delhi  
I can view listing shows on app  
Shahrukh Khan Mumbai  
9319117889 archit.aggarwal023@gmail.com  
I can view listing shows on app  
I can book the shows on app
```

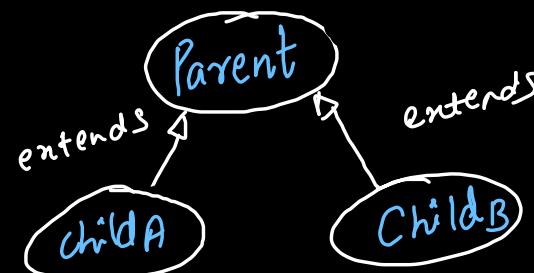
#  
Single  
level  
Inheritance

Q) What are the types of inheritance. Are all allowed in Java?  
Give some real world examples.

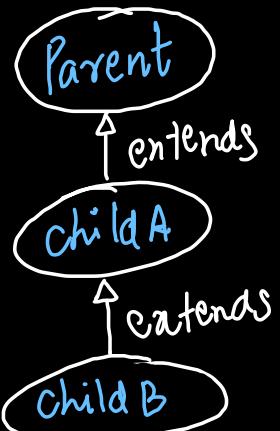
① Single level Inheritance



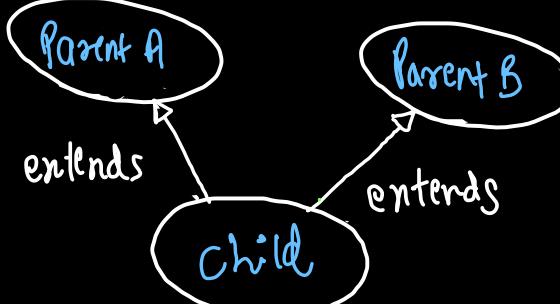
③ Hierarchical Inheritance



② Multilevel Inheritance



L ④ Multiple Inheritance



## # Multilevel Inheritance



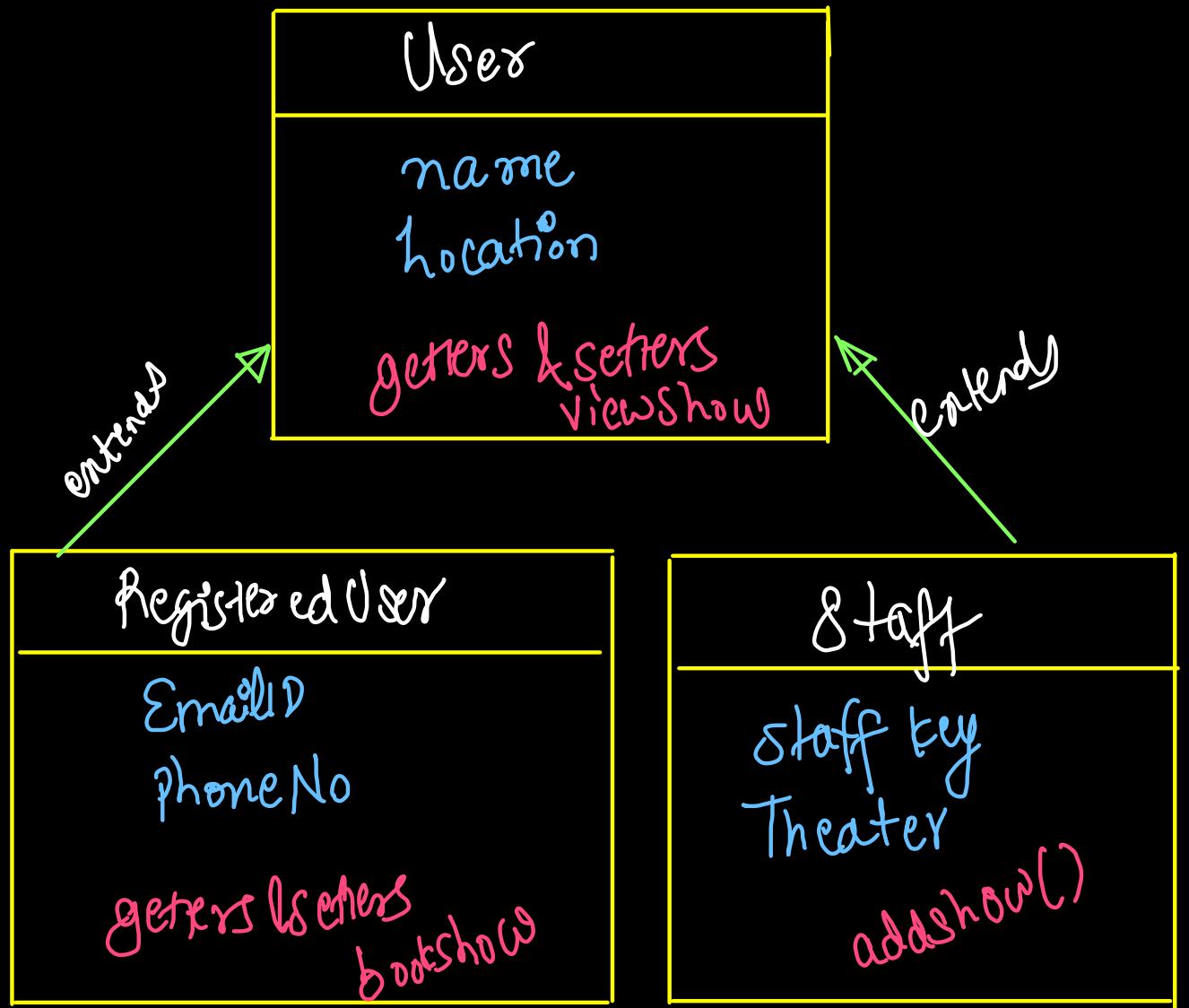
```
class SubscribedUser extends RegisteredUser {
    int validity;
    String subscriptionType;

    public SubscribedUser() {
        validity = 365;
        subscriptionType = "Mobile";
    }

    public void watchShow() {
        System.out.println("I have paid, hence I can watch the movies on OTT");
    }
}
```

● architaggarwal@Archits-MacBook-Air: System Design % java Solution  
registeredUser@gmail.com India Anonymous 9319117888 Mobile 365  
I can view listing shows on app  
I can book the shows on app  
I have paid, hence I can watch the movies on OTT

## # Hierarchical Inheritance



```
public static void main(String[] args) {
    User obj1 = new User();
    obj1.name = "Archit Aggarwal";
    obj1.location = "Delhi";
    System.out.println(obj1.name + " " + obj1.location);
    obj1.viewShow();

    RegisteredUser obj2 = new RegisteredUser();
    obj2.name = "Shahrukh Khan";
    obj2.location = "Mumbai";
    obj2.phoneNo = 9319117889l;
    obj2.emailId = "archit.aggarwal023@gmail.com";

    System.out.println(obj2.name + " " + obj2.location);
    System.out.println(obj2.phoneNo + " " + obj2.emailId);

    obj2.viewShow();
    obj2.bookShow();

    Staff obj3 = new Staff();

    System.out.println(obj3.name + " " + obj3.location);
    System.out.println(obj3.staffId + " " + obj3.theater);

    obj3.viewShow();
    obj3.addShow();

    // Compilation Error
    // System.out.println(obj3.phoneNo + " " + obj3.emailId);
    // obj3.bookShow();
    // System.out.println(obj2.staffId + " " + obj2.theater);
    // obj2.addShow();
}
```

```
class Staff extends User {
    int staffId = 707;
    String theater = "PVR Cinemas";

    public void addShow() {
        System.out.println("I Can Add Show in My Theater");
    }
}
```

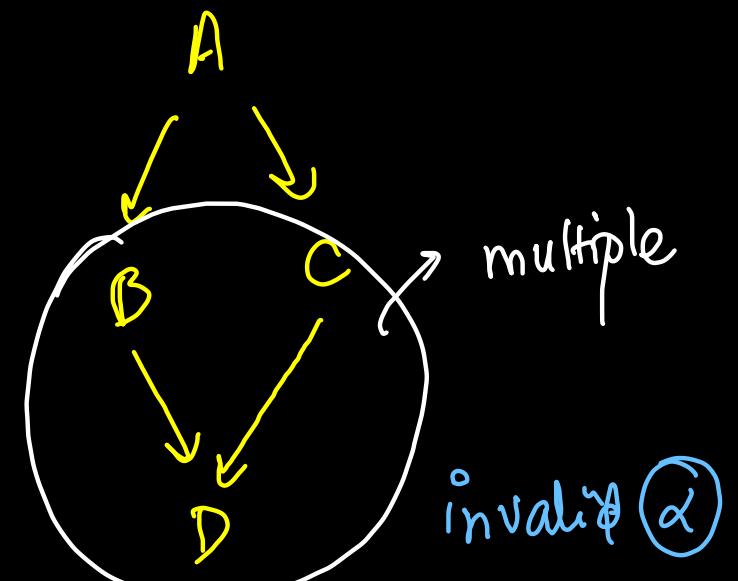
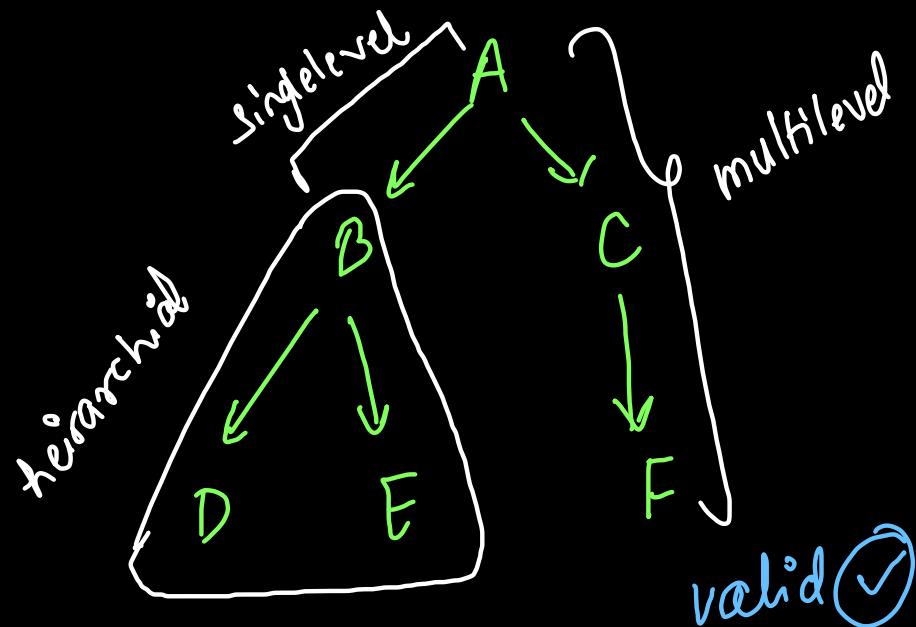
- architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
- architaggarwal@Archits-MacBook-Air System Design % java Solution

Archit Aggarwal Delhi  
I can view listing shows on app  
Shahrukh Khan Mumbai  
9319117889 archit.aggarwal023@gmail.com  
I can view listing shows on app  
I can book the shows on app  
Anonymous India  
707 PVR Cinemas  
I can view listing shows on app  
I Can Add Show in My Theater

```
// Multiple Inheritance is Not Possible in Java, We cannot extend 2 or more classes in same class  
// Compilation Error  
// class Admin extends RegisteredUser, Staff{  
// }
```

Multiple inheritance & not allowed in Java

## ⑤ Hybrid Inheritance

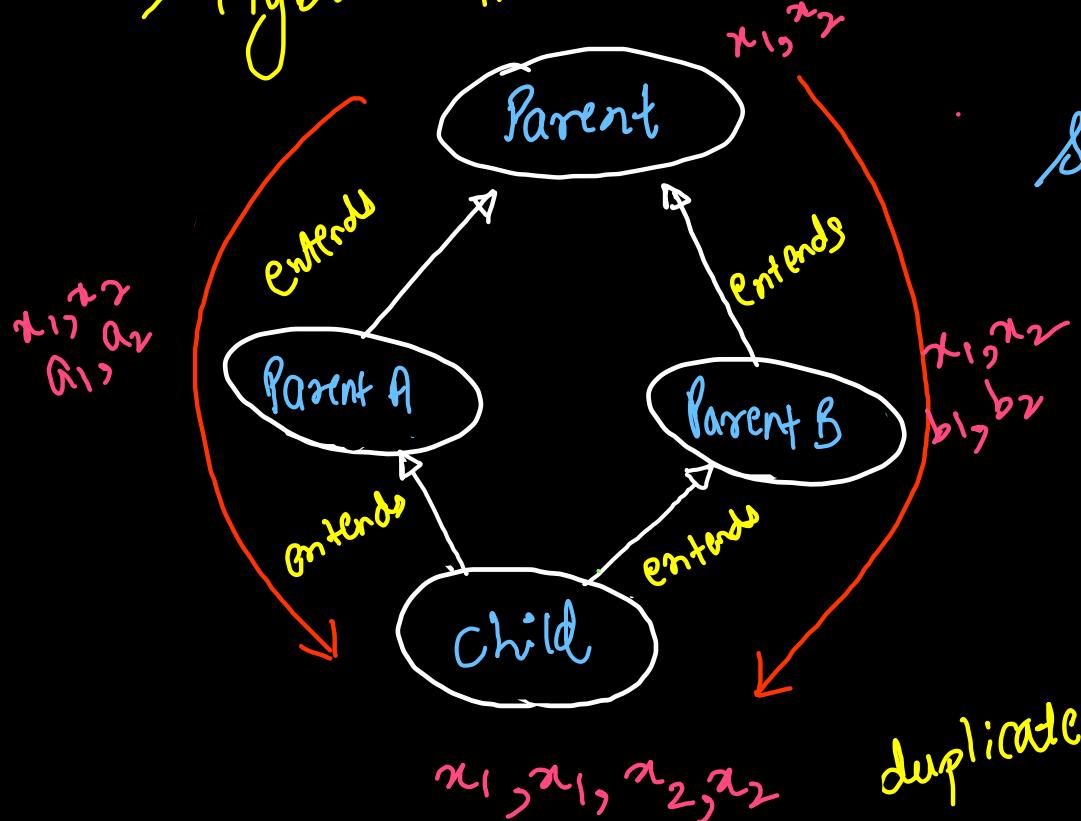


Interview Question →  
Why multiple inheritance using classes is not supported in Java?

Q) Why multiple inheritance using classes is not supported in Java?

A) Diamond Problem or Deadly Diamond of Death.

⇒ Hybrid Inheritance



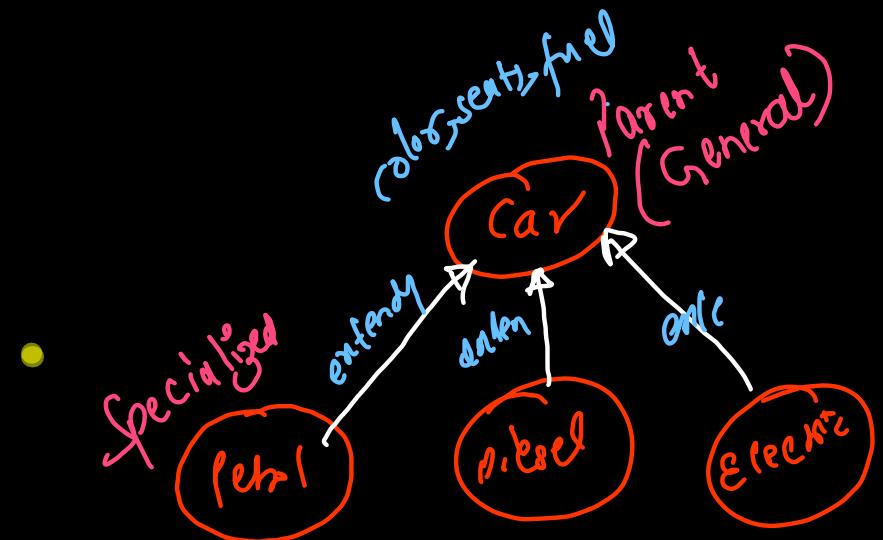
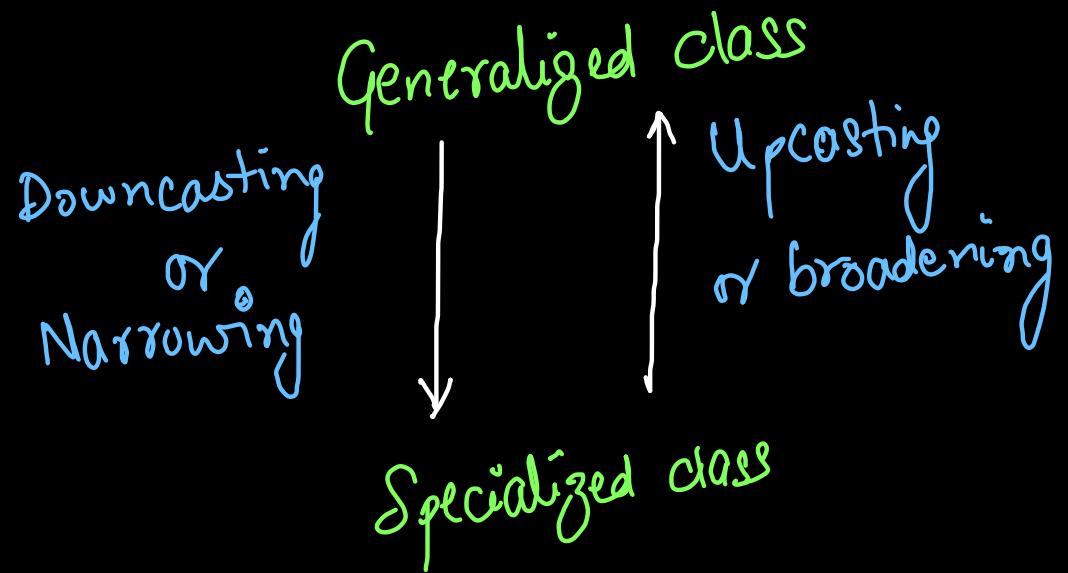
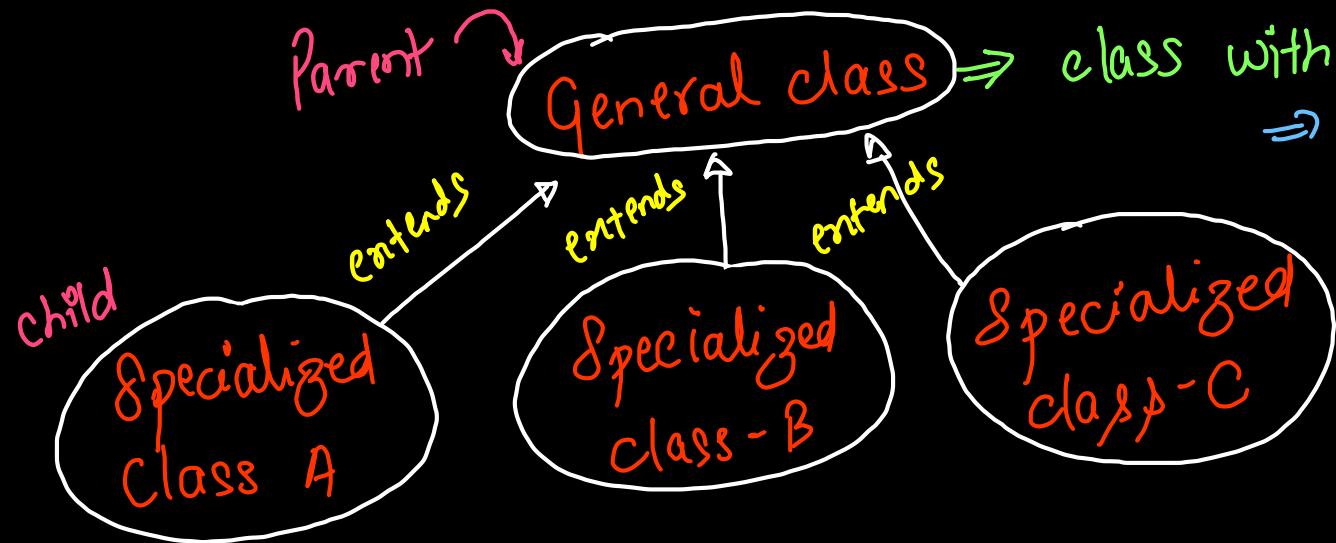
⇒ Ambiguity in parent properties or methods if name collision occurs.

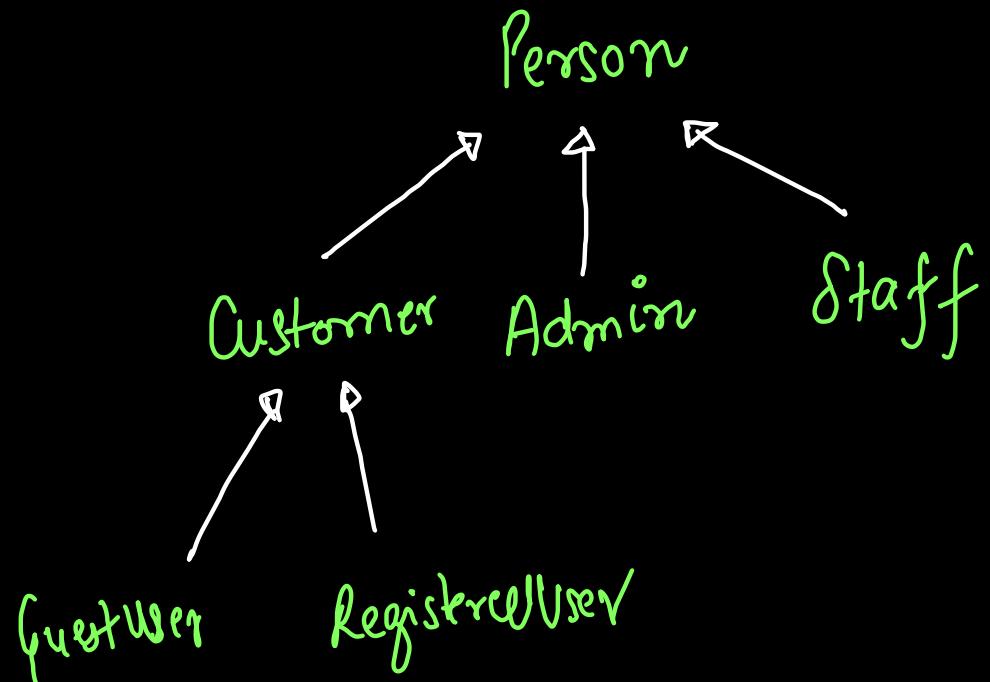
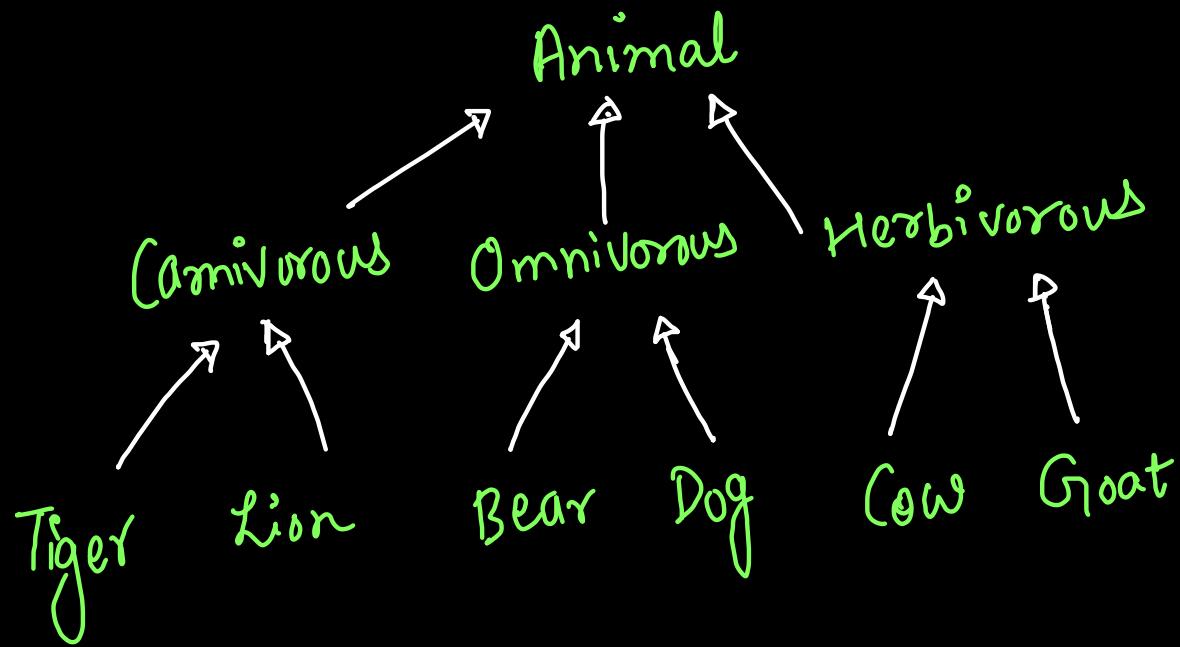
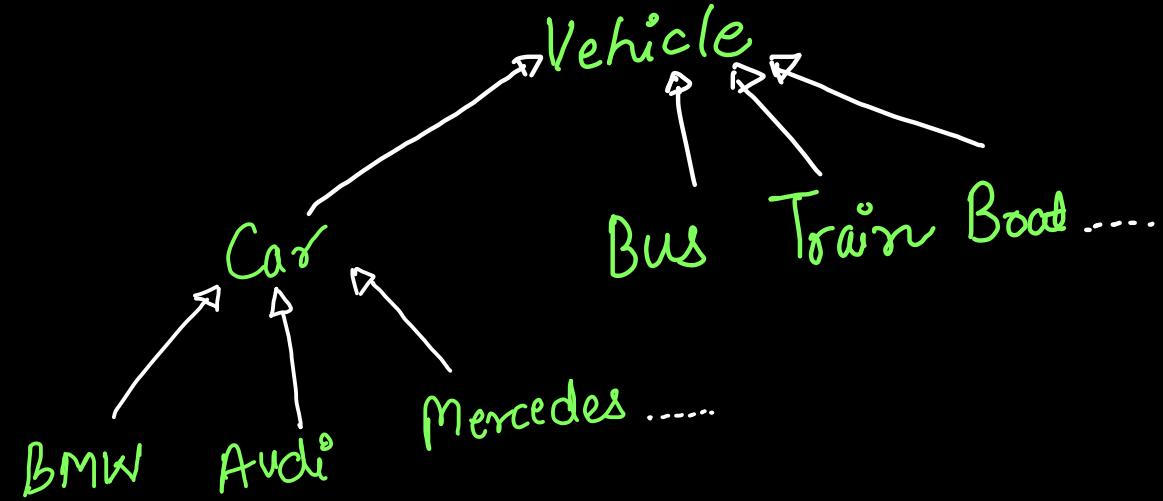
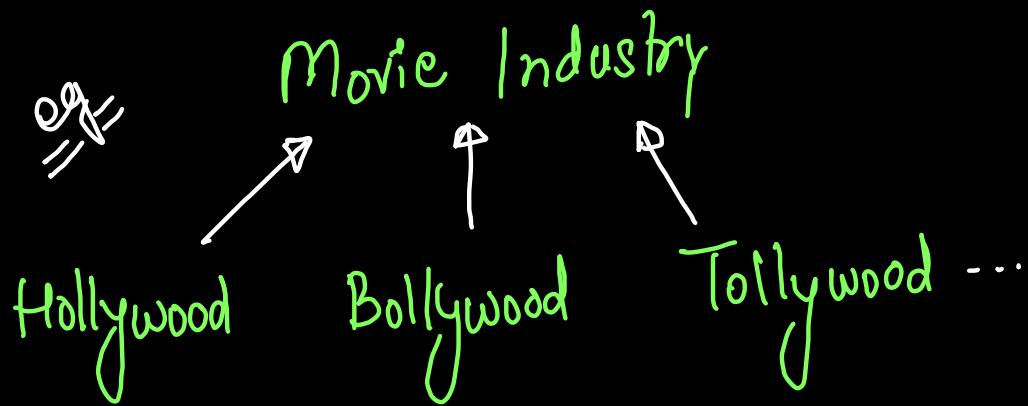
Solution: → Multiple interfaces cannot be implemented in a same class.

Child obj = new Child();  
x1 via Parent A  
x1 via Parent B

duplicate ⇒ ambiguity

## Generalization & Specialization





## Constructors in Inheritance

- 1) Are constructors of parent class inherited in child classes?  
    ↳ not inherited
- 2) Are constructors of parent class accessible in child class object?  
    ↳ not directly accessible  
    ↳ but indirectly accessible via constructor chaining
- 3) What do you mean by "super" keyword in Java?  
    reference variable used to refer immediate parent's  
    class object.  
    ↳ uses → access parent's instance variables  
                  → invoke parent's methods  
                  → invoke parent's constructor

```
class Parent {  
    String parentData = "Parent";  
  
    Parent() {  
        System.out.println("Parent's Constructor");  
        System.out.println(this.parentData);  
    }  
}  
  
You, 1 second ago | 1 author (You)  
class Child extends Parent {  
    String childData = "Child";  
  
    Child() {  
        super();  
        System.out.println("Child's Constructor");  
        System.out.println(this.childData);  
        System.out.println(super.parentData);  
    }  
}
```

You, 33 seconds ago | 1 author (You)  
class Driver {  
 Run | Debug  
 public static void main(String[] args) {  
 Child obj = new Child();  
 }  
}

You, 35 seconds ago • Uncommitted changes

Parent's Constructor  
Parent  
Child's Constructor  
Child  
Parent

```
class Parent {  
    String parentData = "Parent";  
  
    Parent() {  
        System.out.println("Parent's Constructor");  
        System.out.println(this.parentData);  
    }  
  
    public void parentFun() {  
        System.out.println("This is Parent's Method");  
    }  
}
```

You, 29 seconds ago | 1 author (You)

```
class Child extends Parent {  
    String childData = "Child";  
  
    Child() {  
        super(); // parent's constructor  
        System.out.println("Child's Constructor");  
        System.out.println(this.childData);  
  
        System.out.println(super.parentData); // Access Parent's Data Members  
  
        super.parentFun(); // Access Parent's Method  
    }  
}
```

You, 33 seconds ago | 1 author (You)

```
class Driver {  
    Run | Debug  
    public static void main(String[] args) {  
        Child obj = new Child();  
    }  
}
```

You, 35 seconds ago • Uncommitted changes

Parent's Constructor  
Parent  
Child's Constructor  
Child  
Parent  
This is Parent's Method

4) When creating superclass object, how many constructors are called? How much memory is allocated?

5) When creating childclass object, how many constructors are called? How much memory is allocated?

# Creating any object will execute constructors all of its constructors along with its own constructor because all properties of parent class are accessible.

```
class User {  
    String name;  
    String location;  
  
    public User() {  
        this.name = "Anonymous";  
        this.location = "India";  
        System.out.println("Guest User Created");  
    }  
  
    public void viewShow() {  
        System.out.println("I can view listing shows on app");  
    }  
}
```

```
class RegisteredUser extends User {  
    String emailId;  
    long phoneNo;  
  
    public RegisteredUser() {  
        super();  
        this.emailId = "registeredUser@gmail.com";  
        this.phoneNo = 9319117888L;  
        System.out.println("Registered User Created");  
    }  
  
    public void bookShow() {  
        System.out.println("I can book the shows on app");  
    }  
}
```

① User() {  
 User user1 = new User();  
 System.out.println(user1.name);  
 System.out.println(user1.location);  
 user1.viewShow();  
}  
  
② RegisteredUser() {  
 RegisteredUser user2 = new RegisteredUser();  
 System.out.println(user2.name);  
 System.out.println(user2.location);  
 user2.viewShow();  
  
 System.out.println(user2.emailId);  
 System.out.println(user2.phoneNo);  
 user2.bookShow();  
}

```
Guest User Created  
Anonymous  
India  
I can view listing shows on app  
Guest User Created  
Registered User Created  
Anonymous  
India  
I can view listing shows on app  
registeredUser@gmail.com  
9319117888  
I can book the shows on app
```