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

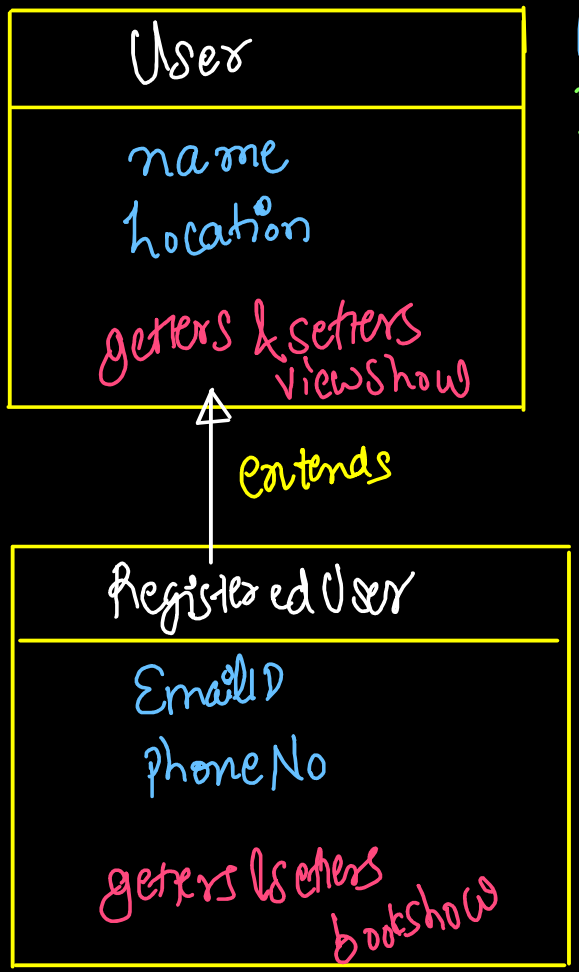
→ Way in which two classes can be related with each other. (is A relationship)

→ Super class or parent class is being inherited by child class or sub class. Properties & Behavior of parent class are acquired by child class directly or indirectly.

→ Hence, it improves reusability, readability, extendibility 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
logined user
Authenticated User

```java
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(x: "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(x: "I can book the shows on app");
    }
}
```

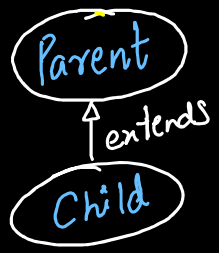
```java
public class Solution {
    Run | Debug
    public static void main(String[] args) {
        User obj1 = new User();
        obj1.name = "Archit Aggarwal";
        obj1.location = "Delhi";
        System.out.println(obj1.name + " " + obj1.location);
        obj1.viewShow();

        // 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();
    }
}
```

Single level Inheritance

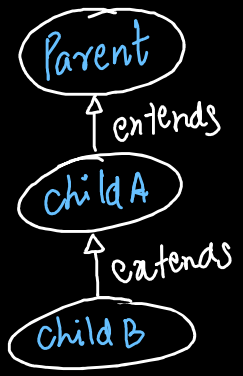
```
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
Archit Aggarwal Delhi
I can view listing shows on app
Shahrukh Khan Mumbai
9319117889 archit.aggarwal023@gmail.com
I can view listing shows on app
I can book the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

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

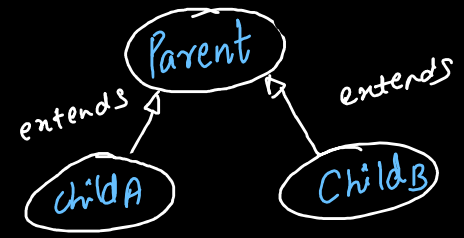
① Single level Inheritance



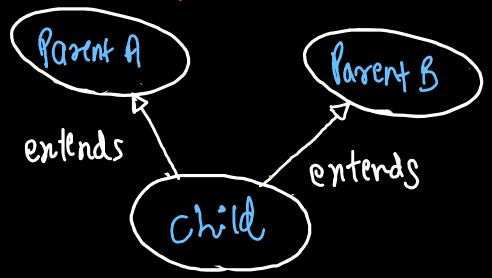
② Multilevel Inheritance



③ Hierarchial Inheritance



④ Multiple Inheritance

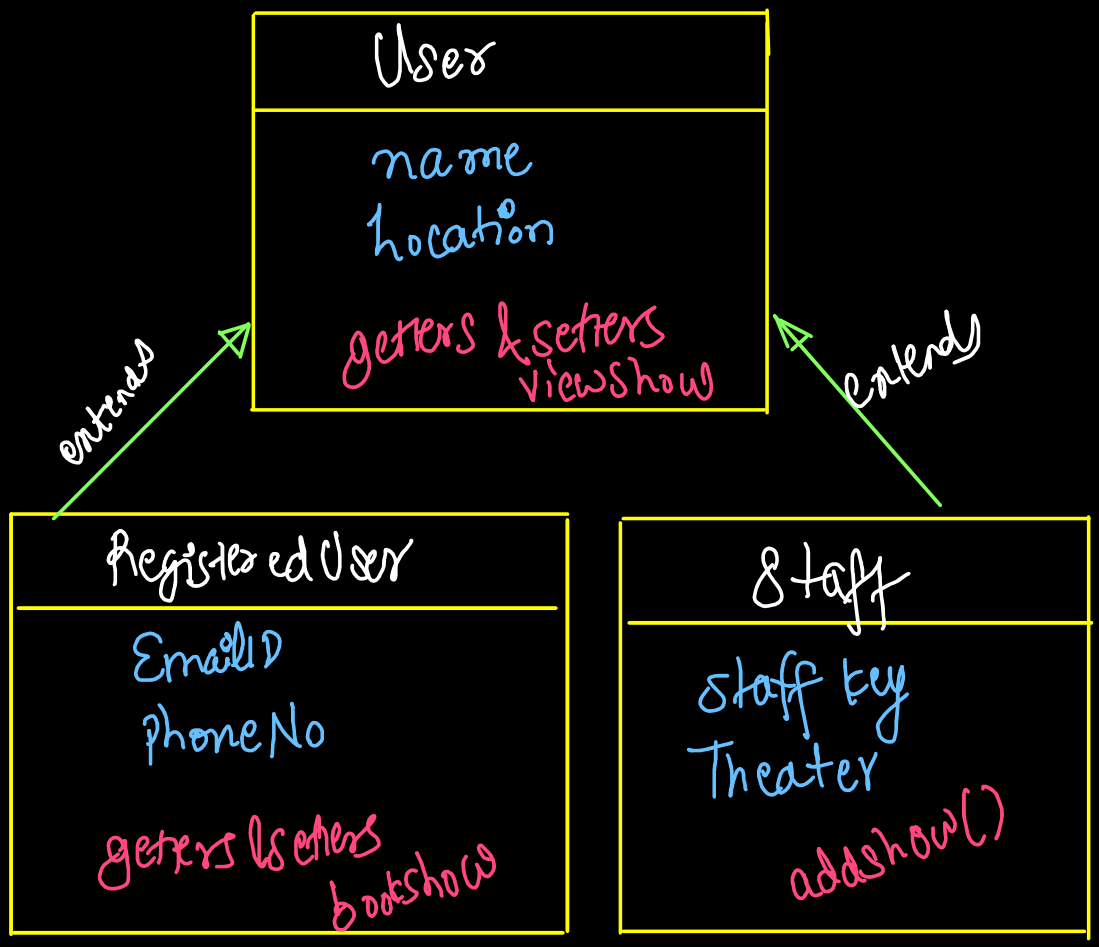


Multilevel Inheritance


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

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

    public void watchShow() {
        System.out.println(x: "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
```

Heirarchial Inheritance



```java
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();
}
```

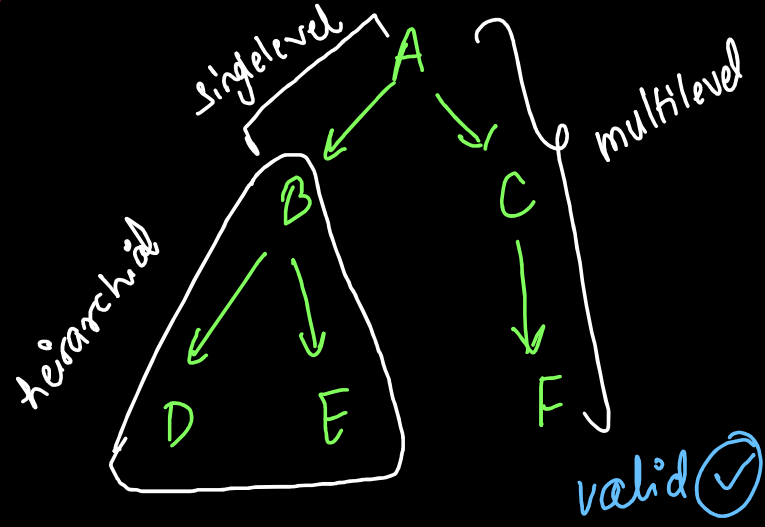
```java
class Staff extends User {
    int staffId = 707;
    String theater = "PVR Cinemas";

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

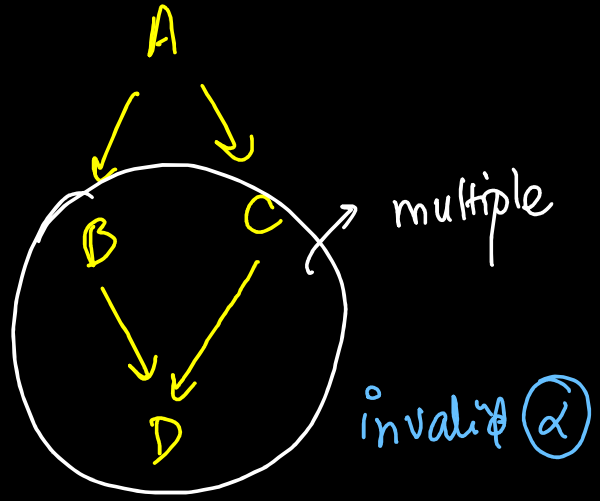
```
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
Archit Aggarwal Delhi
I can view listing shows on app
Shahrukh Khan Mumbai
9319117889 archit.aggarwal023@gmail.com
I can view listing shows on app
I can book the shows on app
Anonymous India
707 PVR Cinemas
I can view listing shows on app
I Can Add Show in My Theater
```

```java
// 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



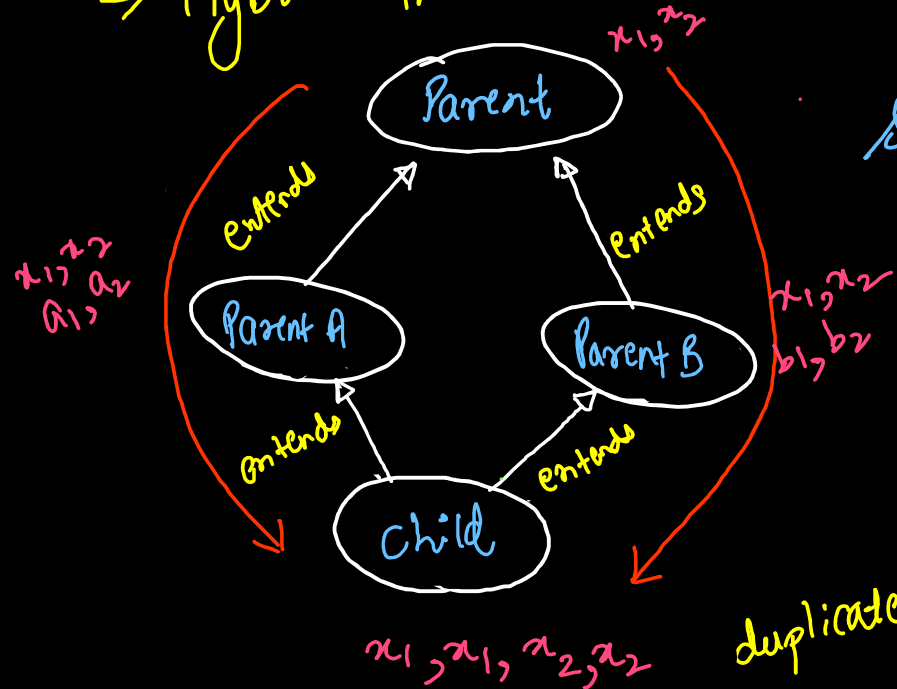


Interview Question :→

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.

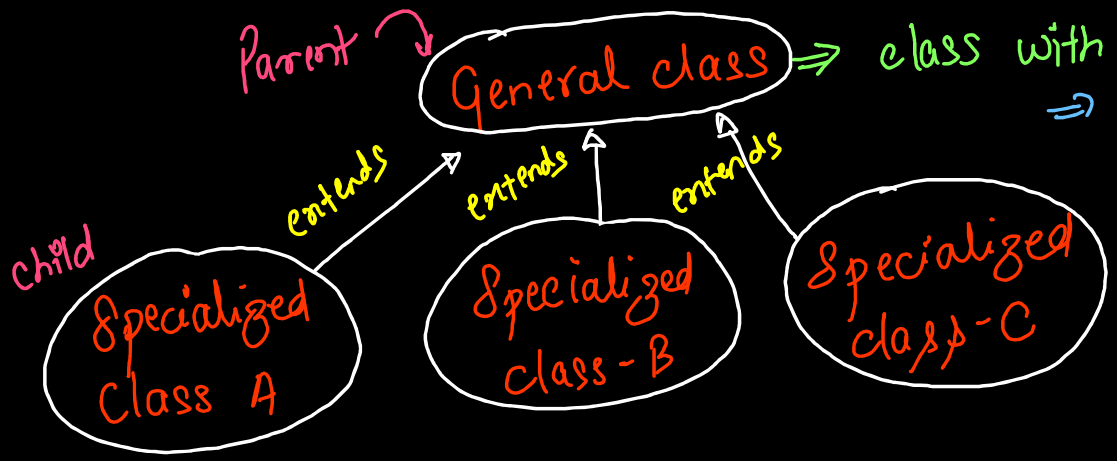
```
Child obj = new Child();
```

x_1 via Parent A

x_1 via Parent B

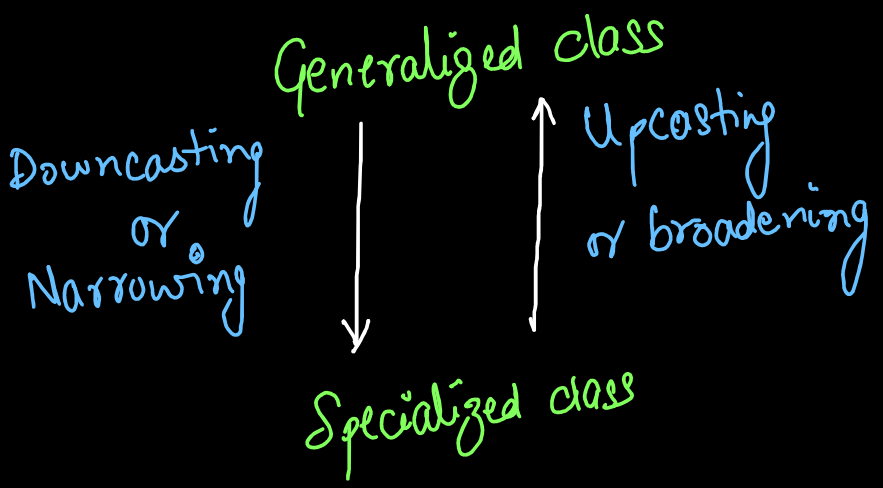
duplicate ⇒ ambiguity

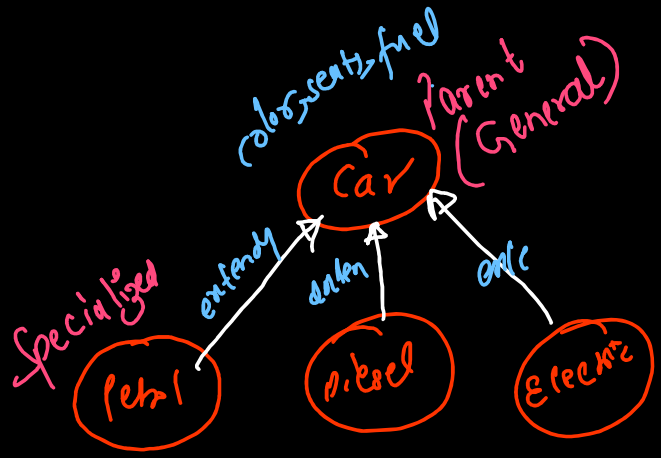
Generalization & Specialization

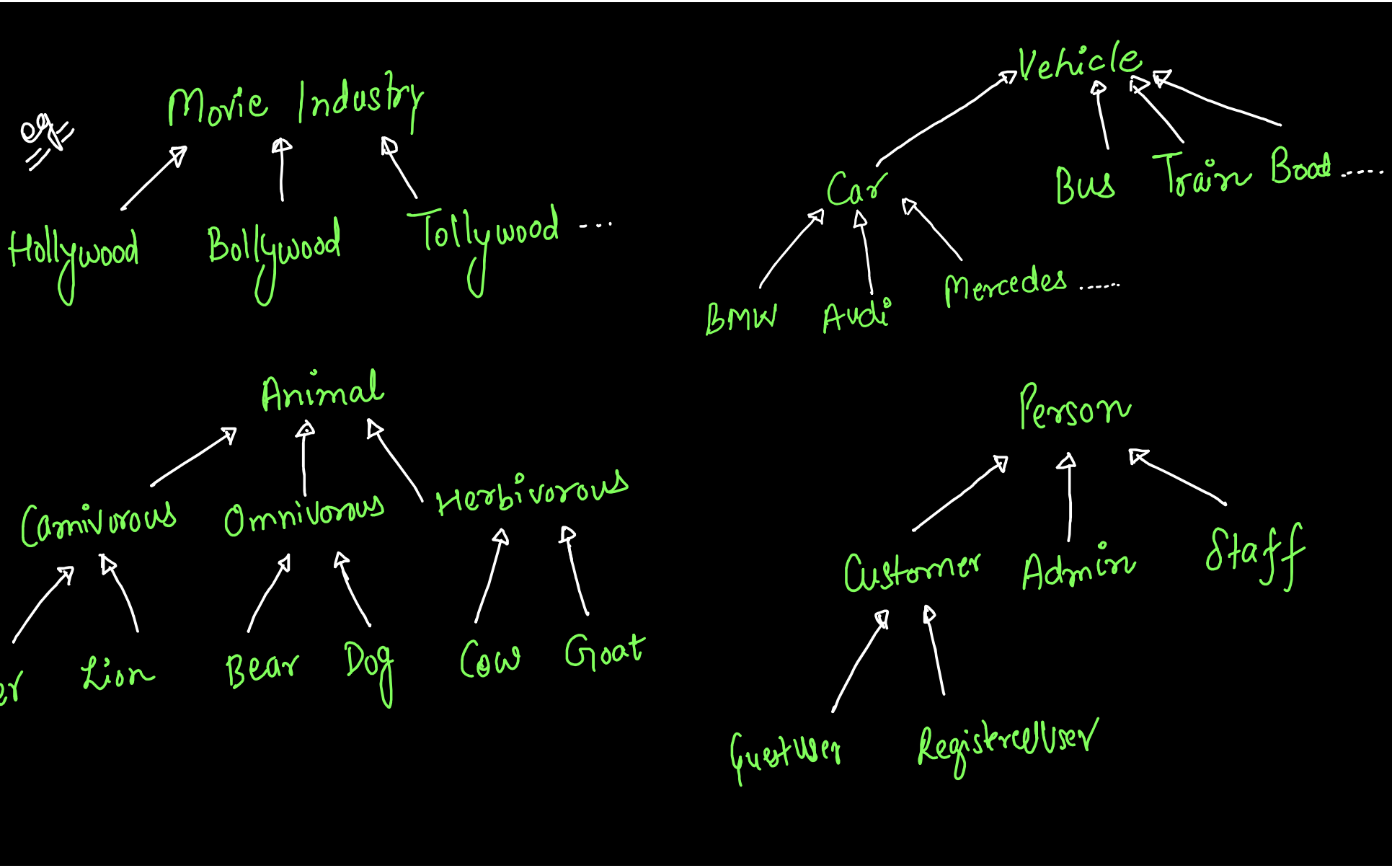


⇒ class with generic/main properties & functions
⇒ usually does not exist in real world.

⇒ classes which are specific, having particular states!
⇒ exist in real world






Movie Industry
Hollywood
Bollywood
Tollywood ...
Vehicle
Car
Bus
Train
Boat .....
BMW
Audi
Mercedes .....
Animal
Carnivorous
Omnivorous
Herbivorous
Tiger
Lion
Bear
Dog
Cow
Goat
Person
Customer
Admin
Staff
GuestUser
RegisteredUser

Constructors in Inheritance

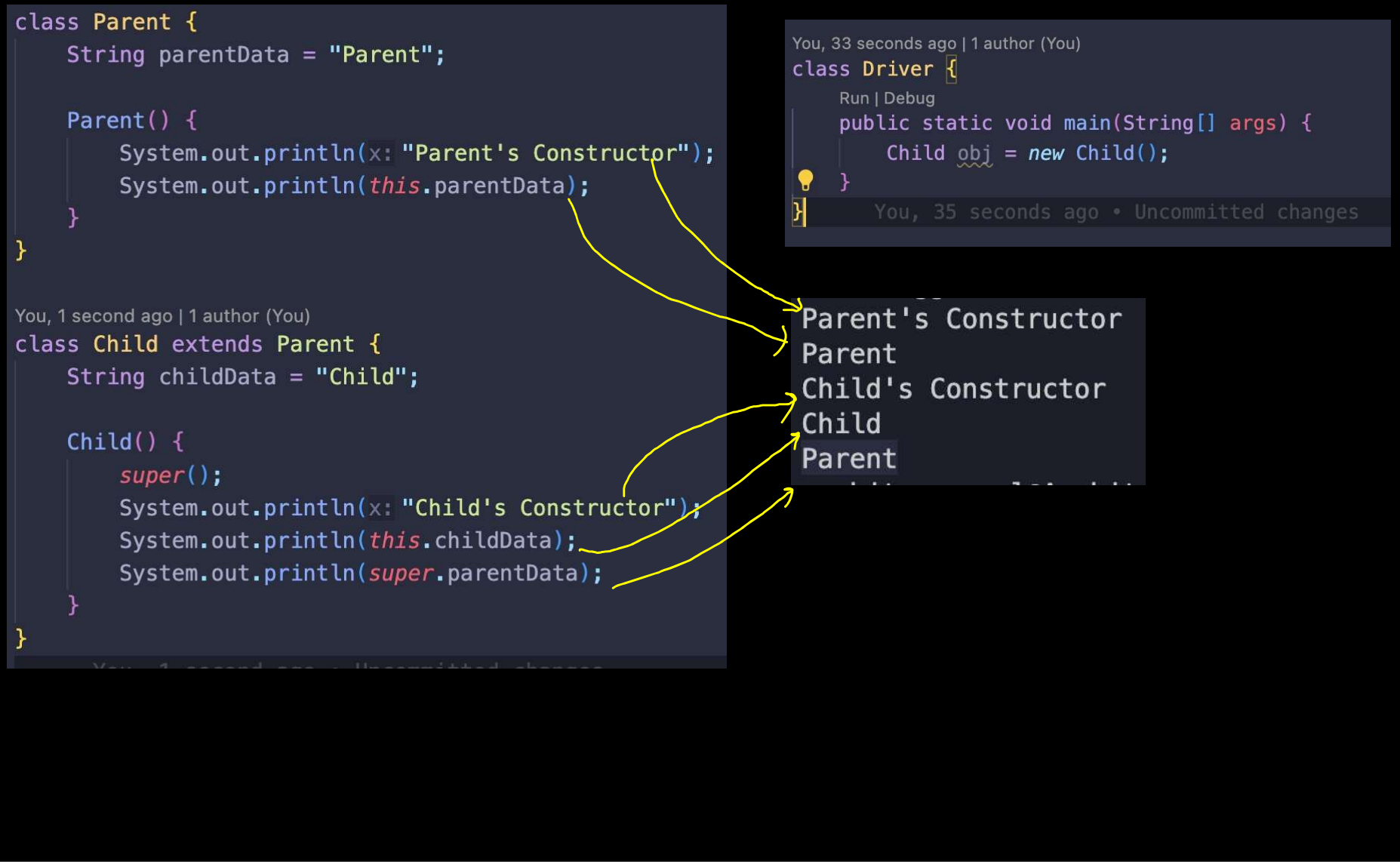
1) Are constructors of parent class inherited in child classes?
   - not inherited

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(x: "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(x: "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

```java
class Parent {
    String parentData = "Parent";

    Parent() {
        System.out.println(x: "Parent's Constructor");
        System.out.println(this.parentData);
    }

    public void parentFun() {
        System.out.println(x: "This is Parent's Method");
    }
}
```

```java
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
```

```java
You, 29 seconds ago | 1 author (You)
class Child extends Parent {
    String childData = "Child";

    Child() {
        super(); // parent's constructor
        System.out.println(x: "Child's Constructor");
        System.out.println(this.childData);

        System.out.println(super.parentData); // Access Parent's Data Members

        super.parentFun(); // Access Parent's Method
    }
}
```

```
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 it's constructors along with it's own constructor because all properties of parent class are accessible!

```java
class User {
    String name;
    String location;

    public User() {
        this.name = "Anonymous";
        this.location = "India";
        System.out.println(x: "Guest User Created");
    }

    public void viewShow() {
        System.out.println(x: "I can view listing shows on app");
    }
}
```

```java
class RegisteredUser extends User {
    String emailId;
    long phoneNo;

    public RegisteredUser() {
        this.emailId = "registeredUser@gmail.com";
        this.phoneNo = 9319117888l;
        System.out.println(x: "Registered User Created");
    }

    public void bookShow() {
        System.out.println(x: "I can book the shows on app");
    }
}
```

super();

```java
class Driver {
    Run | Debug
    public static void main(String[] args) {
        User user1 = new User();
        System.out.println(user1.name);
        System.out.println(user1.location);
        user1.viewShow();

        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();
    }
}
```

① User()

② User()
Register()

16 bytes

32 bytes

```
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
```

6) What is the order of invokation of constructors in inheritance? Is it same as order of constructor execution & object creation?

7) How to pass parameters & custom input to parent class refer variables, ie. parameterized super constructor?

8) Can we write super constructor call anywhere in the child class constructor?

```java
class User {
    String name;
    String location;

    public User() {
        this.name = "Anonymous";
        this.location = "India";
        System.out.println(x: "Guest User Created");
    }

    public void viewShow() {
        System.out.println(x: "I can view listing shows on app");
    }
}
```

```java
class RegisteredUser extends User {
    String emailId;
    long phoneNo;

    public RegisteredUser() {
        this.emailId = "registeredUser@gmail.com";
        this.phoneNo = 9319117888l;
        System.out.println(x: "Registered User Created");
    }

    public void bookShow() {
        System.out.println(x: "I can book the shows on app");
    }
}
```

super();

RegisteredUser user = new RegisteredUser();

Constructor Invocation/Calling

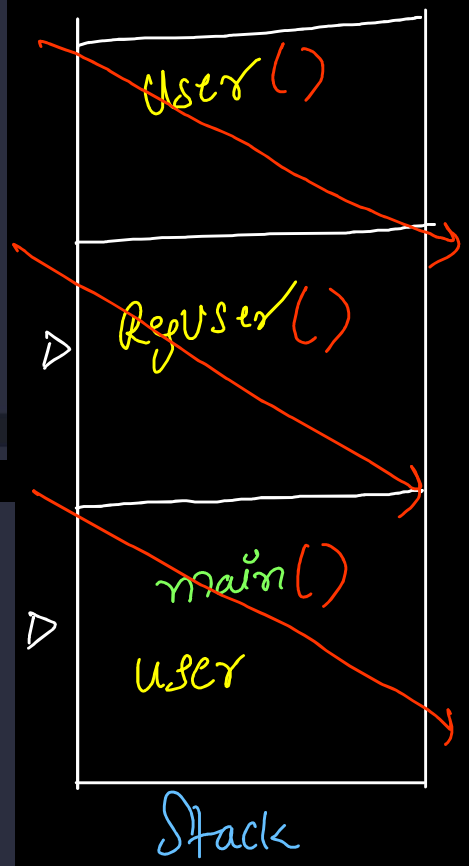
1. RegisteredUser (Child)
2. User (Parent)

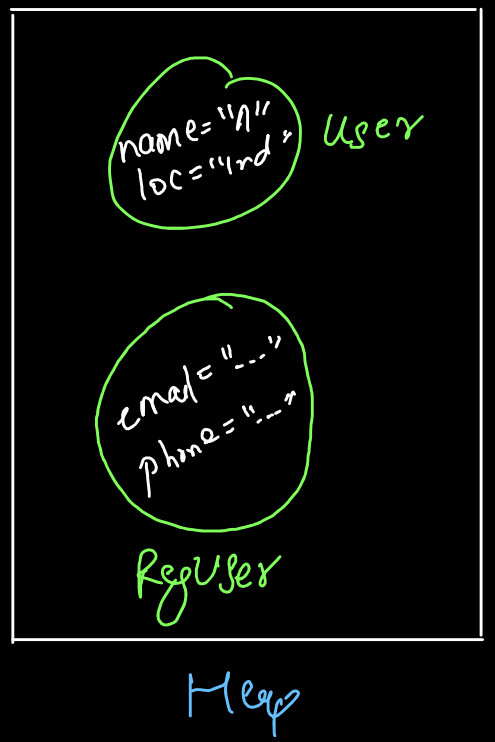
Constructor Execution

1. User (Parent)
2. RegisteredUser (Child)

```java
class User {
    String name;
    String location;

    public User() {
        this.name = "Anonymous";
        this.location = "India";
        System.out.println(x: "Guest User Created");
    }

    public void viewShow() {
        System.out.println(x: "I can view listing shows on app");
    }
}
```

```java
class RegisteredUser extends User {
    String emailId;
    long phoneNo;

    public RegisteredUser() {
        super();
        this.emailId = "registeredUser@gmail.com";
        this.phoneNo = 9319117888l;
        System.out.println(x: "Registered User Created");
    }

    public void bookShow() {
        System.out.println(x: "I can book the shows on app");
    }
}
```





① Guest User Created (Parent)

② RegUser Created (Child)

```java
class User {
    String name;
    String location;

    public User() {
        this.name = "Anonymous";
        this.location = "India";
        System.out.println(x: "Guest User Created");
    }

    public User(String name, String location) {
        this.name = name;
        this.location = location;
    }
```

```java
RegisteredUser user3 = new RegisteredUser(emailId: "archit.aggarwal@gmail.com",
phoneNo: 9319117889l);
System.out.println(user3.name);
System.out.println(user3.location);
System.out.println(user3.emailId);
System.out.println(user4.phoneNo);

RegisteredUser user4 = new RegisteredUser(name: "archit", location: "Delhi",
emailId: "archit@gmail.com", phoneNo: 9319117889l);
System.out.println(user4.name);
System.out.println(user4.location);
System.out.println(user4.emailId);
System.out.println(user4.phoneNo);
```

```java
class RegisteredUser extends User {
    String emailId;
    long phoneNo;

    public RegisteredUser() {
        // super();
        this.emailId = "registeredUser@gmail.com";
        this.phoneNo = 9319117888l;
        System.out.println(x: "Registered User Created");
    }

    public RegisteredUser(String emailId, long phoneNo) {
        // super();
        this.emailId = emailId;
        this.phoneNo = phoneNo;
    }

    public RegisteredUser(String name, String location, String emailId, long phoneNo) {
        super(name, location);
        this.emailId = emailId;
        this.phoneNo = phoneNo;
    }
```

```
Anonymous
India
archit.aggarwal@gmail.com
9319117889
archit
Delhi
archit@gmail.com
9319117889
```

Q) What is the OBJECT CLASS in Java? Give the methods provided by it.

→ Root of the class heirarchy in Java, ie. topmost class.

→ If not specified, every class (both already defined or user defined) will have parent class as Object class.

→ Object class is parent class for every class directly or indirectly, ie every class will inherit Object class methods by default.

→ It can be used to refer to any class' object (chose type we don't know (Object class reference variable, child class object)
↳ This is old fashioned way to implement "Generic Programming"

Important Object class methods

(1) protected Object clone() throws CloneNotSupportedException

→ Creates & return copy of this object.

→ By default : shallow copy

→ class overiding clone() method must implement Cloneable interface to tell JVM.

→ Object class does not implement cloneable, calling clone() method Object class will lead to Exception.

(2) protected void finalize() throws Throwable

→ called by Garbage Collector for objects that are not referenced by any one

(3) public boolean equals (Object obj)

→ Indicates whether other object is equal to this one.

→ By default (object class) : compares on addresses!

→ Overriding can be done on some/all properties equality.

Properties :→

(1) Reflexive : a equals a

(2) Symmetric : a equals b (⇐) b equals a

(3) Transitive : a equals b & b equals c ⇒ a equals c

(4) Consistent : a equals b remains same if a & b remain same

(5) a.equals(null) = false

```java
class Movie {
    String name;
    int duration;
    double rating;

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

    @Override
    public boolean equals(Object other) {
        if (this == other)
            return true;
        return this.name.equals(((Movie) other).name);
    }
}
```

without overriding

true
false
false

default equals logic

with overriding

true
false
true

custom equals logic

```java
Movie m1 = new Movie(name: "Endgame", duration: 180, rating: 4.9);
Movie m2 = m1;
System.out.println(m1.equals(m2)); // Object's are same: true

Movie m3 = new Movie(name: "Infinity War", duration: 150, rating: 4.7);
System.out.println(m1.equals(m3));

Movie m4 = new Movie(name: "Endgame", duration: 200, rating: 5.0);
System.out.println(m1.equals(m4));
```

```java
System.out.println(m1.equals(m1)); // Reflexive
System.out.println(m2.equals(m1)); // Symmetric
System.out.println(m2.equals(m4)); // Transitive
System.out.println(m1.equals(other: null)); // False
```

④ public final Class getClass()

→ Can't be overridden

→ Returns runtime class of an object. "Class" object of this object can be used to get metadata of this class

Eg code

```
Object obj1 = new Object();
Class cobj1 = obj1.getClass();
System.out.println("Object 1 : " + obj1 + " ClassName: " + cobj1.getName());

Object obj2 = new String(original: "Hello World");
Class cobj2 = obj2.getClass();
System.out.println("Object 2 : " + obj2 + " ClassName: " + cobj2.getName());

Object obj3 = new Movie();
Class cobj3 = obj3.getClass();
System.out.println("Object 2 : " + obj3 + " ClassName: " + cobj3.getName());
```

Output

```
Finished in 98 ms
Object 1 : java.lang.Object@511d50c0 ClassName: java.lang.Object
Object 2 : Hello World ClassName: java.lang.String
Object 2 : Movie@1d44bcfa ClassName: Movie
```

(5) public int hashcode()

→ Returns a hash code value for the object.

hash code → Unique integer value which is used to uniquely identify the object.

→ it does not store/return the address of object

It is used to search objects in collections such as HashSet, HashMap, etc.

Default Behavior :- Every different Object (different memory address) will get unique hashcode.

→ Hashcode for a particular object will remain same forever, ie there cannot be two hashcodes for same object during a single run of application.

Note :- i) If two objects are equal (acc to equals()) method, then they must give same hashcode values.

ii) If two objects are unequal, good hash function should give different hash codes for those objects.

⑥ public String toString()

{ return getClass.getName() + "@" + Integer.toHexString(hashCode()); }

→ Returns String Representation of object

Default Behavior: classname @ unsigned hexadecimal hashcode

Other methods related to Synchronization :→

- public final void notify()
- public final void notifyAll()

} → lock released/thread out of critical section

- public final void wait()
- public final void wait(long timeout)
- public final void wait(long timeout, int nanos)

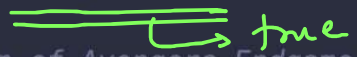
} → lock acquired/thread going in critical section!

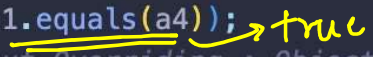
```java
class Movie implements Cloneable {
    String name = "Avengers Endgame";
    int duration = 180;
    double ratings = 4.5;

    public Movie() {
        System.out.println(x: "Object Created - Initialization by Constructor");
    }

    public Movie(String name, int duration, double ratings) {
        this.name = name;
        this.duration = duration;
        this.ratings = ratings;
    }

    @Override
    public boolean equals(Object other) {
        return this.name.equals(((Movie) other).name);
    }

    @Override
    protected Object clone() throws CloneNotSupportedException {
        return super.clone();
    }

    @Override
    public String toString() {
        return ("Name : " + this.name + " , Duration : "
                + this.duration + " , Ratings : " + this.ratings);
    }

    @Override
    protected void finalize() throws Throwable {
        // Deallocation of Resources Holded by Object
        // (Database Connection, File Input/Output Streams, H/W Resources like Camera)
        // Similar to Destructors in C++
        // Automatically Called By Garbage Collector Daemon Thread handled by JVM
        System.out.println(x: "Object Destroyed - Resources Clean Up By Finalize");
    }

    @Override
    public int hashCode() {
        // Custom Hashing Logic
        return this.name.hashCode();
    }
}
```

```java
@SuppressWarnings("rawtypes")
public static void getClassDemo() {
    Object obj1 = new Object();
    Class cobj1 = obj1.getClass();
    System.out.println("Object 1 : " + obj1 + " ClassName: " + cobj1.getName());

    Object obj2 = new String(original: "Hello World");
    Class cobj2 = obj2.getClass();
    System.out.println("Object 2 : " + obj2 + " ClassName: " + cobj2.getName());

    Object obj3 = new Solution();
    Class cobj3 = obj3.getClass();
    System.out.println("Object 3 : " + obj3 + " ClassName: " + cobj3.getName());
}
```

```
architaggarwal@Archits-MacBook-Air 01. Core Java Basics % java Solution
Object 1 : java.lang.Object@136432db ClassName: java.lang.Object
Object 2 : Hello World ClassName: java.lang.String
Object 3 : Solution@7382f612 ClassName: Solution
```

```java
public static void equalsDemo() {
    Movie a1 = new Movie(name: "Avengers Endgame", duration: 180, ratings: 4.9);

    // Different Movies
    Movie a2 = new Movie(name: "Avengers InfinityWar", duration: 150, ratings: 4.8);
    System.out.println(a1.equals(a2)); // False

    // Exactly Same Movie (Addresses/References are Same)
    Movie a3 = a1;
    System.out.println(a1.equals(a3)); // True (Address Comparison First)

    // Extended Version of Avengers Endgame
    Movie a4 = new Movie(name: "Avengers Endgame", duration: 200, ratings: 5.0);
    System.out.println(a1.equals(a4));
    // By Default (Without Overriding : Object's equals) : False
    // With Overriding and Name Comparison (Movie's equals) : True
}
```





```java
public static void cloneDemo() throws Exception {
    Movie a1 = new Movie(name: "Avengers Endgame", duration: 180, ratings: 4.9);
    System.out.println(a1);

    Movie a2 = (Movie) (a1.clone());
    System.out.println(a2);
}
```

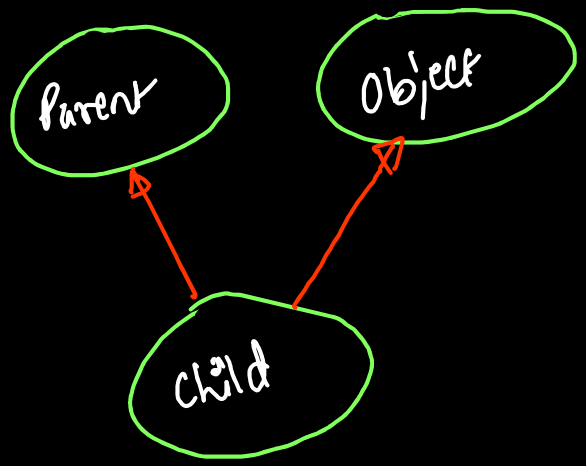
```
architaggarwal@Archits-MacBook-Air 01. Core Java Basics % java Solution
Name : Avengers Endgame , Duration : 180 , Ratings : 4.9
Name : Avengers Endgame , Duration : 180 , Ratings : 4.9
```

Je tostring methods

```java
public static void hashCodeDemo() {
    Movie a1 = new Movie(name: "Avengers Endgame", duration: 180, ratings: 4.9);
    System.out.println(a1.hashCode());

    Movie a2 = new Movie(name: "Avengers InfinityWar", duration: 150, ratings: 4.8);
    System.out.println(a2.hashCode());

    Movie a3 = a1;
    System.out.println(a3.hashCode());

    Movie a4 = new Movie(name: "Avengers Endgame", duration: 200, ratings: 5.0);
    System.out.println(a4.hashCode());
}
```

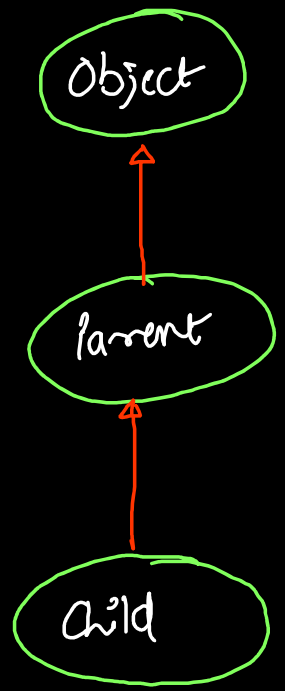
```
architaggarwal@Archits-MacBook-Air 01. Core Java Basics % java Solution
-2084956566
1213999165
-2084956566
-2084956566
```

Q) If Object class is a parent class for Every class, so it means there might be multiple inheritance possible?



Multiple Inheritance(✗)

Hybrid Inheritance(✗)



multilevel inheritance!