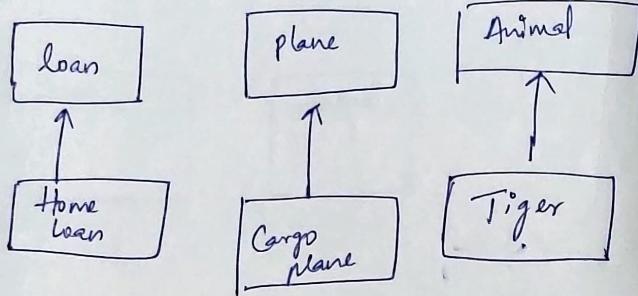


## (25) Inheritance, (Inherited vs Specialised Methods)

### Super, final Keyword in Java

#### ① Inheritance Example



→ loan → Personal loan  
Education loan  
Car loan  
Home loan } Types of loans

→ Common data regarding loans are written in loan class.  
→ this common data inherited in all types of loans.

## (2) Access Specifiers in Java

→ the access modifiers in Java specifies the accessibility (or) scope of a field, Method, Constructor, or class.

- we can change the access level of fields, Constructors, Methods, and class by applying the access modifier on it.
- there are four types of java access Modifiers

① Private - the access level of a private modifier is only within the class. It cannot be accessed from outside the class.

② Default - the access level of and default modifier is only within the package.  
→ it cannot be accessed from outside the package.

→ if you do not specify any access level, it will be the default.



GADDALA ASHISH  
LinkedIn

③ Protected :- the access level of a protected modifier is within the package and ~~the~~ outside through child class.

→ if you do not make child class, it cannot be accessed from outside the package.

④ Public - the access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

⇒ Access Specifiers - a type of access specifiers

- ① Public
- ② Protected
- ③ Default
- ④ Private

} Access Specifiers applied to class, Method, Constructor, Variable.

Example ① :- Default modifier

Void disp()

{

y

→ if we not specify any access level it will be the default.

② Public void disp()

{ → public access specifier

}

③ Private int age;

↳ Restrict access within class

⇒ Project → Multiple functionality

Folder {

- ↳ Addition → multiple classes
- ↳ Subtraction → Multiple classes and Methods
- ↳ Multiplication →

→ these folders we call it as package.

→ Java project

↳ combination of packages.  
↳ Modularity



GADDALA ASHISH

in LinkedIn

### ③ Understanding Java Access Modifiers

	Within a class	Outside class Within Package	Outside P ackage (is-A relationship)	Outside Package no is-A relation
Public	✓	✓	✓	✓
Protected	✓	✓	✓	✗
default	✓	✓	✗	✗
Private	✓	✗	✗	✗

↓  
Outside  
Package by  
Subclasses only

✓ → can be accessed

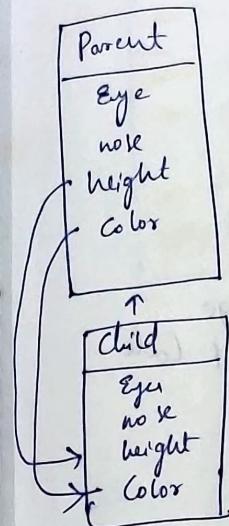
✗ → not accessed

### ④ Access Modifier Visibility:

- ① Public
- ② Protected
- ③ Default
- ④ Private

↑  
Visibility  
Increasing.

→ Public → strongest access specifier  
→ Private → weak access specifier.



→ features inherited from  
Parents  
~~→~~

↳ We cannot modify some  
inherited features.

⑤ Inherited Methods - Methods which are inherited from parent to child. In child class we are using as it is without any modification.

⑥ Overriding Methods - Methods which are inherited from parent to child. In child class as per the requirement it is modified then these methods are called overriding methods.

⑦ Specialized Methods - A method which are present only in child not present in parent are called specialized methods.

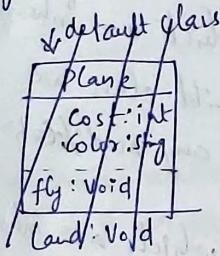


GADDALA ASHISH  
in LinkedIn

→ Remember these symbols

- ① '+' → Public
- ② '#' → Protected
- ③ '~' → default
- ④ '\_' → Private

→ UML = Unified Modeling language  
↳ pictorial representation of what code you have written.



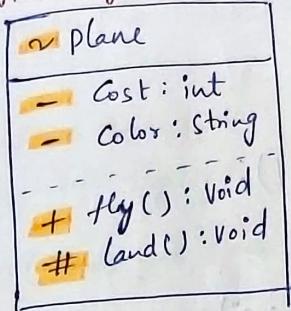
⑦ UML (Unified Modeling language) :- is a general purpose, graphical modeling language in the field of software engineering.

→ UML is used to specify, visualize, construct and document the artifacts (Major Elements) of the software system.

→ ~~class~~

UML diagram

Eg:-



⇒

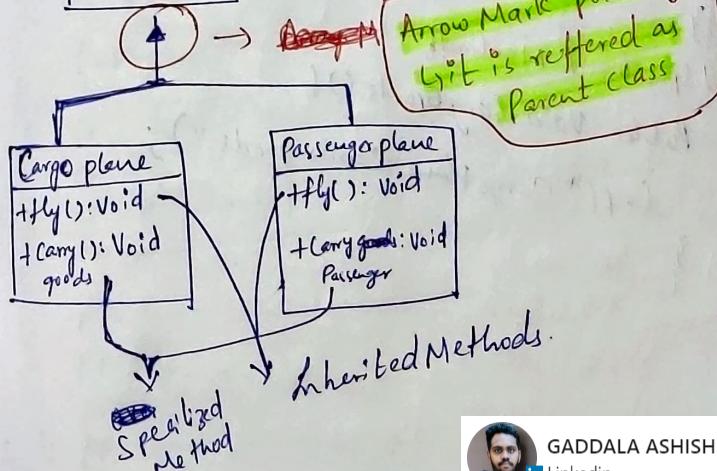
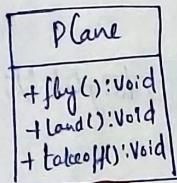
class plane

```
private int Cost;
private string Color;
public void fly();
```

protected void land();

Code

⑧



GADDALA ASHISH  
In LinkedIn

Code :-

Class plane

```

    Public Void takeoff(){
        s.o.p("plane taking off");
    }

    Public Void fly(){
        s.o.p("plane is flying");
    }

    Public Void landing(){
        s.o.p("plane is landing");
    }
}

```

Class CargoPlane Extends Plane

```

    // overridden Method
    Public Void fly(){
        s.o.p("CargoPlane flies at lower height");
    }
}

```

// Specialized Methods

```

    Public Void carry Goods(){
        s.o.p("CargoPlane carries goods");
    }
}

```

Class PassengerPlane Extends Plane

```

    {
        // Overridden Method
        Public Void fly(){
            s.o.p("Passenger plane flies at Medium height");
        }

        // Specialized Method
        Public Void carry Passengers(){
            s.o.p("Passenger plane carries passengers");
        }
    }
}

```

~~Public class Inherit {~~

```

    Public static void main(String[] args){
        CargoPlane cp = new CargoPlane();
        PassengerPlane pp = new PassengerPlane();
    }
}

```

cp.takeoff(); // Inherited Method

cp.fly(); // Overridden Method

cp.carry Goods(); // Specialized Method

cp.landing(); // Inherited Method

pp.takeoff(); // Inherited Method

pp.fly(); // Overridden Method

pp.carry Passenger(); // Specialized Method

pp.landing(); // Inherited Method



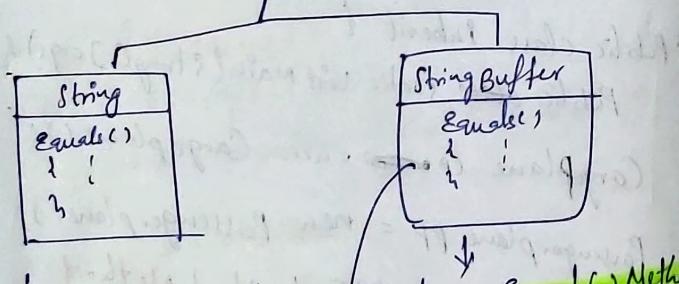
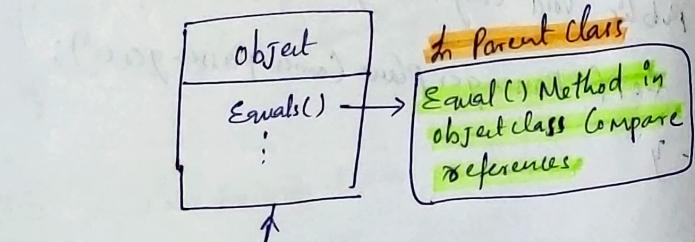
GADDALA ASHISH



LinkedIn

## ⑧ Important Points

- Object class is parent of all classes in Java.
- Equals(), toString() etc are present in object class.
- Inbuilt classes are inheriting from object class (Parent class).

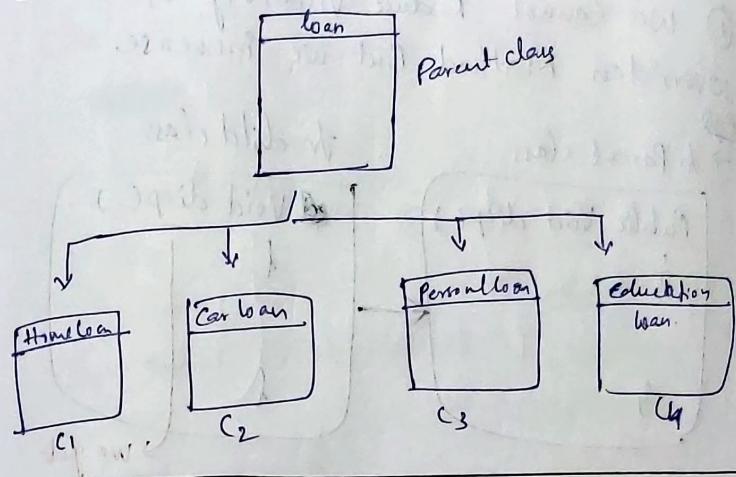


④ here Equals() Method used to compare object Content

⑤ As per the need Equals() Method is modified.

→ Equals() Method in String and StringBuffer acts different.

## Home Work



→ Parent → Child → inherited → overridden → specialized.

⑥ Rules to override method in

Base class



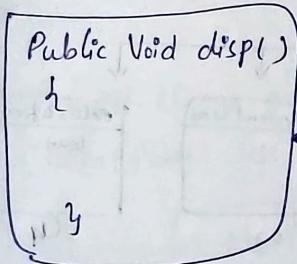
GADDALA ASHISH  
in LinkedIn

## ⑨ Rules to override Method :-

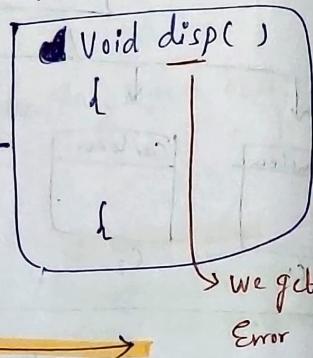
① We Cannot Reduce Visibility of Overridden Method But we increase.

Eg:-

→ In Parent class



In child class



→ Method Visibility is Reduced.

## ② Code (1) We Cannot Reduce Visibility of Overridden Method.

Class Parent {

    Public Void disp()

```
    { System.out.println("Parent class");}
```

}

Class child Extends Parent

{ → here it is default.

    Void disp()

{

    → we get Error.

Method visibility cannot be Reduced

Y Y

## Code (2) We Can Increase Visibility of overridden Method

Class Parent { → here it is default

    Void disp()

{

    System.out.println("Parent class");

Y

Y

Class child Extends Parent

{

    Public Void disp()

{

    System.out.println("Child class");

}

→ here  
Visibility of  
Overridden  
Method is  
Increasing

→ Many inbuilt Methods are public.

↳ In order to override Method we have  
increase it's visibility.



GADDALA ASHISH  
 LinkedIn

Rule ② > Return-type of overridden Method  
Must be same as that of overriding  
Method in parent.

Code :-

```
Class Parent {  
    Public int add() {  
        S.o.p("Parent");  
        return 10+2;  
    }  
}
```

→ here return type  
of add Method  
is int.

class child Extends Parent

// overridden Method

→ // here return type of add Method is void

```
Public void add() {
```

{

```
S.o.p("child");
```

}

Return type  
cannot be  
changed  
it should be  
same as return  
type of overriding  
Method in  
parent class



GADDALA ASHISH

in LinkedIn

Rule-③ → Return type of overridden Method in child class can be different as that of Parent class if it is Co-Variant return-type (return-type is -A relationship in Parent-child relationship).

Code → check this code in github (Inherit 13)

Class Interest {

y Class Interest PersonalLoan Extends Interest

{

y

Class Loans {

// overriding Method

// here return type is Interest  
Public Interest interest() → Return type

{  
Interest it = new Interest();  
return it;

y Class PersonalLoan Extends Loans

{ // overridden Method

Public Interest PersonalLoan interest() → Return type

{  
Interest PersonalLoan ipL = new Interest PersonalLoan();  
return ipL;

→ here is -A relationship (in Parent-child relationship) is there. That's the reason different return type is allowed for overriding and overridden Method.

→ We call it as Co-Variant return-type.

→ In return type there is Parent child relationship then return types can be different.

Rule-④ → Parameters of overridden Method must be same as that of Parent else it will be considered as Specialized Method Considering Method overloading.

Code → Class Parent {

Public int add(int a, int b)

{  
return a+b;

y Class Child Extends Parent  
{

Public int add (int a)

{  
return a;

y

→ here Name of Methods are same but different parameters

→ it is called Method overloading

→ This Method is called as Specialized Method

→ check this code in github (Inherit 14)



GADDALA ASHISH

LinkedIn

## → Super( ) Method

↳ if is written inside Constructor.

↳ it will call parent class constructor.

→ Super - Key word.

## 10) Super keyword :-

the Super Keyword in Java is a reference Variable which is used to refer immediate Parent class object.

### Usage of Java Super Keyword

① Super can be used to refer immediate Parent class instance Variable.

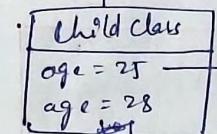
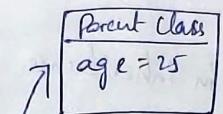
② Super can be invoke immediate Parent class Method.

Code:-

```
Class Parent {
    int age = 25;
}
class Child Extends parent {
    int age = 28;
    void disp() {
        System.out.println(age); → // 28
        System.out.println(super.age); → // 25
    }
}
```

```
public class inherit {
    public static void main(String[] args) {
        Child c = new Child();
        c.disp
    }
}
```

→ here Super keyword referred to immediate Parent class instance Variable.



↳ S.O.P(age) → 28

↳ S.O.P(super.age) → 25

## 11) final keyword :-

the final keyword in Java is used to restrict the Java user. The final can be

① Variable

② Method

③ class

class keyword → to Create class.

new keyword → to Create object.

→ final keyword applied for Class, Method and Variable.



GADDALA ASHISH  
Linkedin

## Q(i) Java final class :

- if you make any class as final, you cannot extend it.
- final class doesn't participate in inheritance

Code :-

```
final class Vehicle
```

```
{
```

```
    void disp()
```

```
{
```

```
    System.out.println("Vehicle");
```

```
}
```

```
Class Car Extends Vehicle
```

```
{
```

```
}
```

```
public class Inherit {
```

```
    public static void main(String[] args) {
```

```
        Car c = new Car();
```

```
        c.disp();
```

```
}
```

```
}
```

↳ we get Error  
 ↳ final class doesn't participate in inheritance

## (ii) Java final Method

- ① if you make any Method as final, you cannot override it.
- ② final Method participated in inheritance.

Code :-

```
class Vehicle {
```

```
    final void disp() {
```

```
        System.out.println("Vehicle");
```

```
}
```

```
Class Car Extends Vehicle
```

```
{
```

Void disp() → we get Error (final method get inherited but we cannot override)

```
{
```

```
    System.out.println("Car");
```

```
}
```

```
}
```

```
public class Inherit {
```

```
    public static void main(String[] args) {
```

```
        Car c = new Car();
```

```
        c.disp();
```

```
}
```

```
}
```



GADDALA ASHISH  
in LinkedIn

### (iii) Java final Variable

→ If you make any variable as final, you cannot change the value of final variable.  
(it will be constant).

Code:-

Class ~~Parent~~ Parent {

    final int i=10;

    Void disp(){}

        i=20; →

        System.out.println(i); // output: 10

        System.out.println("Parent");

}

y

Class child Extends Parent

{

y

Public class inherit

    Public static Void main(String[] args)

{

    Car c = new Car();

    c.disp();

y

~~final~~

### Java final keyword

- ① Stop Value change
- ② Stop Method overriding
- ③ Stop inheritance.

here i will be  
10 not 20



GADDALA ASHISH  
[Linkedin](#)