INHERITANCE
LECTURE - 13

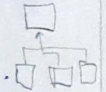
* There is single inheritance, multiple inheritance, they can also be multilevel.

"IS-A"

* Using Inheritance, one can create a general class that include some set of item.

* The inherited class (child) can be used to create more specific classes which has all the items/members from the base class, in addition to some items of its own.

→ SUPERCLASS: A class that is inherited.
→ SUBCLASS: A class that does inheriting.



↳ special version of superclass which contains additional / specific content.

◉ It inherits all of the instance variables & methods defined by the superclass & its own, unique elements

* It helps in reusability of the code.

"EXTENDS" ↳ Keywords to inherit a SUPERCLASS

ie) class <subclass> extends <superclass> {

}

* Function can be overridden.

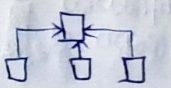
↳ only if same data type, function value, access modifier which is not more restrictive.

Eg: 2D → 3D

TYPES OF INHERITANCE

→ Single Inheritance.

→ Multiple single Inheritance



→ Multilevel single Inheritance.

✗ Multiple Inheritance.
✗ Hybrid Inheritance

NOT POSSIBLE IN JAVA.

METHOD OVERRIDING

SUPER & SUB

* The method that is defined in both class, then the method is overloaded in subclass.

→ It's a runtime polymorphism. ! overriding...

The methods must have same name, parameters, return type ELSE overloaded.

There must be a "IS-A" relationship.

```
Subclass x = (Subclass) Superclass;
2D            2D        3D
```

↑ typecast. ie) SUBCLASS IS A SUPERCLASS

WRT, Subclass ⟶ Can access ⟵ Can't access ⟶ Super class

SUPER KEYWORD

→ It's a keyword.

→ In JAVA its a reference variable that is used to "refer Immediate parent class members."

Whenever you create an instance of sub class, an instance of its parent class is created ~~explicitly~~ implicitly, which is referred by super keyword.

USES:

→ refer immediate parent class instance variable ✓

→ invoke immediate parent class method. ✓

→ referencing immediate parent class constructor. Invoking ✓

```
class Animal {
    Color: white
}
class Dog extends Animal {
    color: black
    print(color)
    print(super.colour)
}
```

! To access parent class property we use super keyword

#TECHNE - 13
INHERITANCE

```
class Animal {
    eat(){ print(1) }
class Dog {
    eat(){ print(2)
        super.eat()
        eat()
    }
```

2
1

* super is used to invoke immediate parent class constructor.

```
class Animal {
    Animal(){ print(1)}
}
class Dog {
    Dog(){
        super()
        print(2)
    }
}
```

The super keyword, can be used to INVOKE the ~~overrid~~ OVERLOADED parent constructor, if

arguments are there, then they should be specified accordingly.

THEN, you have to define the super constructor matching with each constructor.

If there is a no. of overloading constructors in superclass,

DYNAMIC METHOD DISPATCH

* It is also called RUNTIME POLYMORPHISM.
* It is a process in which a call to an overriden method is resolved @ runtime.

↳ In this process, an overriden method is called throughout the reference variable of a super class. The determination of the method to be called is based on the object being referred to by reference variable.

This is POLYMORPHISM IG..

Reference type can be a superclass of the actual object.

Bike b3 = new Splendor(); // up casting.

run()

So if we have to store array of diff types we can declare the arr as a supertype & elements as subclass which are inherited.

ABSTRACT CLASS IN JAVA

* Abstraction is a process of hiding the implementation detail & showing only functionality to user.
* It helps us focus "what object DOES?" INSTEAD "How it does?"

→ Abstract - keyword.

* Abstract method is empty {}
* This class can have abstract & non-abstract method.
  → It can't be instantiated.
  It can have constructors, static methods also.
* It can have final (keyword) method which will force the sub class NOT to change body of method.
* So we can't use / call abstract method directly but use it by overriding / overloading.

FINAL KEYWORD

* It tells that this final class can't be used for inheritance.
* Restrict access of an item from its super class to a sub class.
  ↳ can be implemented variable, method, class
  ↓
  It means no more implementation in any derived class is possible.

ie) final class <classname> ;

5

DEMONSTRATION - 6

#LECTURE - 14

```
Subclass-Constructor {
  super()   // calls the default constructor of the superclass.
}

Subclass-constructor {
  super(a,b,c)   // calls the overloaded constructor in the super class.
}
```

input since new

* A superclass variable can be referring to a subclass object.

Subclass reference	= superclass reference.	✗
Superclass reference	= subclass reference.	✓

super.i ← Via from superclass.

avoids "Name space collision"

- Abstract class can't derive itself to its own object, it shall be used by extending, inheriting and overloading methods.

??? Check ⇒ If we have 2 classes, 1 inherits other, then both constructors are implemented in inheritance.

} same? CODE

A constructor of a superclass can be called by Subclass constructor only.

check 68, 69

Use of "final" → Using STRICT, PROTECTING, RESTRICTING ACCESS.

```
abstract class c {
    final method
}
class a extend c {}
```

We can access them But can't override.

INFORMATION HIDING
#LECTURE-15

* ACCESS MODIFIERS IN JAVA
  → Public → Protected → default → Private
* They "specify accessibility (scope)" of a data member, method, constructor or class.

Access modifiers

default → Visible to "same package"
Public → Visible "anywhere"
Protected → Visible to "INHERITED CLASS"
Private → Visible to the class only.

◎ If NOT specified ⇒ set default.
When that member will be accessible to any class that belongs to same directory, where the class belongs or it is basically belong to same package.

"SCOPE OF DATA MEMBER"

Modifier \ Access levels	CLASS	PACKAGES	SUB-CLASS	EVERYWHERE
Public	✓	✓	✓	✓
Protected	✓	✓	✓ + To inherited classes...	×
default	✓	✓	×	×
Private	✓	×	×	×

DEFAULT

→ default is set by default.
→ It is accessible only within Package.

* Package is a set of files, that belong to a common directory.

◎ if 2 classes are default lie in the same directory, then they can be accessed & (VICE-VERSA) NO COMPILE TIME ERROR.

PUBLIC

→ Public has the widest scope among these 4 modifiers.
→ It is accessible everywhere.
also in any other dir.

```
Package P1;
Public class A{
    void msg(){
    }
}
```

```
import P1.*;
Package P2;
class B{
    Public static void main(){
    }
}
```

Public doesn't matter whether it belongs to same directory or program file.

If same file, set it default than public.
else we will get COMPILATION ERROR.

GOOD PRACTISE ⇒ to save all files in single class.

(!) when a class is public all its members with default access specifiers are also public.

PRIVATE

→ Accessible only within class
→ Can't be accessed outside the class.

(!) when a class is private all its members within class are also private.

CONSTRUCTOR + PRIVATE ⇒ If u make any Constructor private we can't create instance of that class outside that class.

↖ NOT PROVIDED NOTED

PROTECTED

It is accessible within package or from outside package only via INHERITANCE.

→ It can't be applied on a "CLASS"

Diff sub-dir ⇒ Package → than extend to use private modifiers.

* If you are "overriding" it should NOT be more restrictive than that method

* overloading

$X = 2+3 = 5$

$Y = 5+3 = 8$

$X = 11$

$Y =$