2. Inheritance, Interface & Packages

- Inheritance

The mechanism of deriving new class from old one is called inheritance

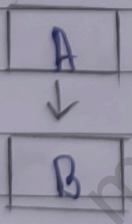
new one class is called as child class, derived class & sub class

old one class is called as parent class, base class & super class.

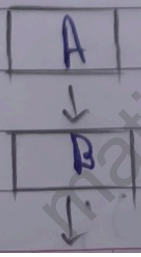
- Types of Inheritance

1) Single inheritance
2) Multilevel inheritance
3) Multiple inheritance
4) Hierarchical inheritance

1) Single inheritance

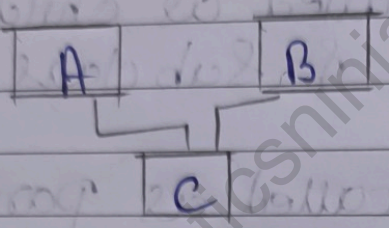


2) Multilevel inheritance



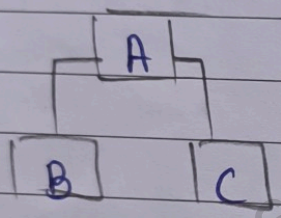
3] Multiple inheritance

Multiple inheritance is not in Java. New concept is used to implement Multiple inheritance that is interface.



This dicision was made to avoid ambiguity problems like a diamond problem

4] Hierarchical inheritance



Extends keyword is used here

44]] Single inheritance

```
Class A
{
void display1()
{
System.out.println("Hello");
}
}
Class B extends A
{
void display2()
{
System.out.println("Java");
}
}
```

```
Class demo {
Public static void main (String args[])
{
    B b1 = new B();
    b1.display1();
    b1.display2();
}
}
```

o/p : Hello
Java

2] Multilevel inheritance

```
Class A
{
Void display1()
{
System.out.println("Hello");
}
}
Class B extends A
{
Void display2()
{
 System.out.println("Java");
}
}
Class C extends B
{
Void display3()
{
```

```
System.out.println("programming students");
}
}
class demo
{
public static void main(String args[])
{
C c1 = new C();
c1.display1();
c1.display2();
c1.display3();
}
}
```

o/p : Hello
Java
programming students.

The mechanism of deriving class from another derived class is known as multilevel inheritance

2) Hierarchical inheritance
When two or more sub classes are derived from single super class then inheritance called as Hierarchical inheritance.

46

```
class A
{
void display1()
{
System.out.println("Hello");
}
}

class B extends A
{
void display2()
{
System.out.println("Java");
}
}

class C extends A
{
void display3()
{
System.out.println("Python");
}
}

class Simple1
public static void main(String args[])
{
B b1 = new B();
b1.display1();
b1.display2();
C c1 = new C();
c1.display1();
c1.display2();
}}
```

O/p:

Hello J
Java
Hello
Python

* Final keyword

Keyword final has uses like
① to prevent overriding
② to prevent inheritance

Final Variable
By declaring Variable with final keyword that Variable can not be inherited
the value of final variable never be change

Final method
Final method can not be override

```
final void show()
{

}
```

We can prevent overiding using final.

4] class A
{
final void math()
{
System.out.println("this is final method");
}
}
class B extends A
{
void math()
{
System.out.println("this method cannot be overridden");
}
}

The math() is declared as a final It can not be overide in B. Any Attempt made so it will generate compile time error.

Final class

A class caul not be Subclass is called as final class

Final keyword prevent classes from inheritance

Ex:

```
final class A
{
}
```

- Abstract method and classes

Abstract keyword is used to declare class abstract.

Abstract classes has no object an abstract class can not be directly instantiated by new operator

Syntax:

```
abstract class class_name
{
    --- 
}
```

Ex:

```
abstract class Shape
{
abstract void draw();
}
```

When class contain one or more abstract method it should be declared abstract.

Although abstract classes can not be used to instantiate objects they can be used to create object references so it is necessary to create reference to an abstract class

While using abstract class must satisfy following condition

① we can not use abstract class to instantiate object directly.

Shape s = new shape ();
is illegal because Shape is an abstract class

Abstract method
The Abstract method of an abstract class must be define in its sub class

Super keyword:

Super keyword is used to refered immediate parent class.
Sub class needs to refer its immediate super class with keyword super

1] Super keyword is used to first refered first to invoke super class constructor to sub class constructor

2] To invoke immediate parent class method

- Accessing super class member
  super.member

5] Ex:

```
class super_class
{
public void printmethod()
{
System.out.println("method in superclass");
}
}
public class Subclass extends Superclass
{
public void printmethod()
{
   super.printmethod();
```

```
System. out. println ("printed in Subclass");
}
public static void main (String args[])
{
 subclass s = new Subclass ();
 s. print method();
}
}
```

- There is Sub class called subclass that overrides prints method ()

o/p
 prented in Super class
 preinted in Subclass

In this program print method is inherited from Super class

- Multiple inheritance not available in Java

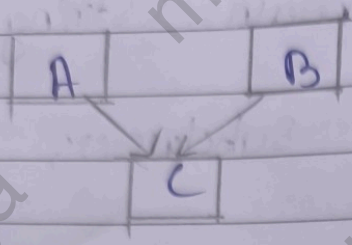
Grand parent

parent 1   parent2

child2

ambiguity

Java can not have more than one Base classes



Java classes can not have more than one super class But in most of application Multiple inheritance is required

In above fig child have inherits the properties of grand parent to separate part

all public & protected member of grand parent and inherited in to child twice

Means child would have dupplicate set of members. this introduce ambiguity and should be avoided

This problem also called as diamond problem

Java provides alternative solution for this approach known as interface

Interface is also kind of class interface contain or define only abstract method and final fields

Syntax:

Access interface interface_name.

Variable of interface are initialized with constant values.

All the variables are implicitaly Public if interface it is declare as a public.

method declaration contains only list of methods without any body and ends with semicolon.

Ex:

```
Interface printable
{
Void print();
}
Class Interfaceexample implements
                    printable
{
Public Void print()
{
System.out.print("hello");
}
```

```
public static void main (String args[])
{
Interface example obj = new interfaceexample
 obj. print();
}
}
```

o/p : Hello

Final class example

7]

```
Final class A
{
Void display()
{
system.out.println (" Base class method").
}
}

class B extends A
{
Void display()
{
super. display();
System. out. println ("Derived class
                      method");
}
}
```

o/p: Cannot inherit from final A

8] Interface Area
{
final static Float pi = 3.14f;
Float compute (Float x, Float y)
}

Class Rectangle implements Area
{
public Float compute (Float x, Float y)
{
return (x * y);
}
}

Class circle implements Area // another interface implementation
{
public Float compute (Float x, Float y)
{
return (pi * x * y);
}
}

Class Simple {
{
Pubjic Static void main (String args[])
{
Rectangle rect = new Rectangle (); // create object of class which implemt interface
Circle cir = new Circle ();
Area area;
Area = rect; // interface obj create.
area refers to the rect object.

```
System.out.println(" area of Rectangle " +
                        area.compute(10,20));

area = cir; // area refers to circle object.
System.out.println(" area of circle :" +
                        area.compute(10,2));
}
}
```

output:

```
Area of rectangle : 200
Area of circle : 0.0
```

- Nesting of interfaces

Interface can be nested similar to class an interface can be nested within class or another interface also.

An interface can be declared inside class body or interface is called as nesting of interfaces

Rules:

① Nested interface must be public if it is declare inside interface

② Nested interface can have any access modifier if declare inside class.

Syntax:

```
interface interface_name
{
  Interface nestedinterface_name
  {
  }
}
```

1. Package:

Reusability can be achived through inheri and interfaces by extending the classes and implementing interface but this is limited to reusing the class with in program

if we want to use classes of other program without copying them into program then this can be achived in java by using packages

Packages can be used for grouping the variety of classes and interfaces together.

Advantages of pakages

1] Reusability
2] class can be unique

packages are acts as a container for classes

There are two types of pacakges
1] API or
Library pacakges
2] User defined pacakage

1] API or library packages:

① LANG / lang
② Util
③ io
④ net
⑤ applet
⑥ awt
⑦ Swing

Suppose class Color is awt package is referred then statement can be used Java.awt.Color;

import package_name.class name OR

import package_name.*;

1] Java.lang
This is the language support package/classes

2] Java.util
This is the language utility classes such as Scanner, vector etc.

3] Java.io
This is input output support classes.

4] Java.awt
Contents the set of classes for implement GUI.

5] Java.net
It contain the classes for networking

6] Java.applet
Creating an implement applets

2] User defined package

Syntax:
Package Package_name;

Ex:
Package Pack1;

Package is a keyword and package name is Pack1

Any no. of classes and interface can define under package declaration consider as a part of package

Syntax:
```
package pack1
{
  class A
  {
    ==
    ==
  }
}
```

```
Class B
{
   =
}

Interface C
 {
   =
 }
```

Accessing members of package

By using import keyword a package created in one program can be imported in other program.

Ex:

```
Package pack1;
Public class Arithmetic operations {
Public int getaddition ( int x, int y)
{
return (x+y);
}

Public int getsubtraction ( int x, int y)
{
  return (x-y);
}
}
```

```
package pack1;
import pack1.*;
```

10.

A.Java,

```
Package eny;
Public class A
{
Public void
   show();
{
System.out.println("A");
}
}
```

B.Java

```
import eny.*;
{
Public class B
  {
Public static void main (string args[])
  {
  A a = new A();
  a.show();
  }
}
```

O/P:

D → ABC → eng → A.text file →
B.text file

SSS > d:
Dir cd ABC
cd eng
set classpath =
Javac A.java.
eng > cd..
Javac B.java
Java B

o/p → A