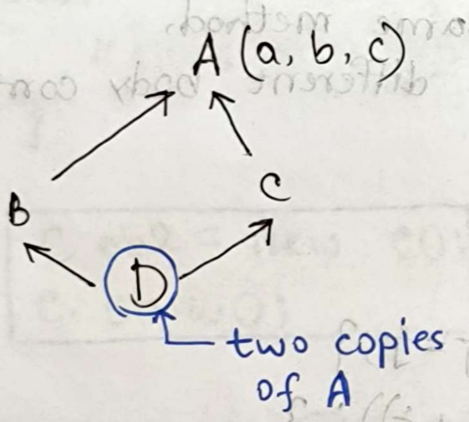
Advanced Programming
CSE341
Class-0 | 19-01-2026

#Inheritance (Recap)

C++ Inheritance
- Single Inheritance
- Multi-level Inheritance

↓

Subclass inherits from multiple base-classes



This ambiguity is resolved using 'virtual' base class

Java supports multi-level inheritance, not multiple inheritance.

☑ Multi-level
☒ Multiple
↳ I mean kind of supported but not directly.

Multi-Level

Level-wise

$$\begin{tikzcd} A \\ B \arrow[u] \\ C \arrow[u] \end{tikzcd}$$

```
public class Main{
 public static void main(String[] args){
    C ob1 = new C();
 }
}
```

```
class A{
   A(){ —("A class"); }
   }
class B extends A{
   B(){ —("B class"); }
   }
class C extends B{
   C(){—("C class"); }
   }
```

II] Method Overriding

$$\begin{tikzcd}
 & \text{Polymorphism} \arrow[dl, no head] \arrow[dr, no head] & \\
\text{Complie time} \arrow[d] & & \text{Runtime} \arrow[d] \\
\text{Overloading} & & \text{Overriding}
\end{tikzcd}$$

↳ In this case, type signature must be the same

↳ Override should happen within the ~~subjec~~ sub class.

↳ Requires inheritance

↳ Same method, different body contents

```
class A {
    int i,j;
    A(int i, int j){ this.i=i; this.j=j;}
    void show(){ —("i&j" +(i+j)); }
}
class B extends A {
    int K;
    B(int i,int j, int K){
        super(i, j);
        this.K =K;
    }
    void show(){ —(" sum is : "+(i+j+K)); }
}
```

```
public class Main{
    public static void main(String[] args){
        B ob1 = new B(1,2,3);
        B. show();
    }
}
```

if parameters are different, it will be overloaded, NOT overridden

```
class C extends A{
    // DOES NOT HAVE 'show()' method
}
```

```
C ob2 = new C();
C. show();
```

parameter type, count and return type must be same to be a method to be overridden.

Runtime Polymorphism

```
C ob1 = new C();
```

→ এ side না থাকলে object create হবে না

creating a Reference variable

Allocating a memory space for an object

এ side এ শুধু Reference variable create হয়।

"Super class এর reference variable Subclass এর object এর point করতে পারে"

```
class flower{
    String name;
    flower (String name){this.name = name;}
    void show(){— "Name: "+name;}
}

class rose~~{~~ extends flower{
    rose(String name){ super(name);}
    void show (){— (name);}
}

public class Main{
   public static void main(String[] args){
     flower ob1 = new flower("rose");
     rose ob2 = new rose("china rose");
     flower ob3;
            reference variable
       ↳ flower ob3= Ob2;
```

"Super এর ref var যে type er obj কে point করবে সে overridden method version টা call করবে"

↖ "Dynamic Method Dispatch" বলে

```
flower ob3 = ob2;
   Ob3. show(); ← calling method from Ob2
      ob3 = ob1;
   ob3. show(); ← calling method from Ob1
```

Abstract class / Pure Virtual Function

Abstract Class

Abstract Method / Concrete Method

↳ 'abstract' Keyword

$$\left[\text{abstract type name(——);}\right]$$

কোনো class এক বা একাধিক abstract method contain করলে সেটা abstract class.

Abstract classes / methods are incomplete

↳ Cannot create an object of an abstract class

→ abstract classes can also contain concrete methods

→ Sub class would be forced to override the abstract methods from the abstract class

→ We can't create Objects of abstract classes but can create reference variables.

abstract classes can have constructors.

```
class flower{
        String name;
        flower (String name){this.name=name;}
        abstract void show();
        void show1() {—"Only for show.");}
}

class rose extends flower {
    rose(String name){super(name);}
    void show () {—("name"); }

}
                rose Ob2=new rose ("china rose");
```

```
Ob3=ob2;
Ob3.show();
Ob3. show1();
```

```
flower Ob3;
```

```
Ob3 = ob2;
Ob3 . show();
Ob2. show1();
```

Scenario দেওয়া থাকবে

abstract method can be a blueprint.

```
abstract class Area{
    double d1, d2;
    Area (double d1, double d2){
        this.d1 = d1;
        this.d2 = d2;
    };
    abstract void calculate();
}
class Rectangle{
    Rectangle(double d1, double d2){
        super (d1, d2);
    }
    void calculate(){
        System.out.println(d1 * d2);
    }
}
class Triangle{
    Triangle(double d1, double d2){
        super(d1, d2);
    }
    void calculate (){
        System.out.println(0.5 * d1 * d2);
    }
}
```

```
public class Main{
  public static void main(String[] args){
     Area Ob1;
     Rectangle ob2= new Rectangle(12, 24);
     Triangle ob3 =new Triangle (3,4);

     Ob1 =ob2;
     ob1.calculate();

     Ob1 =ob3;
     ob1.calculate();

  }
}
```