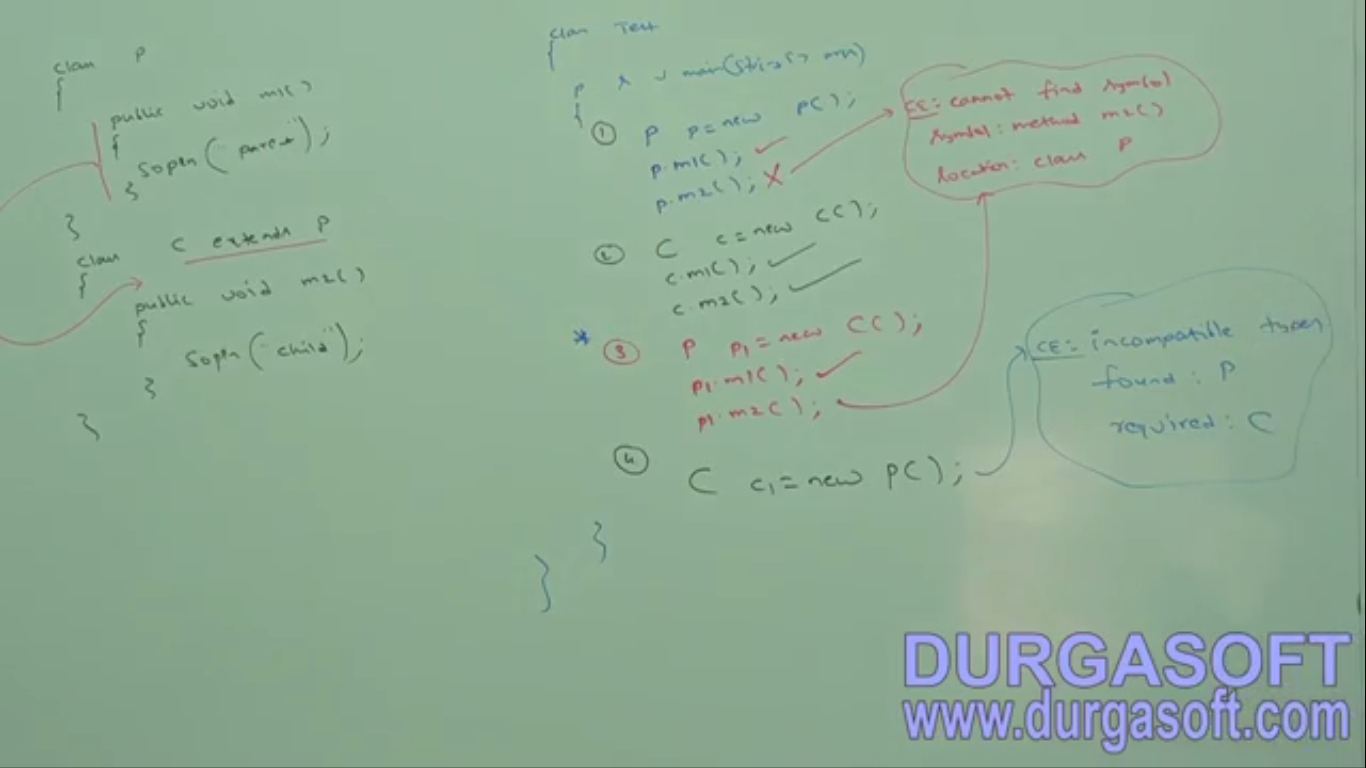
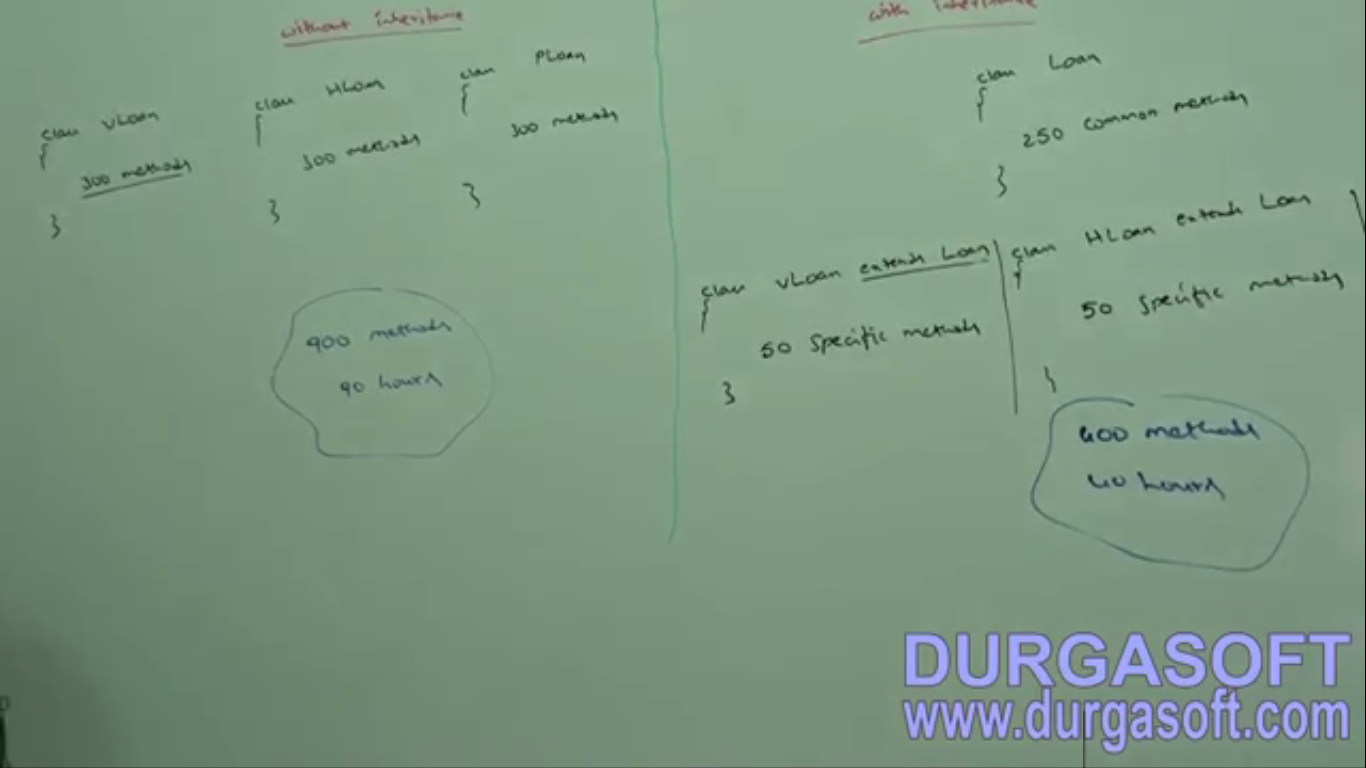
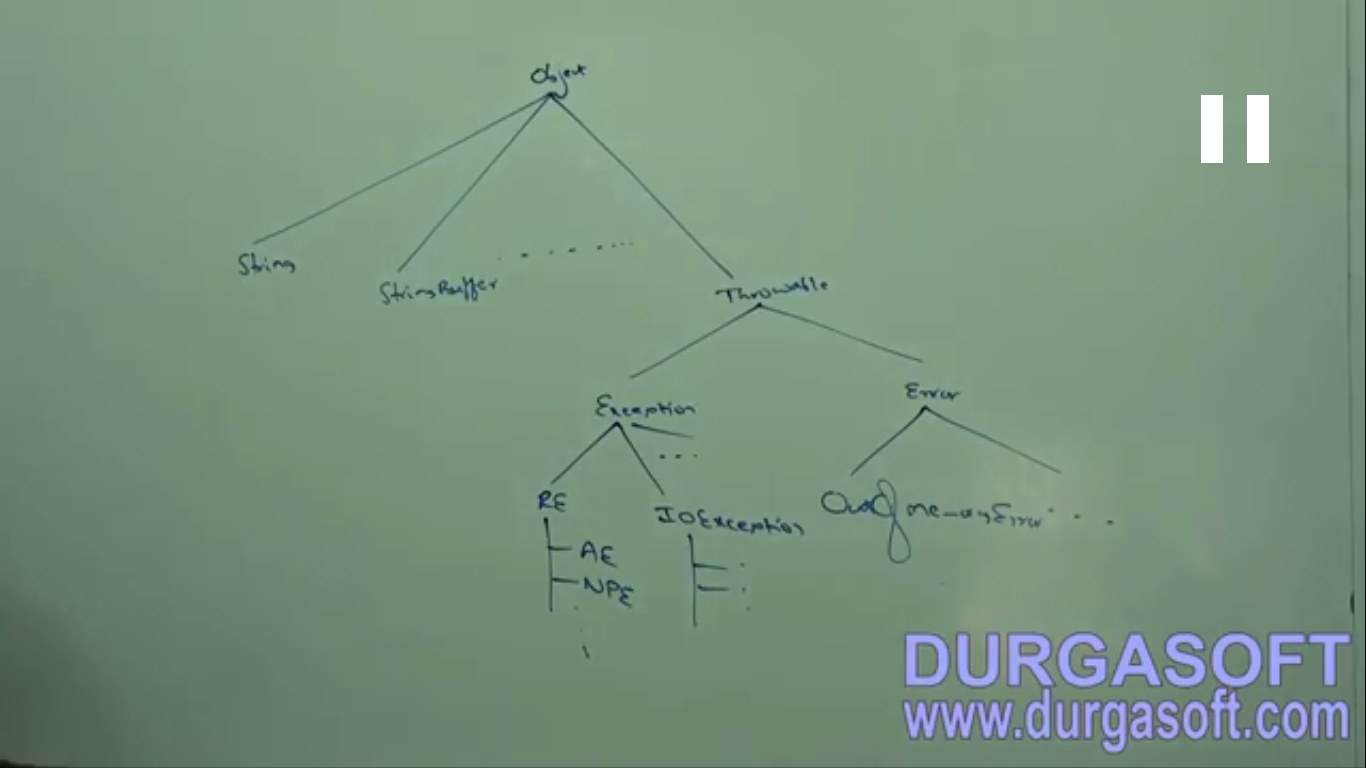
is-a relationship

1. **Also known** as inheritance.
2. Implemented via **extends**
3. **Advantage**: Main advantage of is-a relationship is “**code reusability**”.
4. **There are 4 cases in case of inheritance when playing with parent-child reference variable and parent-child methods using those reference variables.**
5. 

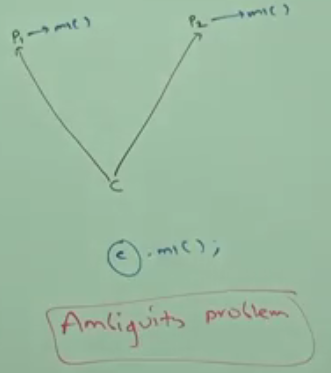
1. Parent class doesn’t have access to child class members  
2. Child class has access to parent members.  
3. Parent Reference variable holding child object can access only parent part of child object.   
4. Child reference type can’t hold reference to parent object.

1. **The following example shows the code without inheritance and with inheritance showing the reusability of code** **NOTE**: The most common methods which are applicable for **any type of** child, we have to define in parent class. The specific methods which are applicable for a particular type, we have to define in child class.
2. Total Java API has been implemented based on inheritance.   
   Object class is the root class which has many child classes  
   String, StringBuffer, Throwable which is parent for Exception and Error and So On.   
   **Proof**: 12 methods which are applicable to any object have been defined in Object class.   
   The above is the reason too for Object class to be declared as the Top most root class as it contains 12 methods which are applicable to any type.   
   **Throwable**: Having those methods which are applicable for Exception and Error simultaneously. That is why, it is the root class for Java Exception Hierarchy.

Total Java API is implemented based on inheritance.

1. **NOTE** : Java doesn’t provide support for multiple inheritance.
2. **NOTE:** If our class doesn’t extend any other class, only then our class is direct child of Object class. If our class extends some other class, then our class indirect child class of Object class.

**Jatin NOTE**: We can explicitly inherit from Object class.   


1. **Why does java not provide support for multiple inheritance?**
   1. 
2. NOTE: Java provides support for multiple inheritance with respect to interfaces.   
   

**Muliple inheritance is not allowed due to chance of getting different implementations from different parent classes as those methods having same method signature. But then why multiple inheritance is allowed in case of interface where one interface extends more than one interface?**

Both interfaces’ m1() has same method signature, above that, same return type too.

Interface01 having m1()

Interface01 having m1()

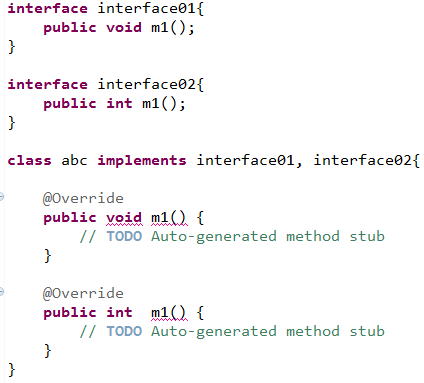
**implementation** of interface03

interface03 will get m1() from interface01 and interface02

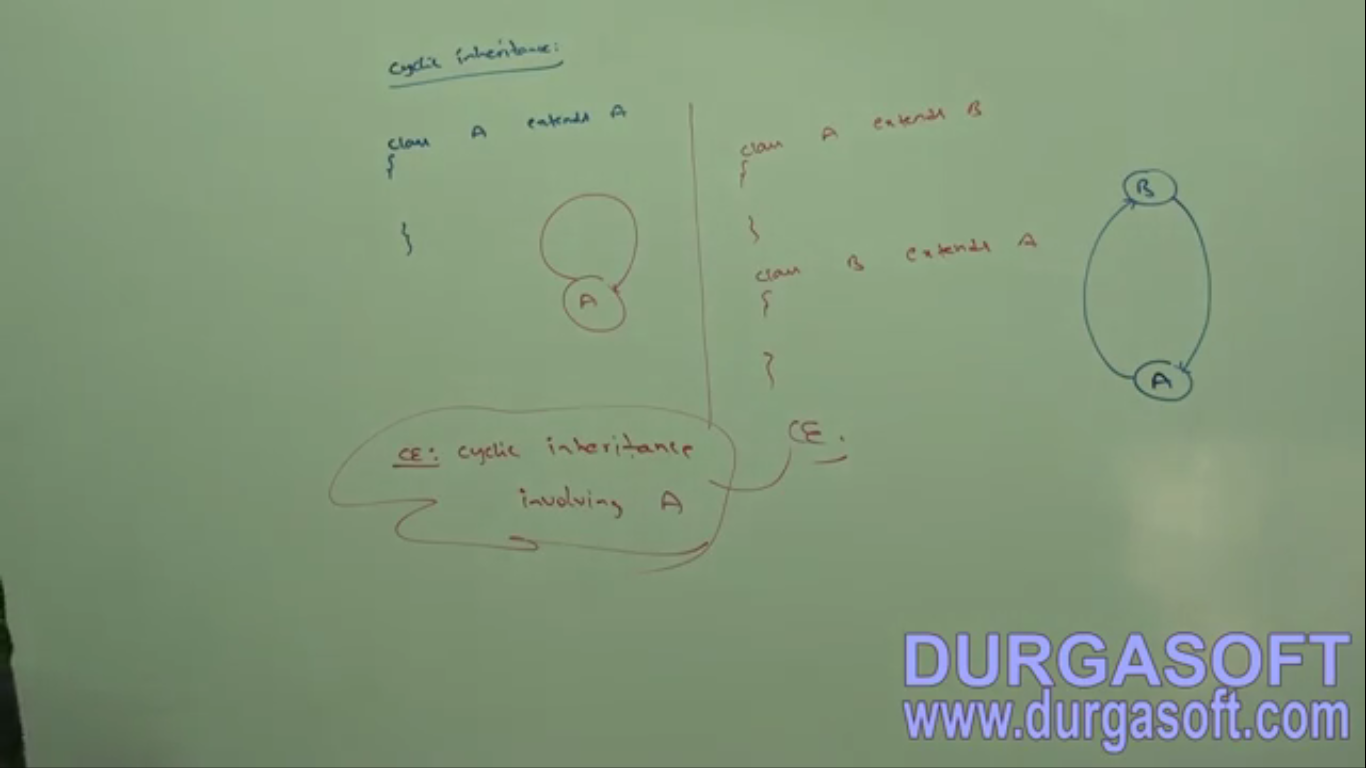
Even though multiple method declarations are available, but implementation is unique and hence there is no chance of ambiguity problem when a class implementing multiple interfaces simultaneously.

Strictly speaking, through interfaces we don’t get any inheritance because inheritance means code reusability, inheritance means borrowing implementation but when one interface inherits other interface, it doesn’t borrows any implementation but just declaration.

Jatin🡺 If both interfaces contain a method say m1() with same signature but different return type and a class abc implementing both interfaces, then?



In the above case, we have to implement m1() from both interfaces but their return type is different but signature is same. So we can’t implement both methods simultaneously 🡺 so we can’t implement both interfaces simultaneously.

1. **Cyclic Inheritance**: Cyclic Inheritance is now allowed in java. Of course, it’s not required in actuality. 
2. d
3. d