***Interface***

1. *Any service requirement specification is considered an Interface.*

*Eg:- (1) JDBC API acts as requirement specification to develop Database driver.*

*🡪 Database vendor is responsible to implement this JDBC API.*

*(Oracle Driver, MySQL Driver)*

*(2) Servlet API acts as requirement specification to develop WebServer.*

*🡪 WebServer vendor is responsible to implement Servlet API.*

*(ApacheTomcat,IBM Websphere)*

1. *From client point of view an interface defines a set of services what he is expecting.*

*From ServiceProvider point of view an interface defines the set of services what he is offering.Hence an contract between client and service-provider is considered as Interface.*

1. *Inside interface every method is abstract by default , thus interface is considered as pure abstract class.*
2. ***Any service requirement specification or any contract between client and service provider or pure abstract class is called as Interface***

***Declaration Rules:***

* *Whenever we are implementing an interface for each and every method of that interface ,we need to provide implementation. Otherwise we should declare class as abstract. Then next level child class is responsibe to provide implementation.*
* *Every interface* ***method is always public and abstract*** *whether we are declaring or not. Whenever we are implementing an interface method compulsory we should declare as public. Otherwise we will get compile time error.*

***Extends vs Implements:-***

1. *A class can extend only one class at a time.*
2. *Interface can extend any no of interfaces simultaneously.*
3. *A class can implement any no of interfaces simultaneously.*
4. *Class can extend another class and can implement any no of interfaces simultaneously.*

*Class A extends B implements C,D,E*

***(Interface Methods)***

***Every method of interface is by default:***

* *public, abstract*

***We cannot declare interface with these modifiers:***

* *private, protected,static, final, synchronized, strictfp, native*

***(Interface Variables)***

1. *Main purpose of interface variable is to define requirement level constants.*
2. *Every interface variable is always* ***public static final*** *, whether we are declaring or not.*

***Public->*** *To make this variable available to every implementation class irrespective of the package.*

***Static ->*** *It is static because we should be able to use that variable without creating an object. As we cannot create objects of interface.*

***Final->****If one implementation class changes value then remaining implementation classes will be affected. To restrict this every interface variable is always final.*

1. *Compulsory we should perform initialization or else we will get Compile Error*

***( Interface naming Conflicts )***

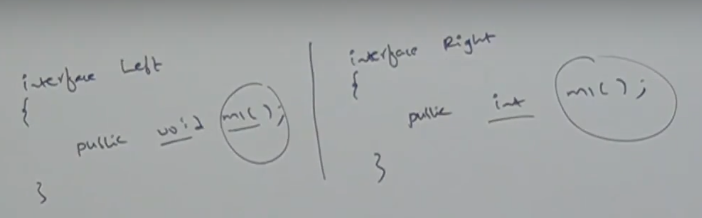
***(Method naming conflicts):-***

***Case 01:*** *If two interfaces contain method with same signature and return type then in implementation class we need to provide implementation for only one method.*

***Case 02:*** *If two interfaces contains method with same name but different argument types then in the implementatio class we need to provide implementation for both methods and these methods act as overloaded methods.*

***Case 03:*** *If two methods contain method with same signature but different return types then it is impossible to implement both interfaces simultaneously.(If return types are not co-variant)*

*Java class an impement any no of interfaces except above case.*



***(Variable naming conflicts):-***

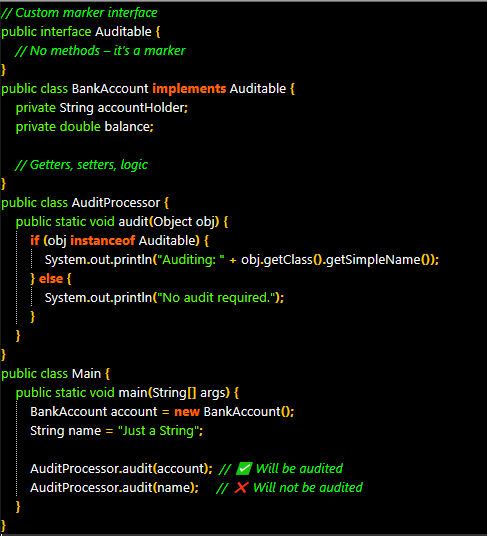
* + *Two interfaces can contain a variable with same name and there maybe a chance of variable naming conflicts. But we can solve this problem by using interface names.*

*Left.x // Right.x*

*(****Marker / Tag Interface )***

1. *If a interface doesn’t contain any methods and by implementing that interface if our objects get any ability such interfaces are called Marker Interfaces.*
2. *Eg:- Serializable, Cloneable, RandomAccess . These are marked as some ability and internally JVM is responsible for providing implementation for these Interfaces.*

***Is it possible to create own / custom Marker / Tag interface? 🡪 Yes***



***Adapter Class:-***

* *It is a simple java class that implements an interface with only empty implementation.*
* *If we implement an interface, for each and every method of that interface then compulsory we should provide implementation whether it is required or not required.*
* *The problem in this approach is it increases length of te code and reduces readability.*
* *We can solve this problem using adapted classes.*
* *Instead of implementing interface , if we extend adapted class we have to provide implementation only for required methods and we are not responsible to provide implementation for each and every method of interface. So that length of code will be reduced.*

***Interface vs Abstract class vs Concrete Class:-***

* *If we don’t know anything about implementation just we have requirement specification then we should go for interface. (Servlet)*
* *If we are talking about implementation but not completely (Partial implementation) then we should use abstract class.(GenericServlet, HttpServlet)*
* *If we are talking about implementation and ready to provide service then we should use Concrete class.(MyOwnServlet)*

|  |  |
| --- | --- |
| *Interface* | *Abstract Class* |
| *If we don’t know anything about implementation and just have requirement specification then use Interface.* | *If we are talking about partial implementation then we should go for abstract class* |
| *Every method is always public , abstract. Hence 100 % pure abstraction is achieved.* | *Every method inside abstract class need not be public and abstract , and can have concrete methods also.* |
| *Private , protected, final, static , syncrhonized, native modifiers for methods are not allowed.* | *No restrictions on abstract class method modifiers.* |
| *Every variable present inside interface is always public, static, final* | *Every variable present in abstract class need not be public, static, final* |
| *Private, protected, volatile, transient modifiers for variables are not allowed* | *No restrictions on abstract class variable modifiers* |
| *Initialization is compulsory while declaration only.* | *Initialization is not compulsory while declaration.* |
| *We cannot declare static and instance blocks* | *We can declare static and instance blocks.* |
| *Interface cannot have constructors.* | *Abstract classes can have constructors.* |

***Note:-***

*1)We cannot create object of abtsract class and interface.*

*2) Anyways we cant create object for interface and abstract class but abstract class can have constructor whereas interface doesn’t have. The main purpose of constructor is to perform initialization of instance variables. Abstract class can have instance variables which are required for child object.*

*3) To perform initialization of those instance variables, constructors is required for abstract class. But every variable present inside interface should compulsorily initilaized while declaration itself. Hence constructor is not required for interfaces.*

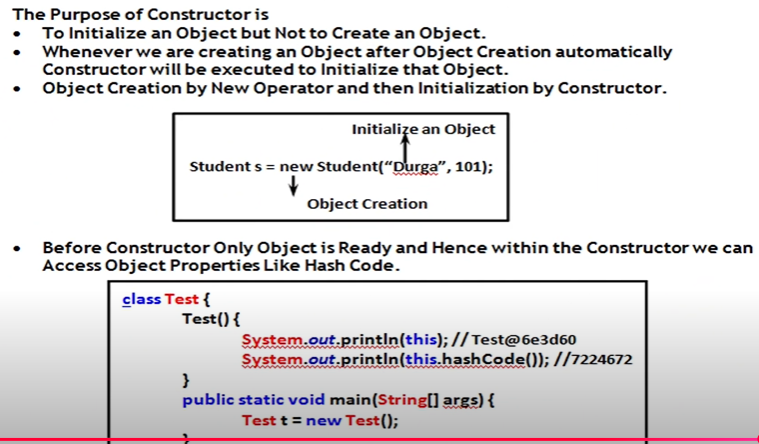
*4) While extending abstract class it is not possible to exten any other class and hence we are missing inheritance benefit.*

*5) Whie implementing interface we can extend any other class , hence we wont miss any inheritance benefit.*

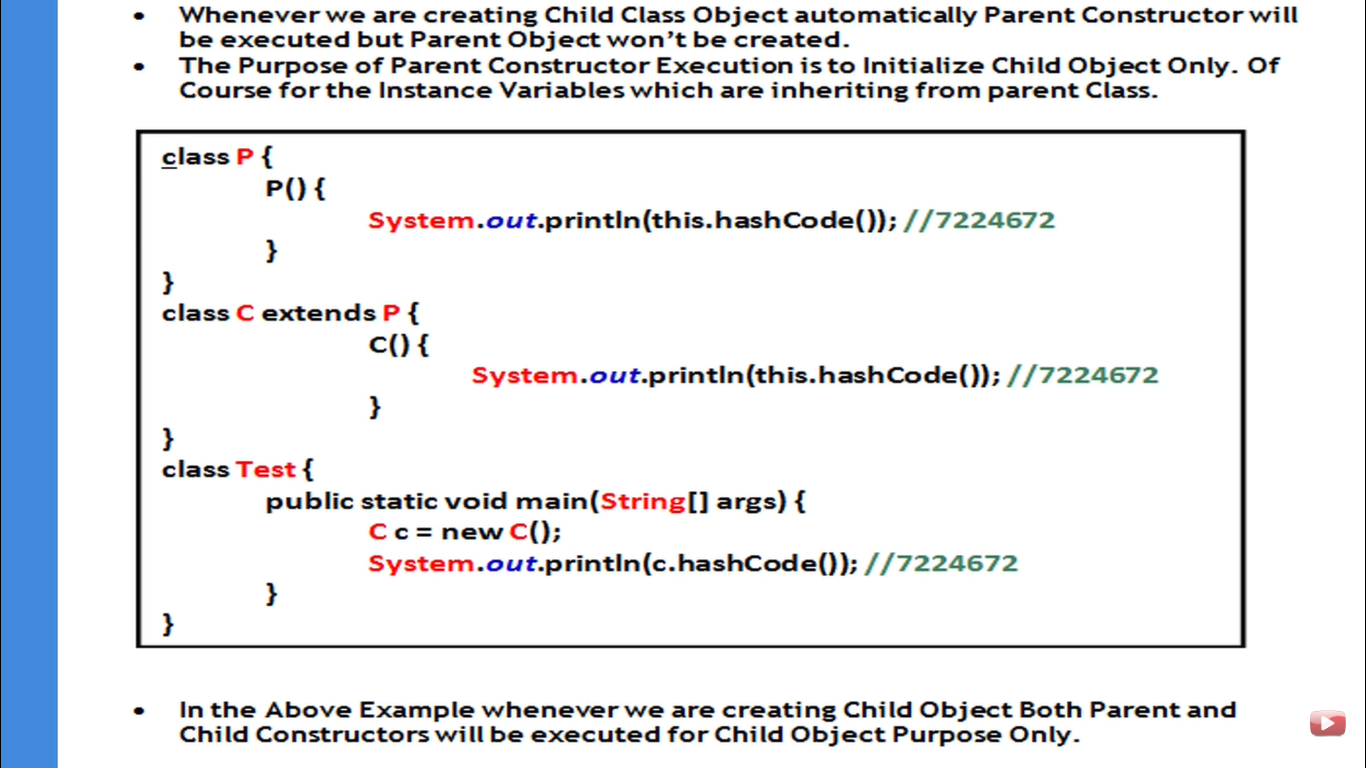
***Interface vs Abstract Class QnA***

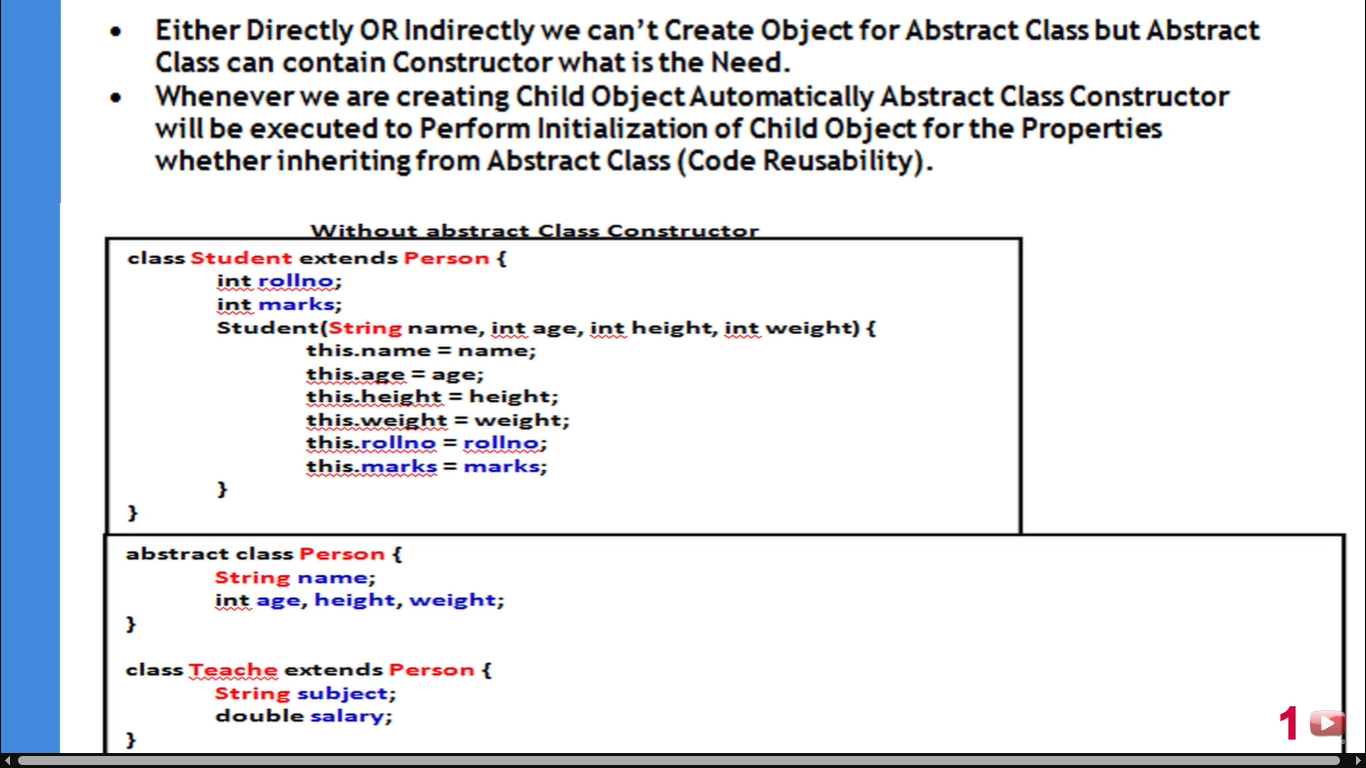
1. ***We cannot create object of abstract class, but abstract class can contain constructor. What is the need?***

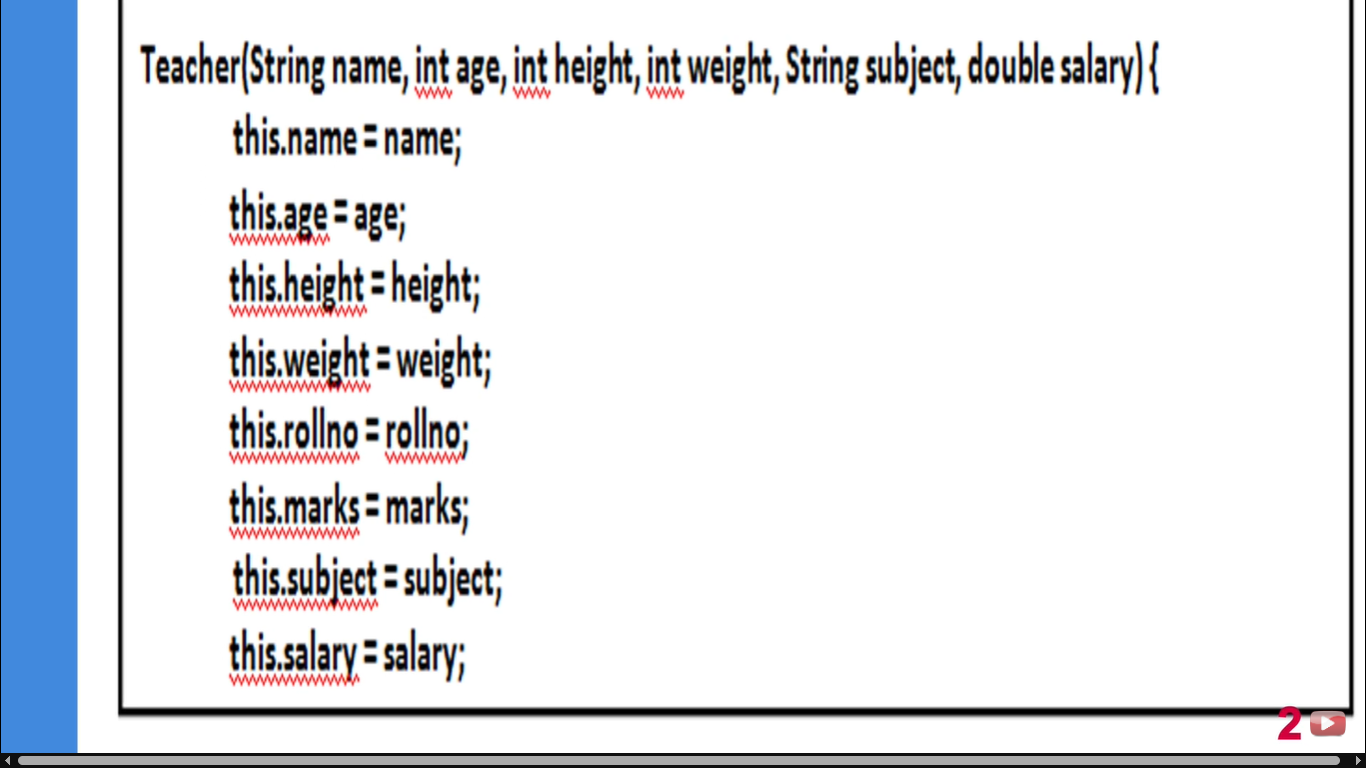
* *Main objective of “ new “ operator is to create an object.*
* *Main purpose of constructor is to initialize that object*
* *First object will be created by “new” then initialization will happen by “constructor”*

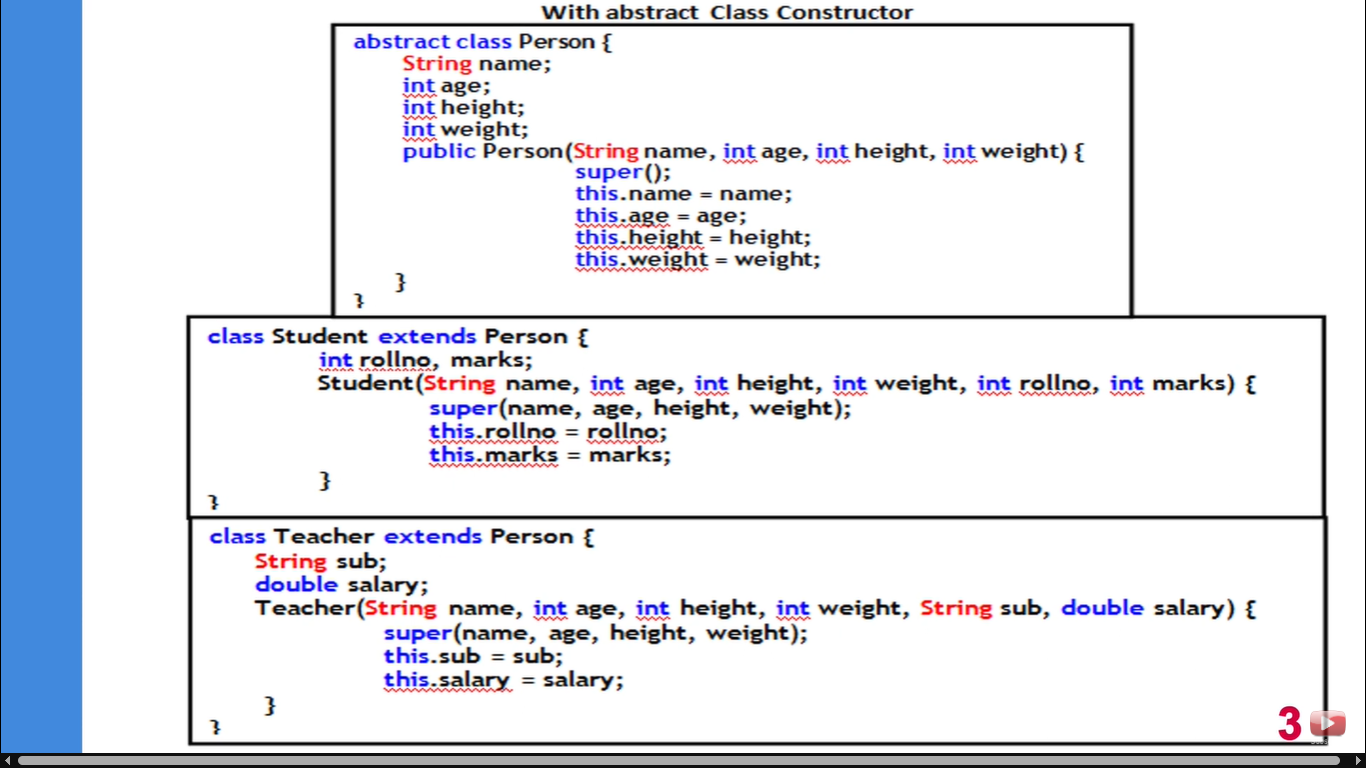


1. ***We cannot create object of abstract class directly but indirectly we can create it . Valid?***

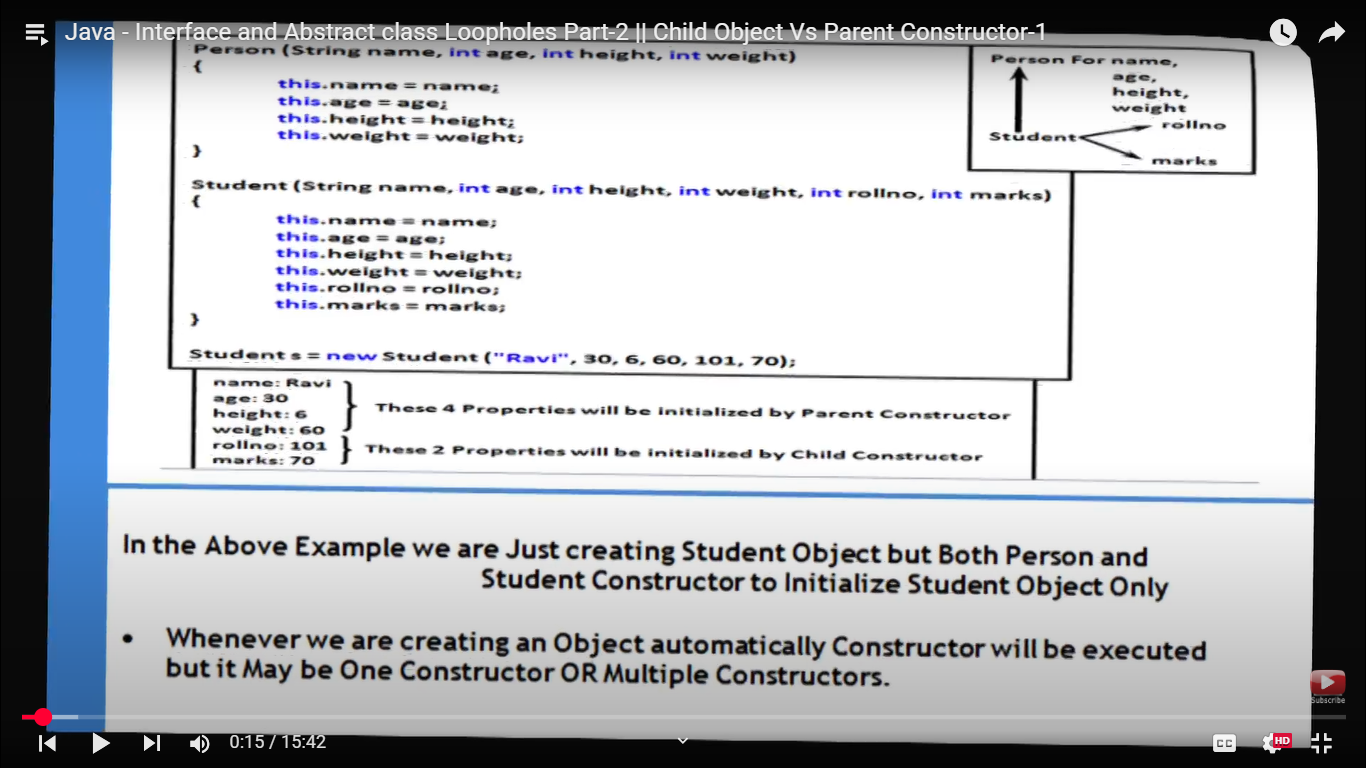




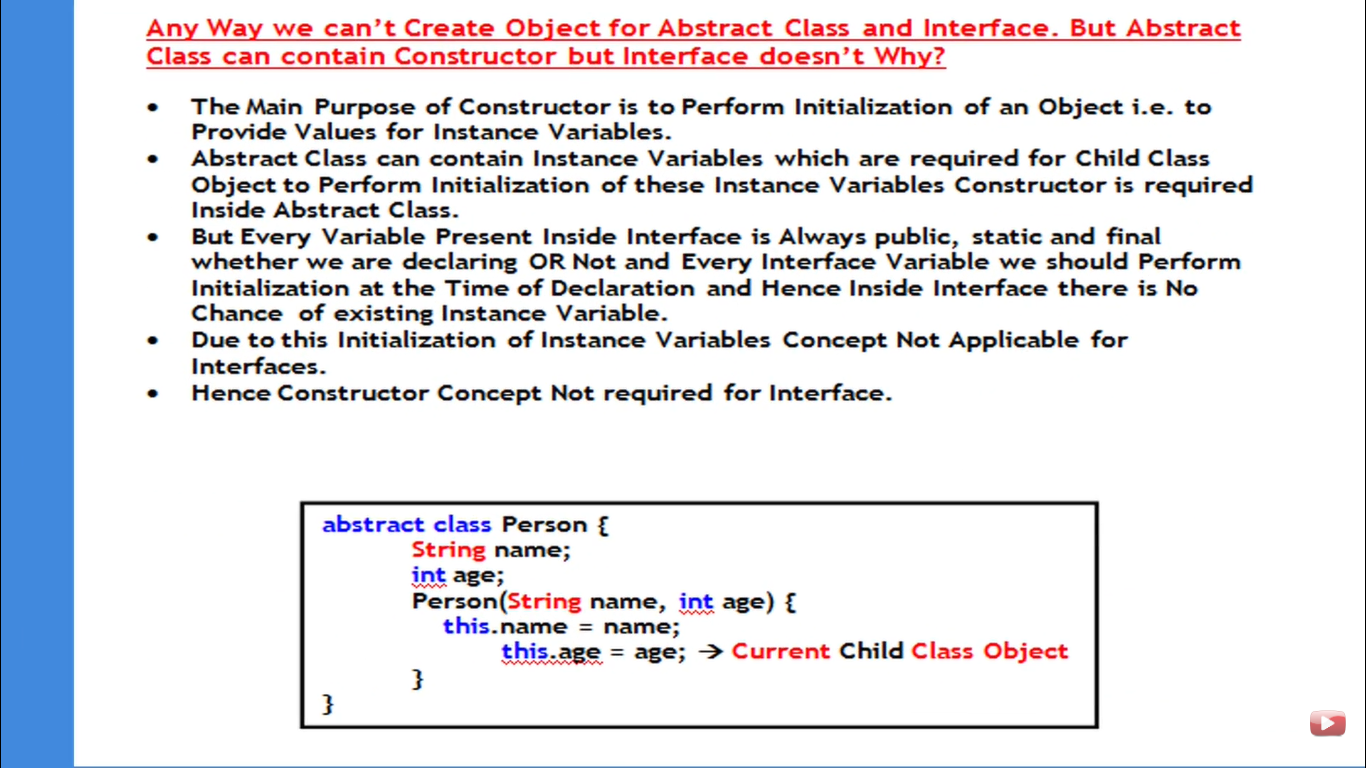




1. *Whenever we are creating child class object automatically parent class object will be created.Valid / Invalid****? Valid🡪 Every first line of Constructor is super() whose job is to invoke parent class constructor.***



1. ***Anyway we cannot create object for abstract class and interface. But abstract class can have constructor but interface does not . Why?***



1. *Inside interface we can take only abstract Methods. But in abstract class also we can take only abstract methods based on our requirement. Then what is the need of interface. Is it possible to replace interface cncept with abstract class?*

