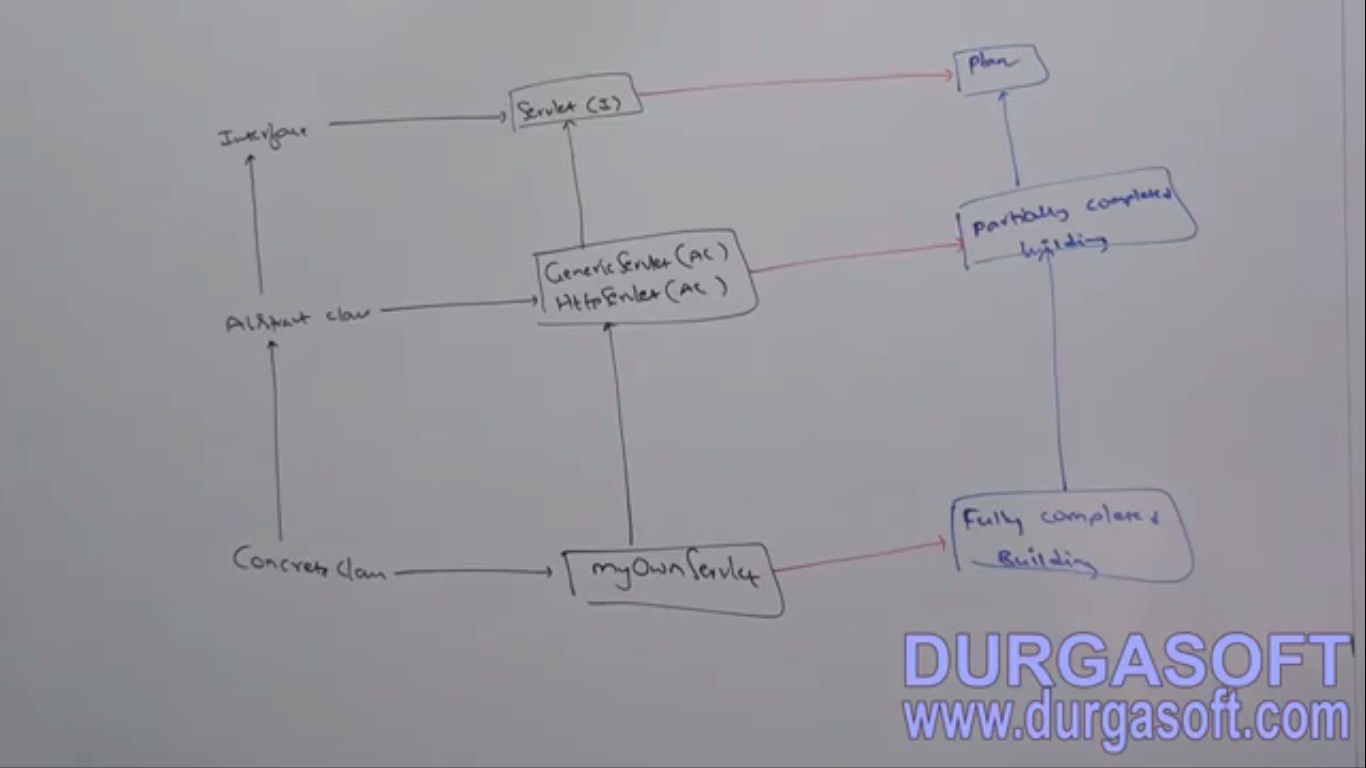
interface vs. abstract vs. concrete class

1. 

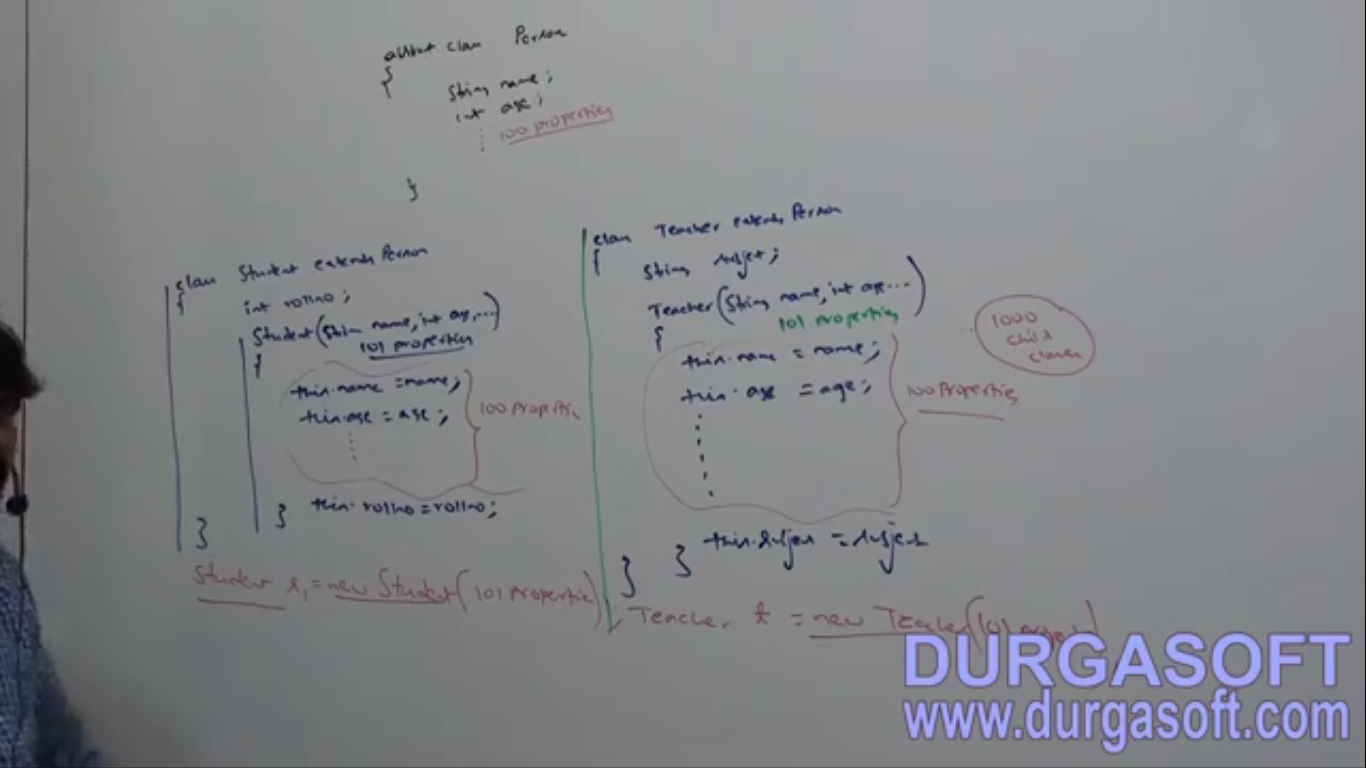
# Difference b/w interface and abstract class

|  |  |
| --- | --- |
| Interface | Abstract class |
| When we have requirement specification. | When Partial implementation  **1st point is the difference b/w them**  **2nd and 3rd points are about methods difference**  **4th, 5th and 6th points are about variable difference.** |
| Each method public abstract = 100% | Each method needs not to be public and abstract, it can have concrete methods also |
| Interface method can’t be declared with modifiers: private, protected, static, synchronized, native, final, strictfp | No restriction |
| Variable always public static final | No restriction |
| These modifiers are allowed for variable: private, protected, volatile, and transient. | No restriction |
| Variable initialization only at the time of declaration | No restriction |
| Static block and instance block not allowed | Allowed |
| Constructor not allowed | allowed |

**By Jatin:**

|  |  |  |
| --- | --- | --- |
|  | **Interface** | **Abstract Class** |
| Difference about specification and implementation | When only requirement specification | When partial implementation |
| Method difference | Each method is abstract public | No restriction |
| Method difference | No other modifiers can be applied on method such as private, protected, static, synchronized, native, final, strictfp | No Restriction |
| Variable Difference | Each variable is final public static | No Restriction |
| Variable Difference | Modifiers not allowed: private, protected, volatile, transient | No Restriction |
| Variable Difference | Variable initialization only at the time of declaration | In static/instance block, declaration time, constructor: |
| Constructor Difference | Constructor not allowed | Allowed |
| Blocks | Static/instance blocks not allowed | Allowed |
|  |  |  |

# We can’t create abstract class object but abstract class can contain constructor. Why?

1. Abstract class constructor will be executed whenever we are creating child class object to perform initialization of child class object’s those properties coming from abstract class.
2. **Jatin: What if we don’t write abstract class constructor and write the abstract class properties initialization code in concrete classes**?
   1. In each concrete class we have to repeat the initialization code.   
      

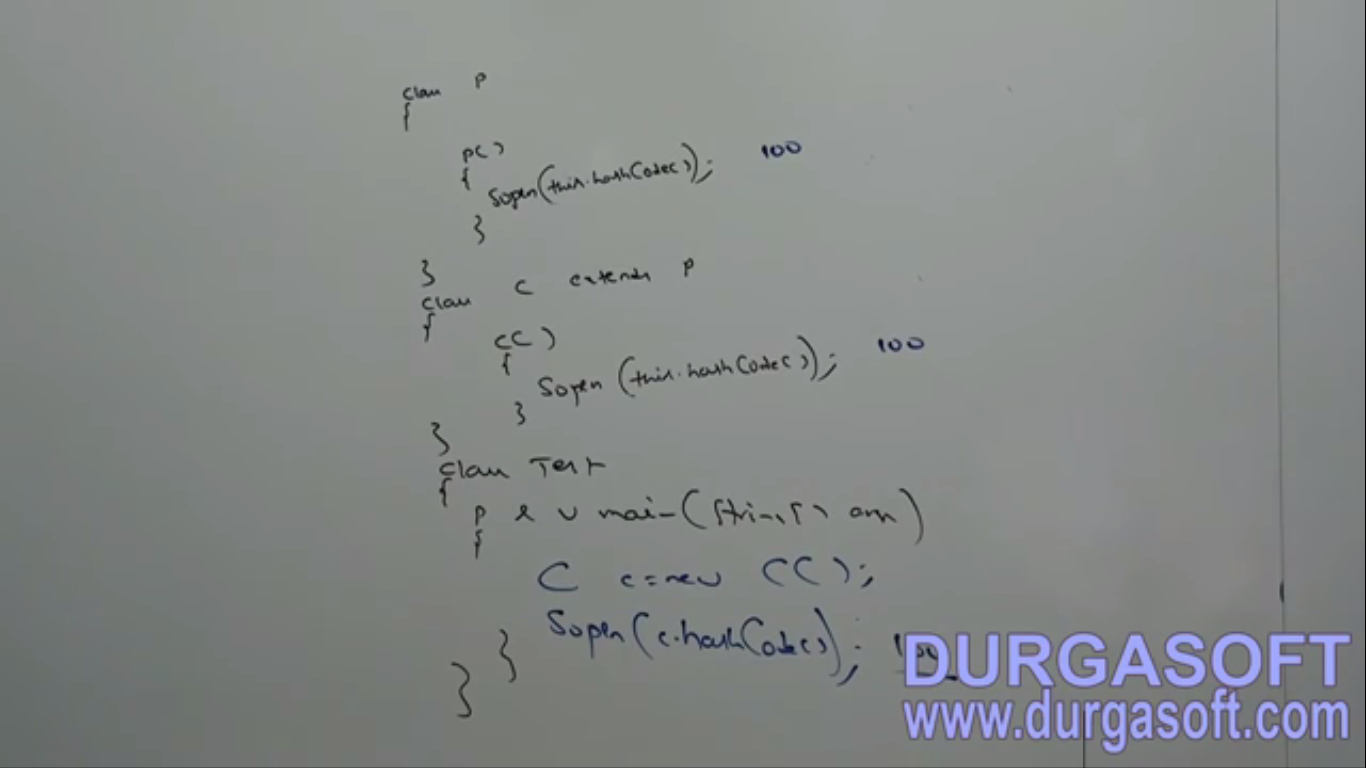
Conclusion from this point 🡪 More Code, Error Prone, Maintainable problem, Update Problem, No Code Reusability. ☺

* 1. What if abstract class some properties are private.

1. d

# Why interface doesn’t have constructor but abstract class?

Because constructor is for initialization for instance variables and interface doesn’t contain instance variables.

Whenever we are creating child class object, parent object will not be created just parent class constructor will be executed for child class object initialization purpose only.   
**Proof**:  


++++++++++++++++++++++++++++++++

**Inside interface, every method is always abstract and we can take only abstract methods inside abstract class also then what is the difference b/w interface and abstract class. That is, is it to replace interface with abstract class?  
Answer**: We can replace interface with abstract class. But it’s not good programming practice. This is something like recruiting IAS officer for Swiping activity.  
If everything is abstract then it’s highly recommended to go interface.

Technical problem if we take abstract class in place of interface.

1. The implemented class needs to extend the abstract class instead of implementing it. Thus losing reusability of code. Now this implemented class can’t inherit from other class.
2. When implemented class instance is created, abstract class constructor and instance blocks controls are executed and taken care by JVM which puts extra over head on instantiation.   
   