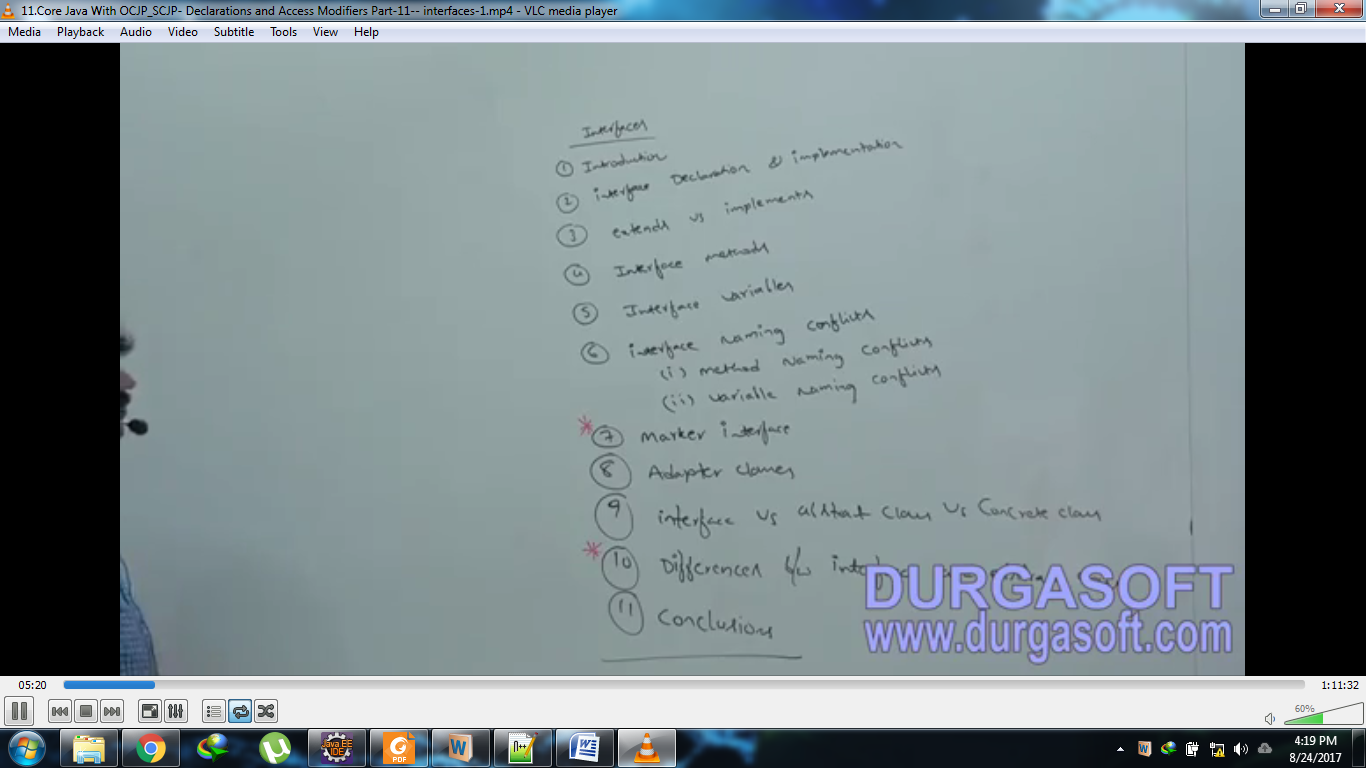
Interface

# syllabus:

1. introduction
2. interface Declaration and implementation
3. extends vs. implements
4. interface methods.
5. interface variables
6. interface naming conflicts
   1. Method naming conflict
   2. Variable naming conflict
7. Marker Interface
8. Adapter classes
9. interface vs. abstract class vs. Concrete class
10. Difference b/w interface and Abstract class
11. conclusions

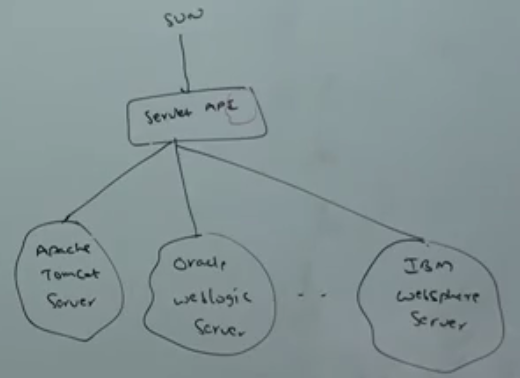


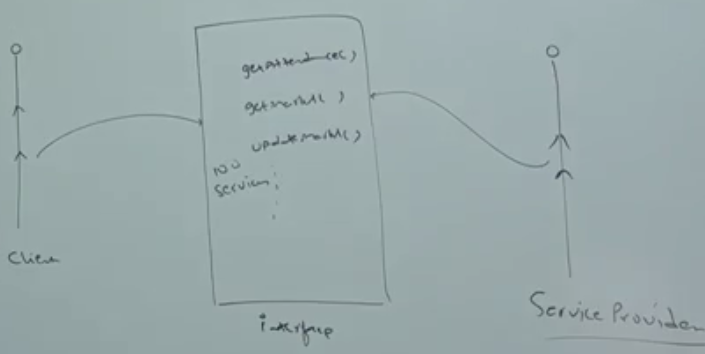
interface introduction

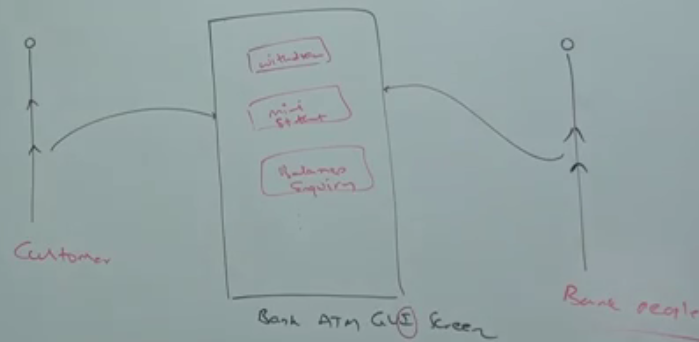
1. **What is an interface?**
   1. **Definition 01:**
      1. **Definition**: Any service requirement specification (SRS) is considered as an interface.
      2. **Example 01**:
         1. JDBC API provides requirement specification to develop database driver. Database vendor is responsible to implement JDBC API.

****

🡨 Requirement Specification. Note I letter in API

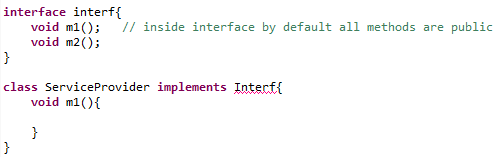
* + 1. **Example 02:**
       1. Servlet API access requirement specification to develop Web Server. Web Server vendors are responsible to implement Servlet-API.   
          ****
  1. **Definition 02:**
     1. **Definition:** from client point of view, interface defines the set of service what he is expecting. From Service Provider point of view, the interface defines the set of service what he is offering. Hence, any contract b/w client and service provider is considered as **an interface.**

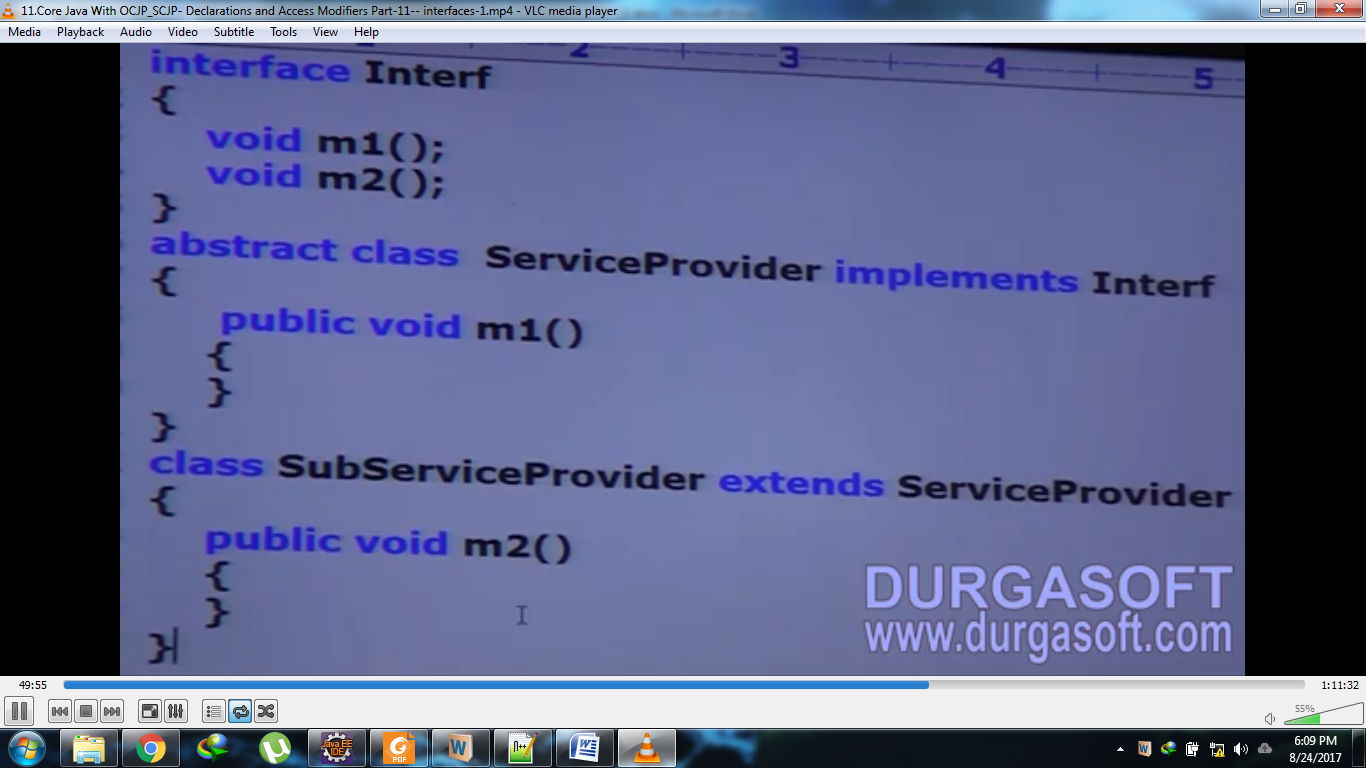
****

* + 1. **Example**: Through BANK ATM GUI screen, Bank people are highlighting the set of services what they are offering. At the same time, the same GUI screen represents the set of services what a customer is expecting. Hence, this GUI access contract b/w customers and Bank People.  
       ****
  1. **Definition 03:**
     1. Inside interface, every method is always abstract whether we are declaring or not. Hence, Interface is considered as **100% pure Abstract Class**.
  2. **Summary Definition:** 
     1. Any service requirement specification or any contract b/w client and service provider or 100% pure abstract class is nothing but **interface**.

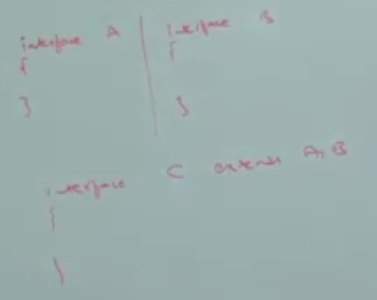
1. **d**

Interface Declaration and implementation



1. Whenever, we are implementing an interface, for each and every method of that interface we have to provide implementation otherwise we have to declare class as **abstract**. Then next level child class is responsible to provide implementation.
2. Every interface method is always public and abstract by default whether we are declaring or not. Hence, whenever we are implementing interface, compulsory we should declare as public. Otherwise we will get compile time error.   
   
3. d

extends vs. implements

1. A class can extend only one class at a time.
2. An interface can extends any number of interfaces simultaneously.   
   
3. A class can implement any number of interfaces simultaneously.
4. A class can extend another class and can implement any number of interfaces simultaneously.   
   

# Which of the following are valid?

1. A class can extend any number of classes at a time?  
   Invalid
2. A class can implement only one interface at a time?  
   Invalid
3. Interface can extend only one interface at a time?  
   Invalid
4. An interface can implement any number of interfaces at a time?  
   Invalid
5. A class can extend another class or can implement an interface but not both simultaneously?  
   Invalid
6. Consider the following expression 🡪 X extends Y. Which of the following options are valid?  
   