Session-19 Abstract classes and Interface

Friday, August 26, 2022 9:06 PM

Abstraction

As per dictionary, abstraction is the quality of **dealing with ideas** rather than events. For example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery. Likewise, In object oriented programming, **Abstraction** is a process of **hiding the implementation details and showing only functionality to the user.**

How to Achieve Abstraction

There are two ways to achieve abstraction in Java

a) Abstract classes (0-100%)

b) Interface (100%)

Abstract Classes

a) An abstract class must be declared with an abstract keyword

b) It can have abstract and non-abstract methods. Abstract methods doesn't have body. The body is provided by the subclass (inherited from).

c) It cannot be instantiated. To access it, it must be inherited from another class.

d) It can have constructors and static methods

e) It can have final methods which will force the subclass not to change the body of the method

3:48 / 1:00:22

Here, we have created the abstract class :-
We will created constructor.
We will created the abstract method and non-abstract method

```
package java_hub;

public abstract class New_047_abstraction_bike {

    New_047_abstraction_bike(){
        System.out.println("bike class created");
    }
```

```java

    abstract void run(); //abstract method

    void start() { // non-abstract method
        System.out.println("bike started ");
    }

    void stop() { // non-abstract method
        System.out.println("bike stop ");
    }

}

```

Created the other class

```java
package java_hub;

public class New_047_abstraction_Honda extends New_047_abstraction_bike {

    @Override
    void run() {
        System.out.println("honda bike run");

    }

}

```

Created the other class

```java
package java_hub;

public class New_047_abstraction_TVS extends New_047_abstraction_bike {

    @Override
    void run() {
        System.out.println("tvs bike run");

    }

}

```

Created the other class

```java
package java_hub;

public class New_047_abstraction_yamaha extends New_047_abstraction_bike {

    @Override
    void run() {
        System.out.println("yamaha bike run");

    }

}

```

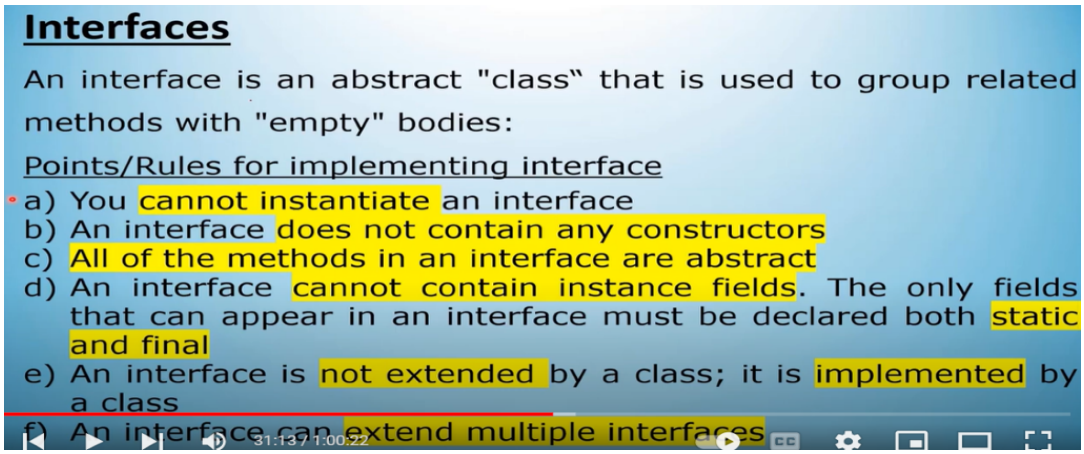
Crated the main class

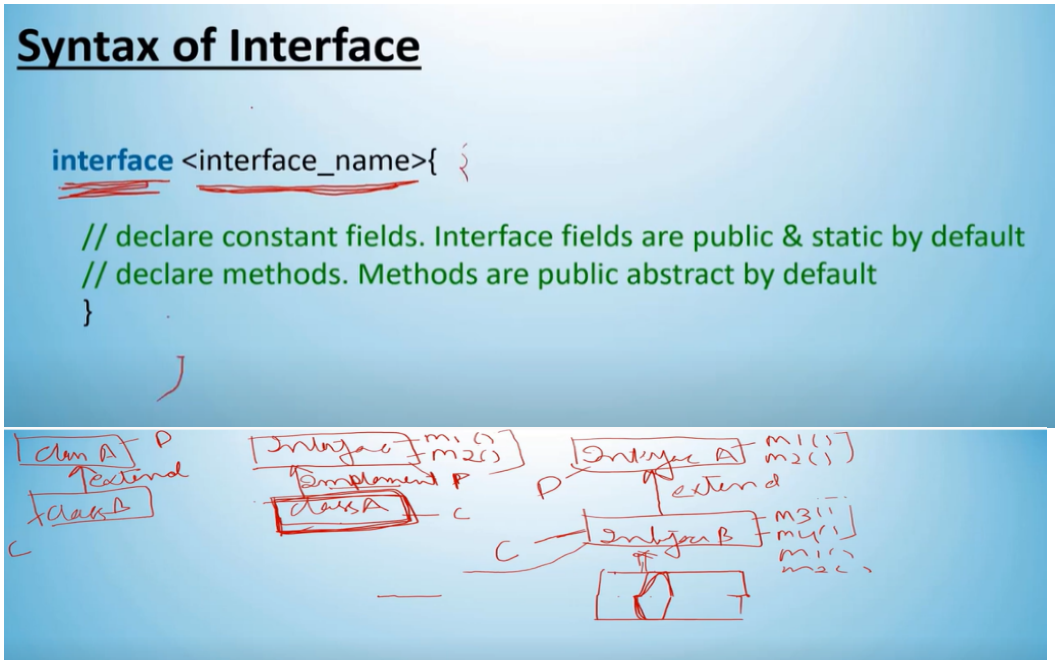
```java
package java_hub;

public class New_047_abstraction_main {

    public static void main(String[] args) {

        New_047_abstraction_bike obj = new New_047_abstraction_Honda();
        obj.start();
        obj.run();
        obj.stop();

        New_047_abstraction_bike obj1 = new New_047_abstraction_TVS();
        obj1.start();
        obj1.run();
        obj1.stop();

        New_047_abstraction_bike obj2 = new New_047_abstraction_yamaha();
        obj2.start();
        obj2.run();
        obj2.stop();


    }

}
```

```
<terminated> New_047_abstraction_main [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (26-Aug-2022, 9:57:38 pm – 9:57:38 pm) [pid: 19392]
bike class created
bike started
honda bike run
bike stop
bike class created
bike started
tvs bike run
```

Points/Rules form implementing abstract class-

1) An abstract class must be declared with an abstract keyword
2) It can have abstract and non-abstract methods. Abstract methods doesn't have body. The body is provided by the subclass (inherited from).
3) It cannot be instantiated. To access it, it must be inherited from another class.
4) It can have constructors and static methods
5) It can have final methods which will force the subclass not to change the body of the method





File Edit Format View Help

An interface is an abstract "class" that is used to group related methods with "empty" bodies

Points/Rules for implementing interface
---
1. You cannot instantiate an interface
2. An interface does not contain any constructors
3. All of the methods in an interface are abstract
4. An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final
5. An interface is not extended by a class; it is implemented by a class
6. An interface can extend multiple interfaces