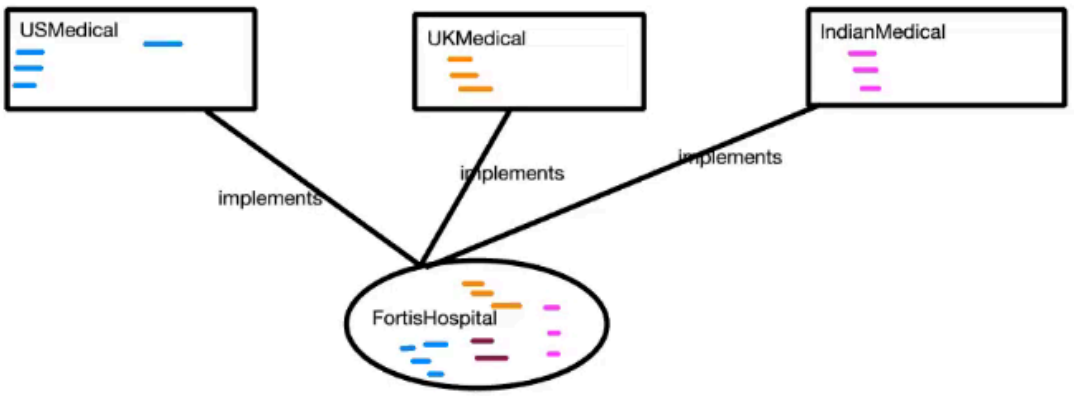
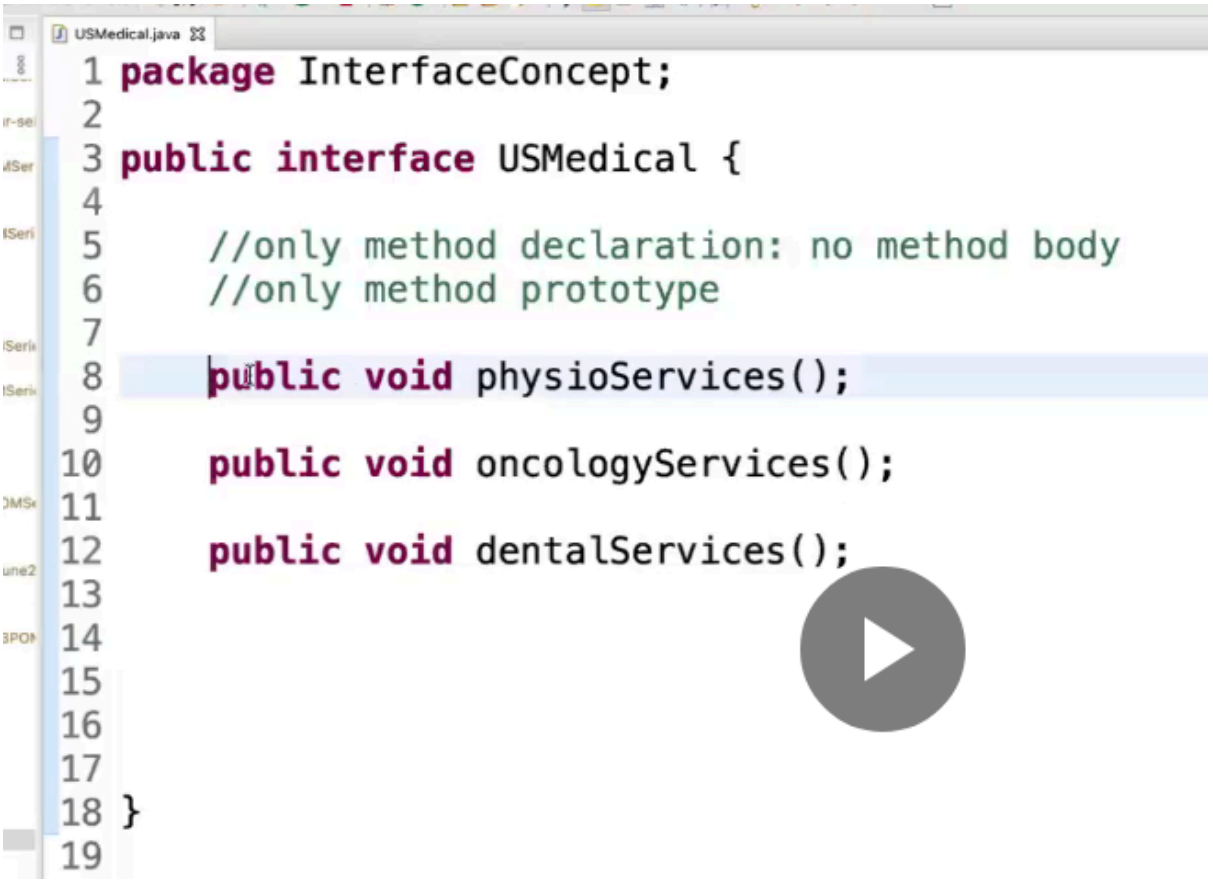
Interface-

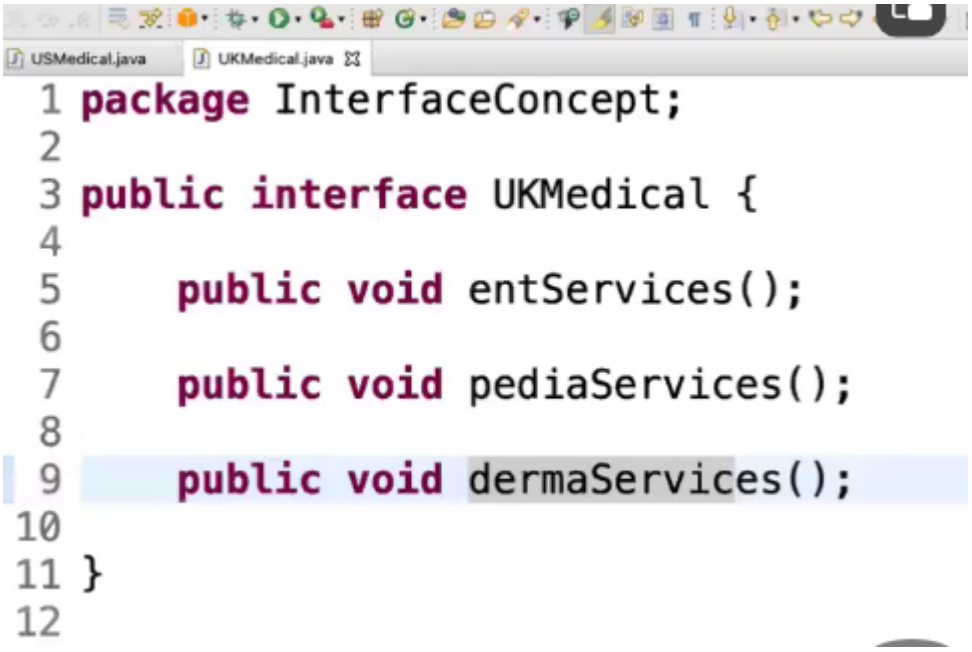


Interface will give error when trying to give method body- //Abstract methods do not specify a body

USMedical.java

```
1  package InterfaceConcept;
2
3  public interface USMedical {
4
5      public void test() {
6
7      }
8
9
10
```


USMedical.java
1 package InterfaceConcept;
2
3 public interface USMedical {
4
5 //only method declaration: no method body
6 //only method prototype
7
8 public void physioServices();
9
10 public void oncologyServices();
11
12 public void dentalServices();
13
14
15
16
17
18 }
19


USMedical.java
UKMedical.java
1 package InterfaceConcept;
2
3 public interface UKMedical {
4
5 public void entServices();
6
7 public void pediaServices();
8
9 public void dermaServices();
10
11 }
12

USMedical.java UKMedical.java IndianMedical.java

```
1  package InterfaceConcept;
2
3  public interface IndianMedical {
4
5      public void cardioServices();
6
7      public void orthoServices();
8
9      public void nuroServices();
10
11 }
12
```

Abstract methods-

USMedical.java UKMedical.java IndianMedical.java FortisHospital.java

```
1  package InterfaceConcept;
2
3  public interface USMedical {
4
5      //only method declaration: no method body
6      //only method prototype
7      //abstract methods: a method without body
8
9      public void physioServices();
10
11     public void oncologyServices();
12
13     public void dentalServices();
14
15 }
```

USMedical.java | UKMedical.java | IndianMedical.java | FortisHospital.java

```java
package InterfaceConcept;

public class FortisHospital implements USMedical {

    @Override
    public void physioServices() {
        System.out.println("FH -- physioServices");
    }

    @Override
    public void oncologyServices() {
        System.out.println("FH -- oncologyServices");
    }

    @Override
    public void dentalServices() {
        System.out.println("FH -- dentalServices");
    }

}
```

```java
public class FortisHospital implements USMedical, UKMedical {
```

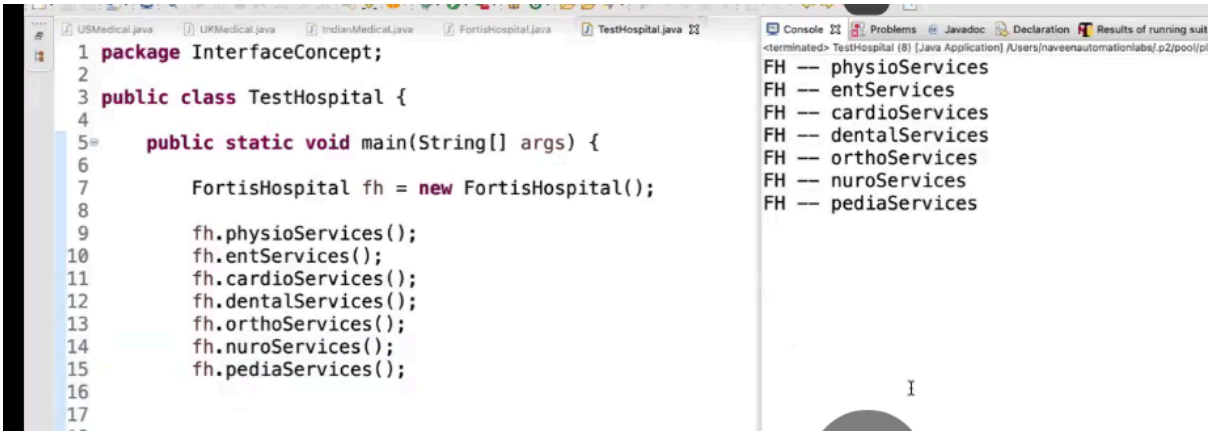
```java
    }

    //UK
    @Override
    public void entServices() {
        System.out.println("FH -- entServices");

    }

    @Override
    public void pediaServices() {
        System.out.println("FH -- pediaServices");

    }

    @Override
    public void dermaServices() {
        System.out.println("FH -- dermaServices");

    }

}
```

```java
public class FortisHospital implements USMedical, UKMedical, IndianMedical {
```

```
39
40      //India
41      @Override
42      public void cardioServices() {
43          System.out.println("FH -- cardioServices");
44
45      }
46
47      @Override
48      public void orthoServices() {
49          System.out.println("FH -- orthoServices");
50
51      }
52
53      @Override
54      public void nuroServices() {
55          System.out.println("FH -- nuroServices");
56
57      }
58
```

```
59      //Individual:
60      public void medicalTraining() {
61          System.out.println("Fh -- medical training");
62      }
63
64      public void pathalogyServices() {
65          System.out.println("Fh -- pathalogyServices");
66
67      }
68
```

Call all methods-



Call individual methods-

```
17          fh.medicalTraining();
18          fh.pathalogyServices();
19
```

```
Fh -- pediaServices
Fh -- medical training
Fh -- pathalogyServices
```

Common method in all interfaces-

Us, uk, india.

```
14
15      public void emergencyServices();
16
```

Only once we have to override-

```
68
69      //common
70      @Override
71      public void emergencyServices() {
72          System.out.println("Fh -- emergencyServices");
73      }
74
```

test class-

```
19
20          fh.emergencyServices();
21
```

```
Fh -- pathalogyServices
Fh -- emergencyServices
```

```
4
5       //only method declaration: no method body
6       //only method prototype
7       //abstract methods: a method without body
8       //can not create the object of interface
9
```

Paste usmedical3-

usmedical3.java

```java
package com.day19;

public interface usmedical3 {

    public void physio();
    public String onco();
    public void dental();

    //common method in all interfaces.
    public void emergency();

    //static method inside interface.
    public static int getstatic(String name) {
        System.out.println("static method" + name);
        return 10;
    }

    //final method inside interface.
    //Illegal modifier for the interface method getfinal; only public, private,
    //abstract, default, static and strictfp are permitted
//  public final int getfinal(String name) {
//      System.out.println("static method" + name);
//      return 10;
//  }
```

```java
//      return 10;
//  }

    //Illegal modifier for the interface method getfinalandstatic;
    //only public, private, abstract, default, static and strictfp are permitted
    public final static int getfinalandstatic(String name) {
        System.out.println("static method" + name);
        return 10;
    }

    //Illegal modifier for the interface method getfinal; only public,
    //private, abstract, default, static and strictfp are permitted
    public final int getfinal(String name);

    //This method requires a body instead of a semicolon
    public final static int getfinalandstatic(String name);



}
```

Paste usmedical7-

usmedical7.java

```java
package com.day19;

public interface usmedical7 {

    public void physio();
    public String onco();
    public void dental();

    //common method in all interfaces.
    public void emergency();

    //static method inside interface.
    public static int getstatic(String name) {
        System.out.println("static method" + name);
        return 10;
    }

    //static method inside interface.
    //This method requires a body instead of a semicolon
    //static methods have to be implemented where declared else error.
        public static int getstatic1(String name);

}
```

Paste usmedical8, testfor8, fortis8-

usmedical8.java

```java
package com.day19;

public interface usmedical8 {

    public void physio();
    public String onco();
    public void dental();

    //common method in all interfaces.
    public void emergency();

    //static method inside interface.
    public static int getstatic(String name) {
        System.out.println("static method" + name);
        return 10;
    }

        public static int getstatic1(String name) {
            System.out.println("get static 1 method from interface " + name);
            return 10;
        }

}
```

usmedical8.java | fortis8.java

```java
package com.day19;

//can implement multiple interface.
public class fortis8 implements usmedical8{

    @Override
    public void physio() {
        System.out.println("fortis physio");
    }

    @Override
    public String onco() {
        System.out.println("fortis onco");
        return "tiger";
    }

    @Override
    public void dental() {
        System.out.println("fortis dental");
    }

    @Override
    public void emergency() {
        System.out.println("fortis emergency");
    }
```

```java
        System.out.println("fortis emergency");
    }

    //not mandatory to override static method.
    //if we write override, we get error.
    //The method getstatic1(String) of type fortis8 must override or implement a supertype method
//  @Override
//  public static int getstatic1(String name) {
//      System.out.println("get static 1 method");
//      return 10;
//  }

    public static int getstatic1(String name) {
        System.out.println("get static 1 method from child class");
        return 190;
    }
```

```java
        return 190;
    }



}
```

usmedical8.java | fortis8.java | testfor8.java

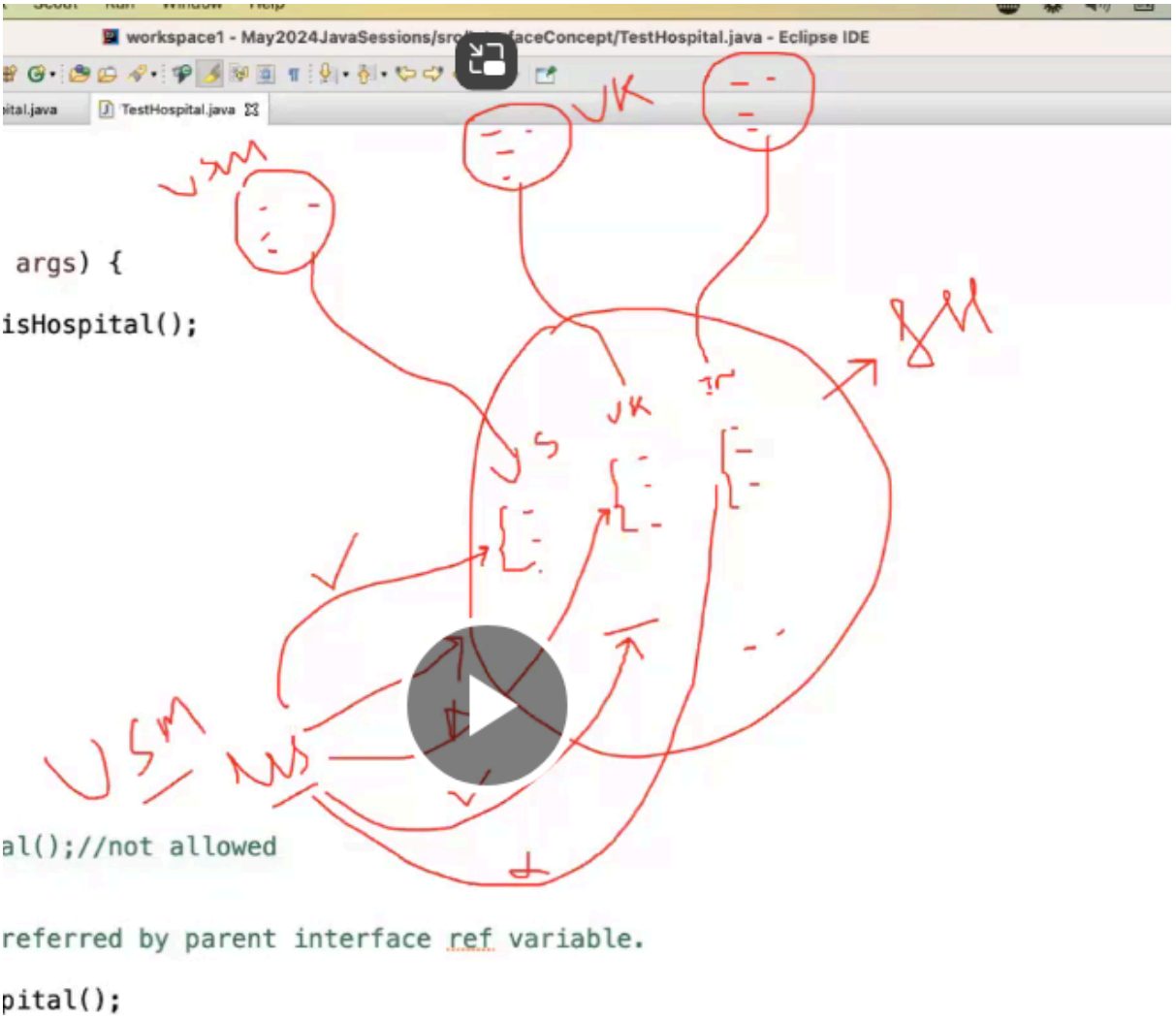
```java
package com.day19;

public class testfor8 {

    public static void main(String[] args) {

        //create object of fortis8 and call all methods.
        //capture all return values and print them also.

        fortis8 f1=new fortis8();
        f1.physio();
        String o1=f1.onco();
        System.out.println(o1);
        f1.dental();
        f1.emergency();
        int i1=f1.getstatic1("karan");
        //warning.
        //The static method getstatic1(String) from the type fortis8
        //should be accessed in a static way
        System.out.println(i1);

        //access with child class name.
        int i2=fortis8.getstatic1("lion");
        System.out.println(i2);

        int i3=usmedical8.getstatic1("cheetah");
        System.out.println(i3);
```

```
26          int i3=usmedical8.getstatic1("cheetah");
27          System.out.println(i3);
28
29      }
30
31  }
32
33  //fortis physio
34  //fortis onco
35  //tiger
36  //fortis dental
37  //fortis emergency
38  //get static 1 method from child class
39  //190
40  //get static 1 method from child class
41  //190
42  //get static 1 method from interface cheetah
43  //10
44
45
46
47
```

Object of interface not allowed-

Cannot instantiate the type usmedical9

```
22
23          //
24          //USMedical us = new USMedical();//not allowed
25
```



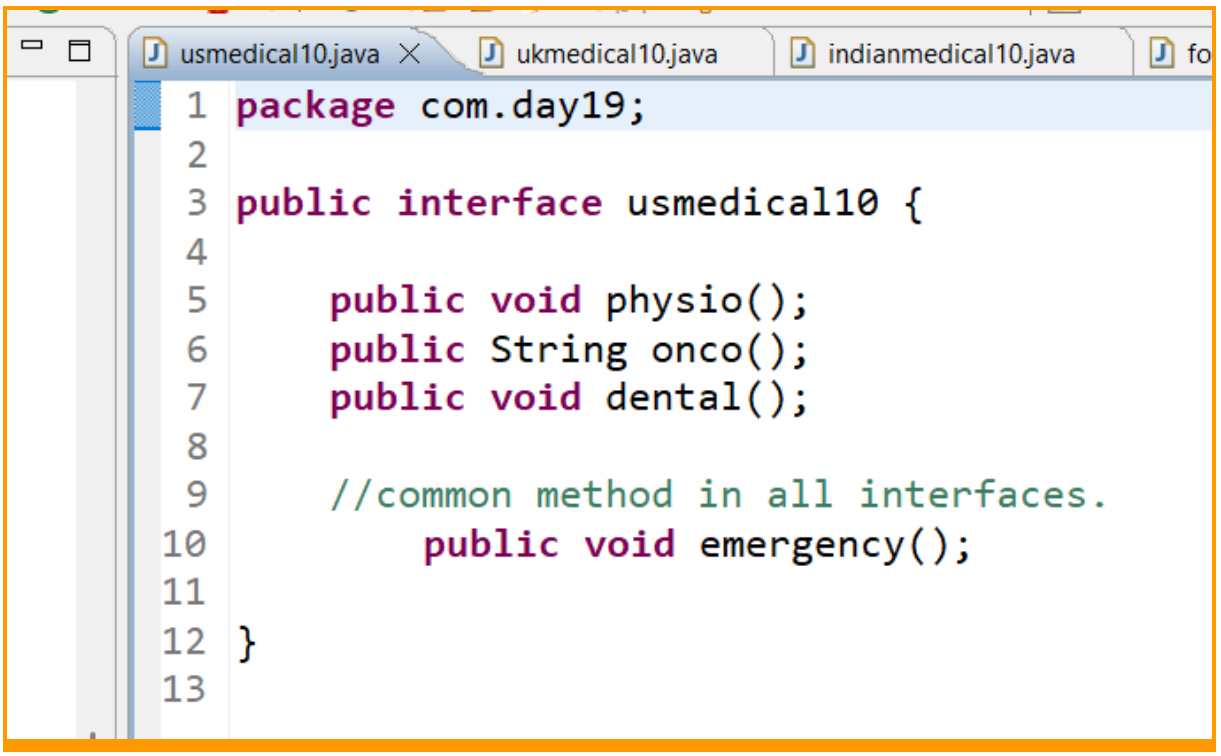
We can access only of USA below-

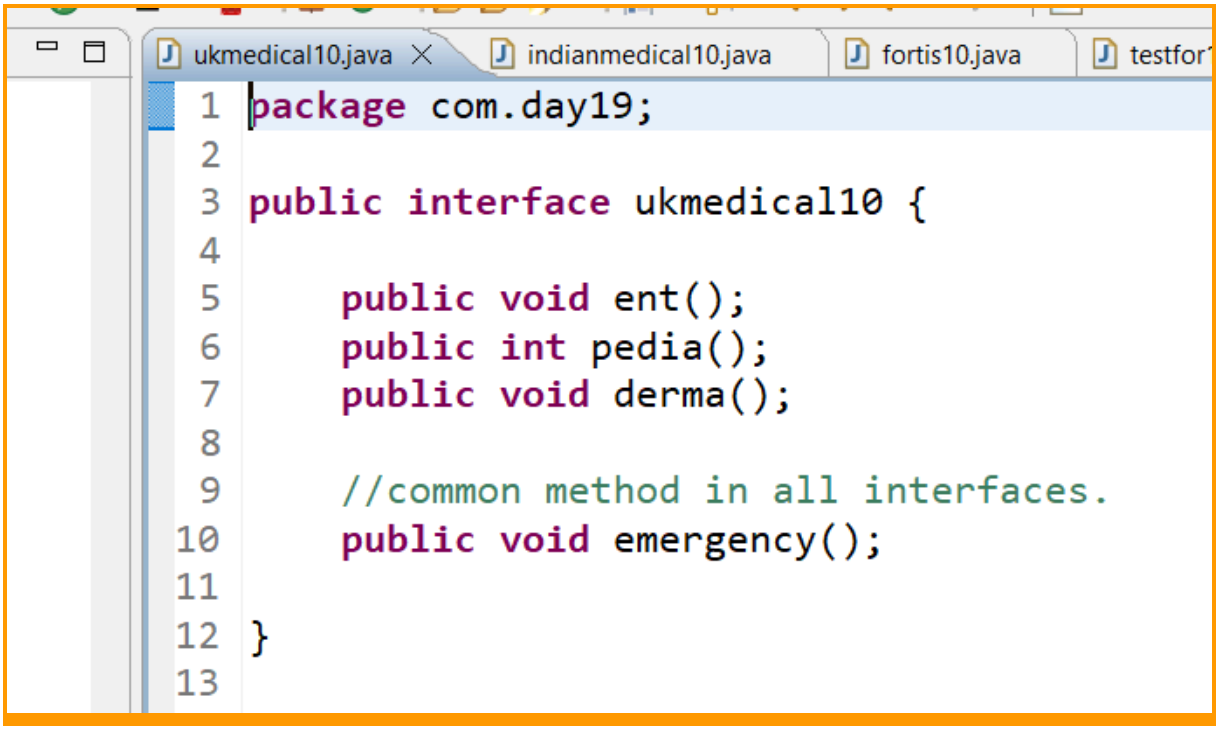
India, UK cant be accessed.

```
25
26          //top casting:
27          //child class object can be referred by parent interface ref variable.
28
29          USMedical us = new FortisHospital();
30          us.oncologyServices();
31          us.physioServices();
32          us.dentalServices();
33          us.emergencyServices();
34
35
```

Common methods can be accessed.

Paste usmedical10, fortis10, ukmedical10, indianmedical10, testfor10-





indianmedical10.java | fortis10.java | testfor10.java

```java
package com.day19;

public interface indianmedical10 {

    public void cardio();
    public String neuro();
    public void ortho();

    //common method in all interfaces.
        public void emergency();

}

```

fortis10.java | testfor10.java

```java
package com.day19;

//can implement multiple interface.
public class fortis10 implements usmedical10, ukmedical10, indianmedical10{

    String name;
    int age;

    public fortis10(String name, int age) {
        this.name=name;
        this.age=age;
        System.out.println("this is constructor");
    }

    @Override
    public void physio() {
        System.out.println("fortis physio");

    }

    @Override
    public String onco() {
        System.out.println("fortis onco");
        return "onco";
    }
```

```
24          return "onco";
25      }
26
27      @Override
28      public void dental() {
29          System.out.println("fortis dental");
30
31      }
32
33      @Override
34      public void ent() {
35          System.out.println("fortis ent");
36
37      }
38
39      @Override
40      public int pedia() {
41          System.out.println("fortis pedia");
42          return 10;
43      }
```

```
42          return 10;
43      }
44
45      @Override
46      public void derma() {
47          System.out.println("fortis derma");
48
49      }
50
51      @Override
52      public void cardio() {
53          System.out.println("fortis cardio");
54
55      }
56
57      @Override
58      public String neuro() {
59          System.out.println("fortis neuro");
60          return "a";
61      }
```

```
60          return "a";
61      }
62
63      @Override
64      public void ortho() {
65          System.out.println("fortis ortho");
66      }
67
68      //create two methods inside fortis.
69
70      public void eye() {
71          System.out.println("fortis eye");
72      }
73
74      public void ear() {
75          System.out.println("fortis ear");
76      }
```

```
75          System.out.println("fortis ear");
76      }
77
78      //common methods across interface needs to be implemented only once.
79      @Override
80      public void emergency() {
81          System.out.println("fortis emergency");
82      }
83
84  }
85
```

testfor10.java

```java
package com.day19;

public class testfor10 {

    public static void main(String[] args) {

        //can access only usmedical10 and common methods.
        usmedical10 u1=new fortis10("karan", 10);
        u1.dental();
        u1.emergency();
        String s1=u1.onco();
        System.out.println(s1);
        u1.physio();
    //only us medical and common methods across interfaces can be accessed.



    }

}

//this is constructor
//fortis dental
//fortis emergency
//fortis onco
//onco
//fortis physio

```

Downcasting not allowed-

```java
        //down casting:
        //FortisHospital f1 = new USMedical();
```

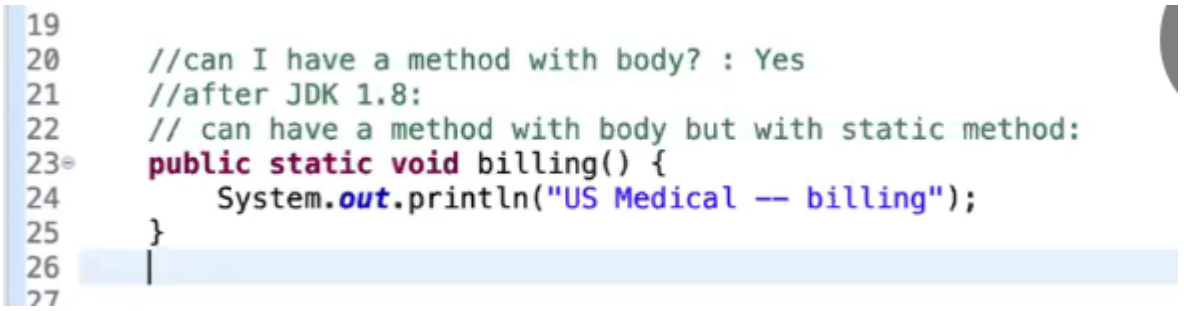
Paste testfor11-

testfor11.java

```java
package com.day19;

public class testfor11 {

    public static void main(String[] args) {

        //when we create object of interface, it asks to implement them.
        fortis10 f1=new usmedical10() {

            @Override
            public void physio() {
                // TODO Auto-generated method stub

            }

            @Override
            public String onco() {
                // TODO Auto-generated method stub
                return null;
            }

            @Override
            public void emergency() {
                // TODO Auto-generated method stub

            }

            }

            @Override
            public void dental() {
                // TODO Auto-generated method stub

            }
        };



    }

}

//this is constructor
//fortis dental
//fortis emergency
//fortis onco
//onco
//fortis physio
```

us medical interface-

we can have “n” number of static methods.



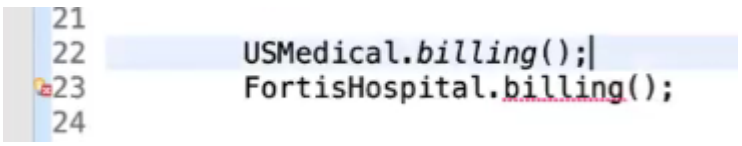
```
19
20      //can I have a method with body? : Yes
21      //after JDK 1.8:
22      // can have a method with body but with static method:
23      public static void billing() {
24          System.out.println("US Medical -- billing");
25      }
26
27
```

If no definition for static then error -

// This method requires a body instead of a semicolon

test class-

Can be called by interface only and not fortis-



```
21
22          USMedical.billing();
23          FortisHospital.billing();
24
```

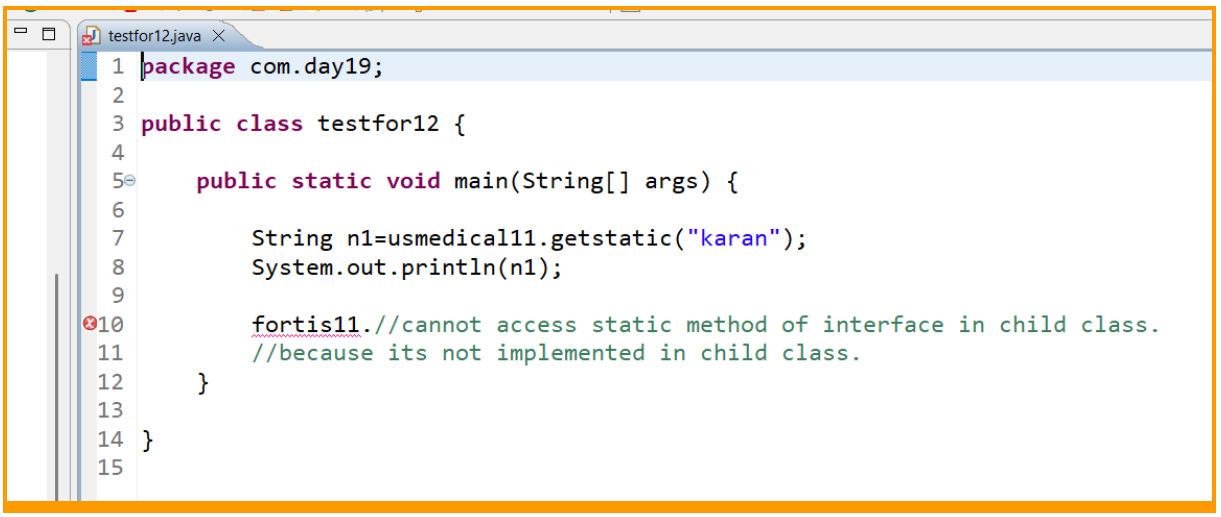
Paste usmedical11, fortis11, testfor12 -

usmedical11.java | fortis11.java | testfor12.java

```java
package com.day19;

public interface usmedical11 {

    //in interface we can have static method but we have to define it.
    //create one static method
    public static String getstatic(String name) {
        System.out.println("static method" + name);
        return "tiger";
    }

    //if no definition then error-
//  This method requires a body instead of a semicolon
//  public static String getstatic1(String name);

}
```

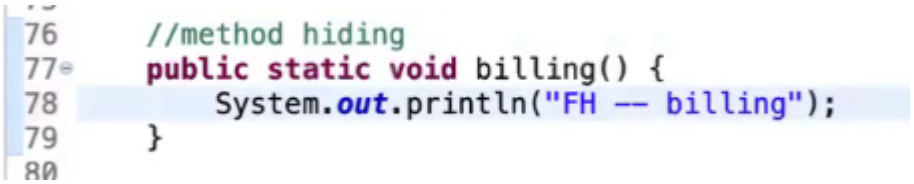
fortis11.java | testfor12.java

```java
package com.day19;

//can implement multiple interface.
public class fortis11 implements usmedical11{

    public void physio() {
        System.out.println("physio");
    }
    public void onco() {
        System.out.println("onco");
    }
    public void dental() {
        System.out.println("dental");
    }

}
```



```java
package com.day19;

public class testfor12 {

    public static void main(String[] args) {

        String n1=usmedical11.getstatic("karan");
        System.out.println(n1);

        fortis11.//cannot access static method of interface in child class.
        //because its not implemented in child class.
    }

}
```

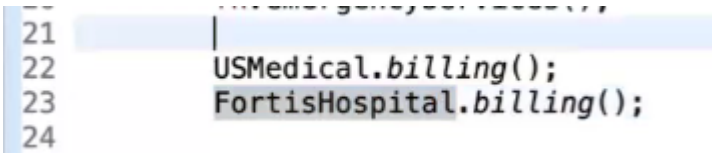
Same method of interface in fortis class-



```java
    //method hiding
    public static void billing() {
        System.out.println("FH -- billing");
    }
```

static method cannot be overridden, so no use of writing @override in child class. so we wont get error in child class to implement the static method without annotation or if we write the same method also.

Now fortis can call its method-



```java

        USMedical.billing();
        FortisHospital.billing();
```

Paste usmedical12, fortis12, testfor13-

usmedical12.java | fortis12.java | testfor13.java

```java
package com.day19;

public interface usmedical12 {

    //in interface we can have static method but we have to define it.
    //create one static method
    public static String getstatic(String name) {
        System.out.println("static method" + name);
        return "tiger";
    }

    //if no definition then error-
//  This method requires a body instead of a semicolon
//  public static String getstatic1(String name);

}

```

fortis12.java | testfor13.java

```java
package com.day19;

//can implement multiple interface.
public class fortis12 implements usmedical12{

    public void physio() {
        System.out.println("physio");
    }
    public void onco() {
        System.out.println("onco");
    }
    public void dental() {
        System.out.println("dental");
    }

    //The method getstatic(String) of type fortis12 must override or implement a supertype method
    //cannot use override for static method.
//  @Override
//  public static String getstatic(String name) {
//      System.out.println("static method" + name);
//      return "tiger";
//  }
```

```
21 //      return "tiger";
22 //  }
23
24     //without override it works.
25     //treated like another method.
26     public static String getstatic(String name) {
27     System.out.println("static child class method" + name);
28     return "lion";
29 }
30
31 }
32
```

testfor13.java

```
1  package com.day19;
2
3  public class testfor13 {
4
5      public static void main(String[] args) {
6
7          String s1=usmedical12.getstatic("karan");
8          System.out.println(s1);
9
10         String s2=fortis12.getstatic("naman");
11         System.out.println(s2);
12     }
13
14 }
15
16 //static methodkaran
17 //tiger
18 //static child class methodnaman
19 //lion
20
```

Static methods has to have body.

Abstract methods need not have body.

```
5     //only method declaration: no method body
6     //only method prototype
7     //abstract methods: a method without body(only for non static methods)
8     //can not create the object of interface
9     //static method with body
```

us medical interface -

```
27
28      // 2. can have a non static with method body but it should written with default keyword:
29      default void medicalInsurance() {
30          System.out.println("US -- medical insurance");
31      }
32
```

test class-

```
20          fh.emergencyServices();
21          fh.medicalInsurance();
22
```

```
Fh -- emergencyServices
US -- medical insurance
```

Interface can also call-

```
36          us.emergencyServices();
37          us.medicalInsurance();
38
```

Paste usmedical13, fortis13, testfor14-

usmedical13.java | fortis13.java | testfor14.java

```java
package com.day19;

public interface usmedical13 {

    //in interface we can have default method.
    //but it has to have body.

    //create default method.
    default int getdefault(String name) {
        System.out.println("default from interface method" + name);
        return 10;
    }


    //without body error -
    //This method requires a body instead of a semicolon
//  default int getdefault(String name);

}
```

fortis13.java | testfor14.java

```java
package com.day19;

//can implement multiple interface.
public class fortis13 implements usmedical13{

    //cannot override default.
    //Cannot reduce the visibility of the inherited method from usmedical13
//  @Override
//  default int getdefault(String name) {
//      System.out.println("default method" + name);
//      return 10;
//  }


}
```

testfor14.java

```java
package com.day19;

public class testfor14 {

    public static void main(String[] args) {

        fortis13 f1=new fortis13();
        int v1=f1.getdefault("tiger");
        System.out.println(v1);

        //Cannot instantiate the type usmedical13
//      usmedical13 u1=new usmedical13();
//      int v2=u1.getdefault("gorilla");
//      System.out.println(v2);


        usmedical13 u1=new fortis13();
        int v2=u1.getdefault("gorilla");
        System.out.println(v2);
    }

}

//default from interface methodtiger
//10
//default from interface methodgorilla
//10

```

How to override default method in class-

Fortis class.

```java

    //trying to override default method of interface with public:
    @Override
    public void medicalInsurance() {
        System.out.println("US -- medical insurance");
    }
```

We have to change access modifier to something other than default, else throws error.

//Cannot reduce the visibility of the inherited method from usmedical13

Us medical-

```
27
28      // 2. can have a non static with method body but it should written with default keyword:
29      //can we override the default method of Interface?: YES
30      default void medicalInsurance() {
31          System.out.println("US -- medical insurance");
32      }
33
34  }
35
```

We can have n number of default and static methods inside the interface.

us medical interface-

```
4
5       int MIN_FEE = 10;
6       //interface vars are static and final in nature by default
7
```

Whether we write or not.

Test-

```
25
26          System.out.println(USMedical.MIN_FEE);
27
```

10

Same variable in fortis class-

USMedical.java | UKMedical.java | IndianMedical.java | FortisHospital.java | TestHospital.java

```
1  package InterfaceConcept;
2
3  public class FortisHospital implements USMedical, UKMedical, IndianMedical {
4
5
6      static final int MIN_FEE = 50;
7
```

Test-

```
25
26          System.out.println(USMedical.MIN_FEE);//10
27          //USMedical.MIN_FEE = 100;
28
29          System.out.println(FortisHospital.MIN_FEE);//50
```

Paste usmedical14, fortis14, testfor15-

usmedical14.java | fortis14.java | testfor15.java

```
1  package com.day19;
2
3  public interface usmedical14 {
4
5      int MINFEE=214234; //by default public static final.
6
7      //in interface we can have default method.
8      //but it has to have body.
9
10     //create default method.
11     default int getdefault(String name) {
12         System.out.println("first default from interface" + name);
13         return 10;
14     }
15
16
17 //n number of default and static methods allowed.
18     default void getdefault1(String name) {
19         System.out.println("second default from interface" + name);
20     }
21
22     //Illegal modifier for the interface method getstatic1; only public, private, abstract, default,
   static and strictfp are permitted
23 //  public static final int getstatic1(String name) {
24 //      System.out.println("third default from interface" + name);
25 //      return 10;
26 //  }
```

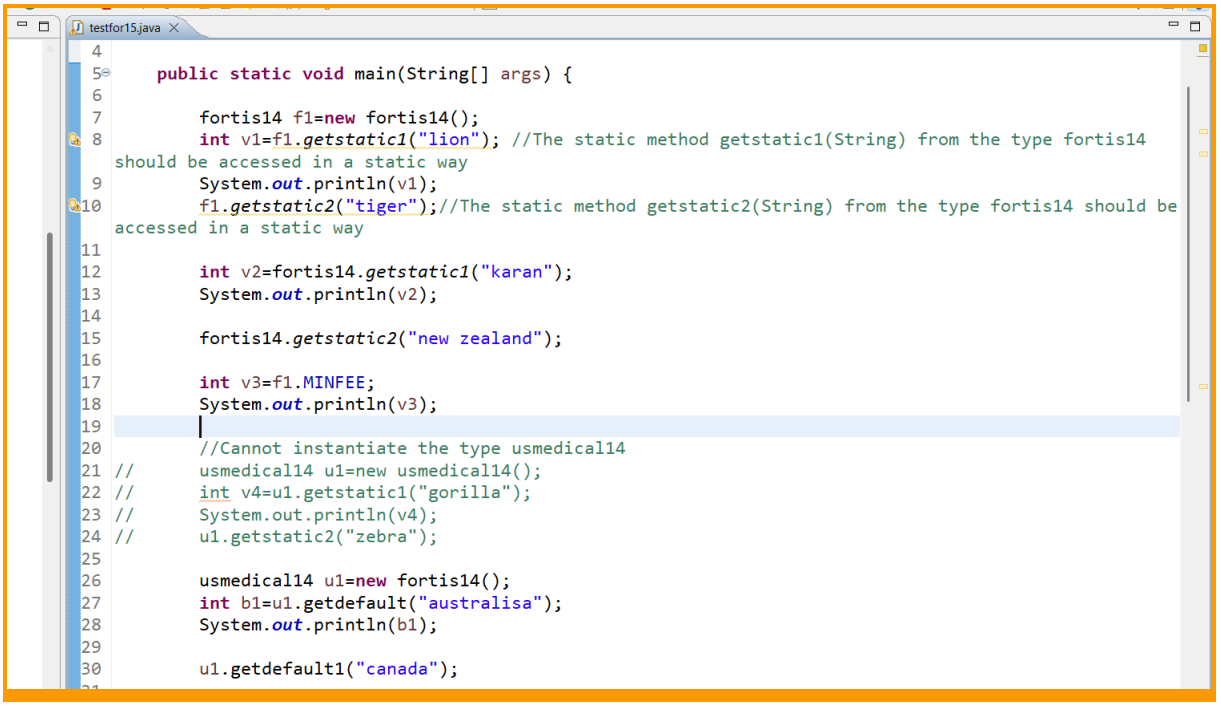
```
25 //      return 10;
26 //  }
27
28      public static void getstatic2(String name) {
29          System.out.println("first static from interface" + name);
30      }
31
32      public static int getstatic1(String name) {
33          System.out.println("second static from interface" + name);
34          return 10;
35      }
36 }
37
```

fortis14.java testfor15.java

```java
package com.day19;

//can implement multiple interface.
public class fortis14 implements usmedical14{

    int MINFEE=654657567;

    //Cannot reduce the visibility of the inherited method from usmedical14
//  @Override
//  default int getdefault(String name) {
//      System.out.println("first default from interface" + name);
//      return 10;
//  }

    //Cannot reduce the visibility of the inherited method from usmedical14
//  default int getdefault(String name) {
//      System.out.println("first default from interface" + name);
//      return 10;
//  }

    //Cannot reduce the visibility of the inherited method from usmedical14
//  @Override
//  default void getdefault1(String name) {
//      System.out.println("second default from interface" + name);
//  }

//Cannot reduce the visibility of the inherited method from usmedical14
//  default void getdefault1(String name) {
//      System.out.println("second default from interface" + name);
//  }
```

```java
//      System.out.println("second default from interface" + name);
//  }

    //The method getstatic2(String) of type fortis14 must override or implement a supertype method
//  @Override
//  public static void getstatic2(String name) {
//      System.out.println("first static from interface" + name);
//  }

    public static void getstatic2(String name) {
        System.out.println("first static from child class" + name);
    }

    //The method getstatic1(String) of type fortis14 must override or implement a supertype method
//  @Override
//  public static int getstatic1(String name) {
//      System.out.println("second static from interface" + name);
//      return 10;
//  }
```

```java
//      return 10;
//  }


    public static int getstatic1(String name) {
        System.out.println("second static from child class" + name);
        return 10;
    }



}
```


testfor15.java
4
5 public static void main(String[] args) {
6
7 fortis14 f1=new fortis14();
8 int v1=f1.getstatic1("lion"); //The static method getstatic1(String) from the type fortis14 should be accessed in a static way
9 System.out.println(v1);
10 f1.getstatic2("tiger");//The static method getstatic2(String) from the type fortis14 should be accessed in a static way
11
12 int v2=fortis14.getstatic1("karan");
13 System.out.println(v2);
14
15 fortis14.getstatic2("new zealand");
16
17 int v3=f1.MINFEE;
18 System.out.println(v3);
19
20 //Cannot instantiate the type usmedical14
21 // usmedical14 u1=new usmedical14();
22 // int v4=u1.getstatic1("gorilla");
23 // System.out.println(v4);
24 // u1.getstatic2("zebra");
25
26 usmedical14 u1=new fortis14();
27 int b1=u1.getdefault("australisa");
28 System.out.println(b1);
29
30 u1.getdefault1("canada");

```
29
30          u1.getdefault1("canada");
31
32          int b2=u1.MINFEE;
33          System.out.println(b2);
34
35          //cant access interface static with the object name.
36          //once same method written in child,
37          //parent cannot access it.
38
39          //can access with interface name.
40          int c1=usmedical14.getstatic1("usa");
41          System.out.println(c1);
42
43
44          usmedical14.getstatic2("west indies");
45      }
```

```
44          usmedical14.getstatic2("west indies");
45      }
46
47  }
48
49  //second static from child classlion
50  //10
51  //first static from child classtiger
52  //second static from child classkaran
53  //10
54  //first static from child classnew zealand
55  //654657567
56  //first default from interfaceaustralisa
57  //10
58  //second default from interfacecanada
59  //214234
60  //second static from interfaceusa
61  //10
62  //first static from interfacewest indies
63
64
```

Paste usmedical15, fortis15, testfor16-

usmedical15.java | fortis15.java | testfor16.java

```java
package com.day19;

public interface usmedical15 {

    int MINFEE=214234; //by default public static final.

    //in interface we can have default method.
    //but it has to have body.

    //create default method.
    default int getdefault(String name) {
        System.out.println("first default from interface" + name);
        return 10;
    }


//n number of default and static methods allowed.
    default void getdefault1(String name) {
        System.out.println("second default from interface" + name);
    }

    //Illegal modifier for the interface method getstatic1; only public, private, abstract, default, static and strictfp are permitted
//  public static final int getstatic1(String name) {
//      System.out.println("third default from interface" + name);
//      return 10;
//  }
//      return 10;
//  }

    public static void getstatic2(String name) {
        System.out.println("first static from interface" + name);
    }

    public static int getstatic1(String name) {
        System.out.println("second static from interface" + name);
        return 10;
    }
}

```

fortis15.java testfor16.java

```java
package com.day19;

//can implement multiple interface.
public class fortis15 implements usmedical15{

    int MINFEE=654657567;

    //change default to another access specifier to use it.
    @Override
    public int getdefault(String name) {
        System.out.println("first default from child class" + name);
        return 10;
    }


//  protected int getdefault(String name) {
//      System.out.println("first default from interface" + name);
//      return 10;
//  }

    //Cannot reduce the visibility of the inherited method from usmedical15
    //protected not allowed.
//  @Override
//  protected void getdefault1(String name) {
//      System.out.println("second default from interface" + name);
//  }
```

```java
//      System.out.println("second default from interface" + name);
//  }

    //Cannot reduce the visibility of the inherited method from usmedical15
    //cannot have private.
//  @Override
//  private void getdefault1(String name) {
//      System.out.println("second default from interface" + name);
//  }

    //only public allowed.
    @Override
    public void getdefault1(String name) {
        System.out.println("second default from child class" + name);
    }
```

```java
        System.out.println("second default from child class" + name);
    }

//Cannot reduce the visibility of the inherited method from usmedical14
//  default void getdefault1(String name) {
//      System.out.println("second default from interface" + name);
//  }

    //The method getstatic2(String) of type fortis14 must override or implement a supertype method
    //cannot use override with static.
//  @Override
//  public static void getstatic2(String name) {
//      System.out.println("first static from interface" + name);
//  }

    //without override works, method hiding.
    public static void getstatic2(String name) {
        System.out.println("first static from child class" + name);
    }
```

```
55          System.out.println("first static from child class" + name);
56      }
57
58      //The method getstatic1(String) of type fortis14 must override or implement a supertype method.
59      //cannot use override with static.
60  //  @Override
61  //  public static int getstatic1(String name) {
62  //      System.out.println("second static from interface" + name);
63  //      return 10;
64  //  }
65
66
67      public static int getstatic1(String name) {
68          System.out.println("second static from child class" + name);
69          return 10;
70      }
71
72
73
74
75  }
76
```

testfor16.java

```
1  package com.day19;
2
3  public class testfor16 {
4
5      public static void main(String[] args) {
6
7          fortis15 f1=new fortis15();
8          int v1=f1.MINFEE;
9          System.out.println(v1);
10
11         int v2=f1.getdefault("karan");
12         System.out.println(v2);
13
14         f1.getdefault1("pawan");
15
16         int v3=f1.getstatic1("ranjani");
17         //warning.
18         //The static method getstatic1(String) from the type fortis15 should be accessed in a static
   way
19         System.out.println(v3);
20
21         int v4=fortis15.getstatic1("poland");
22         System.out.println(v4);
23
24         f1.getstatic2("uk");
25         //warning.
26         //The static method getstatic2(String) from the type fortis15 should be accessed in a static
   way
```

```
25         //warning.
26         //The static method getstatic2(String) from the type fortis15 should be accessed in a static
   way
27
28         fortis15.getstatic2("germany");
29
30         //Cannot instantiate the type usmedical15
31  //       usmedical15 u1=new usmedical15();
32
33         usmedical15 u1=new fortis15();
34         int b1=u1.getdefault("canada");
35         System.out.println(b1);
36
37         u1.getdefault1("austalisa");
38
39         int b2=u1.MINFEE;
40         System.out.println(b2);
41
42         //cannot access static from interface using objects.
43
44         //using interface name we can access.
45         int c1=usmedical15.getstatic1("hungary");
46         System.out.println(c1);
```

```
        //using interface name we can access:
        int c1=usmedical15.getstatic1("hungary");
        System.out.println(c1);

        usmedical15.getstatic2("budapest");


    }

}
```

```
    }

}

//654657567
//first default from child classkaran
//10
//second default from child classpawan
//second static from child classranjani
//10
//second static from child classpoland
//10
//first static from child classuk
//first static from child classgermany
//first default from child classcanada
//10
//second default from child classaustalisa
//214234
//second static from interfacehungary
//10
//first static from interfacebudapest
```

Us medical-

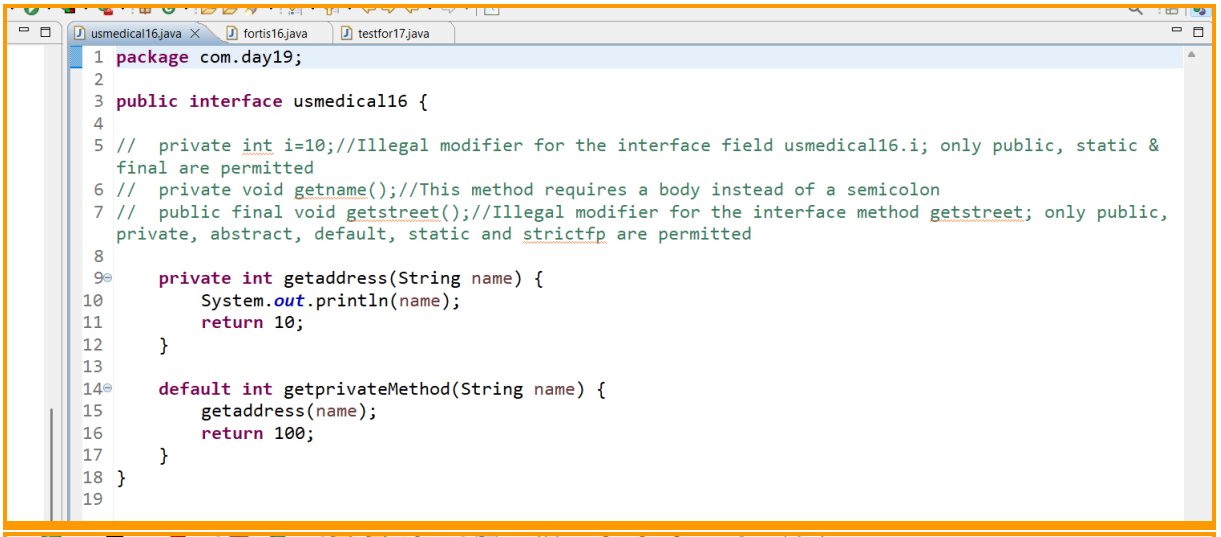
Private methods not allowed in interfaces, error.

```
23      //private non static method: abstract?: not allowed
24      //private void freeMedical();
25
```

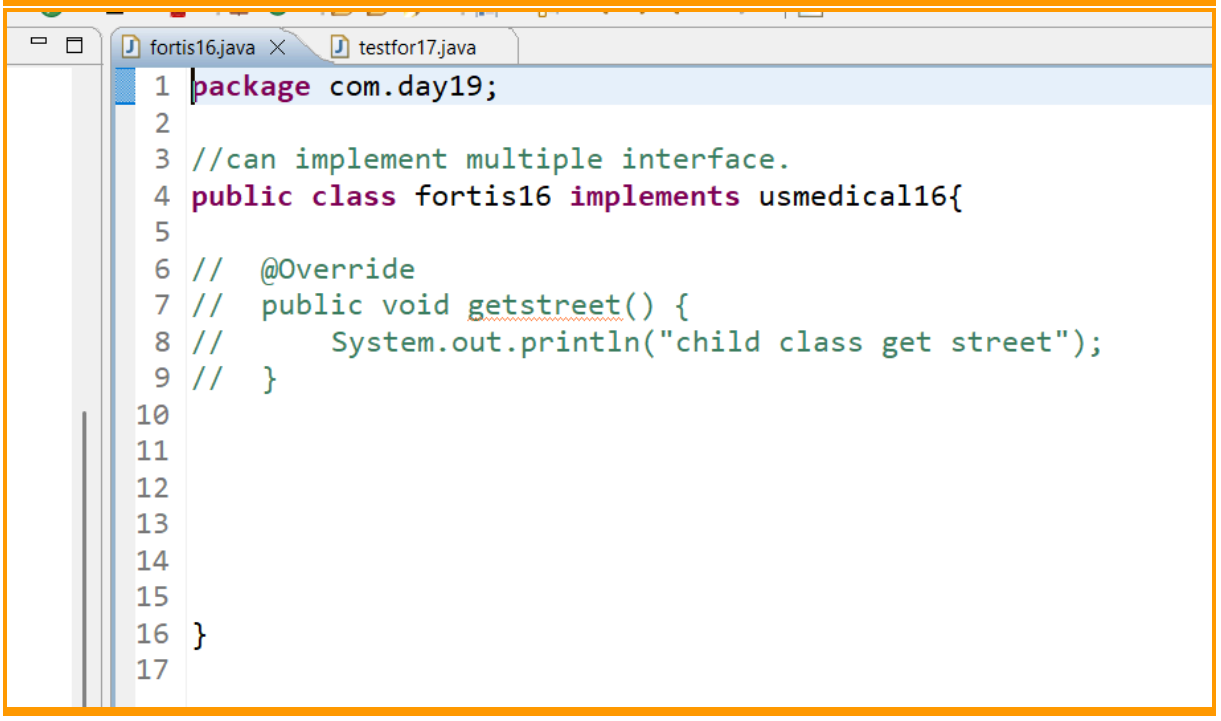
//This method requires a body instead of a semicolon

Private method with body allowed-

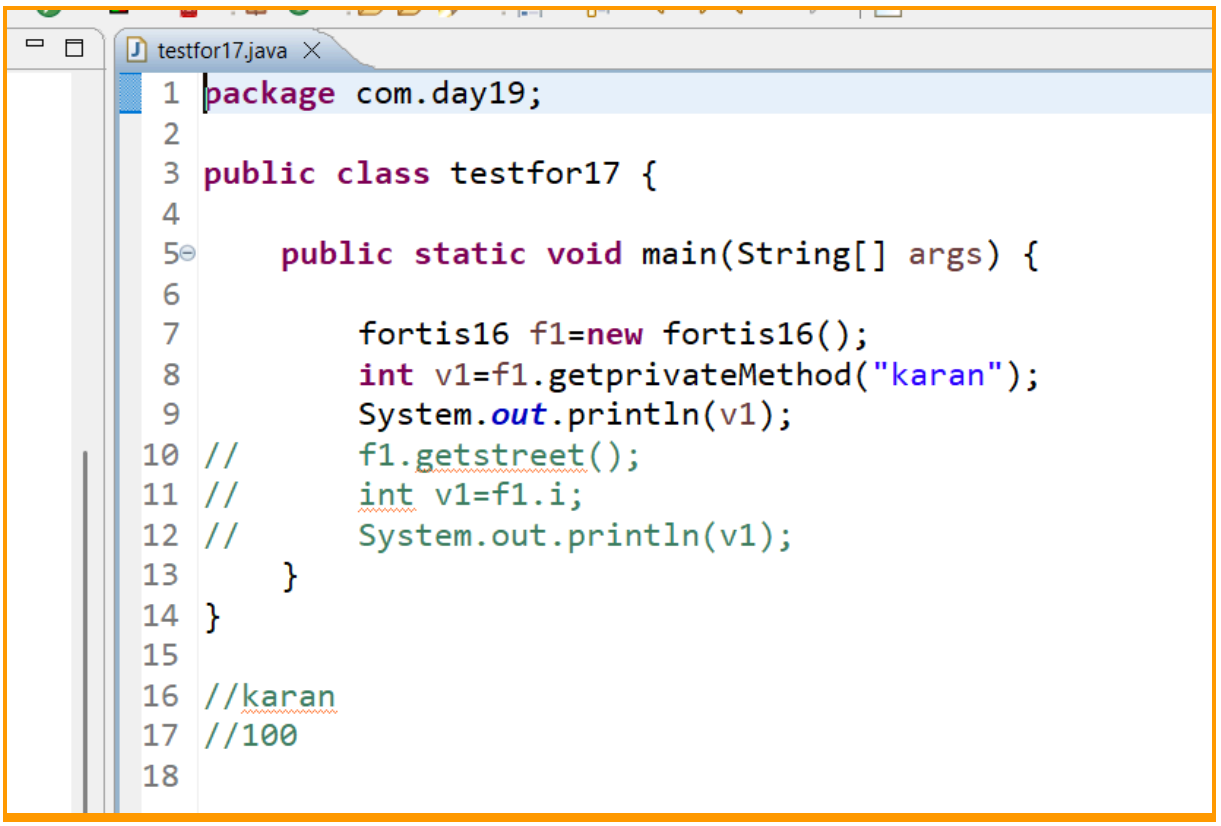
Paste usmedical16, fortis16, testfor17 -



```java
package com.day19;

public interface usmedical16 {

//  private int i=10;//Illegal modifier for the interface field usmedical16.i; only public, static & final are permitted
//  private void getname();//This method requires a body instead of a semicolon
//  public final void getstreet();//Illegal modifier for the interface method getstreet; only public, private, abstract, default, static and strictfp are permitted

    private int getaddress(String name) {
        System.out.println(name);
        return 10;
    }

    default int getprivateMethod(String name) {
        getaddress(name);
        return 100;
    }
}

```



```java
package com.day19;

//can implement multiple interface.
public class fortis16 implements usmedical16{

//  @Override
//  public void getstreet() {
//      System.out.println("child class get street");
//  }



}

```

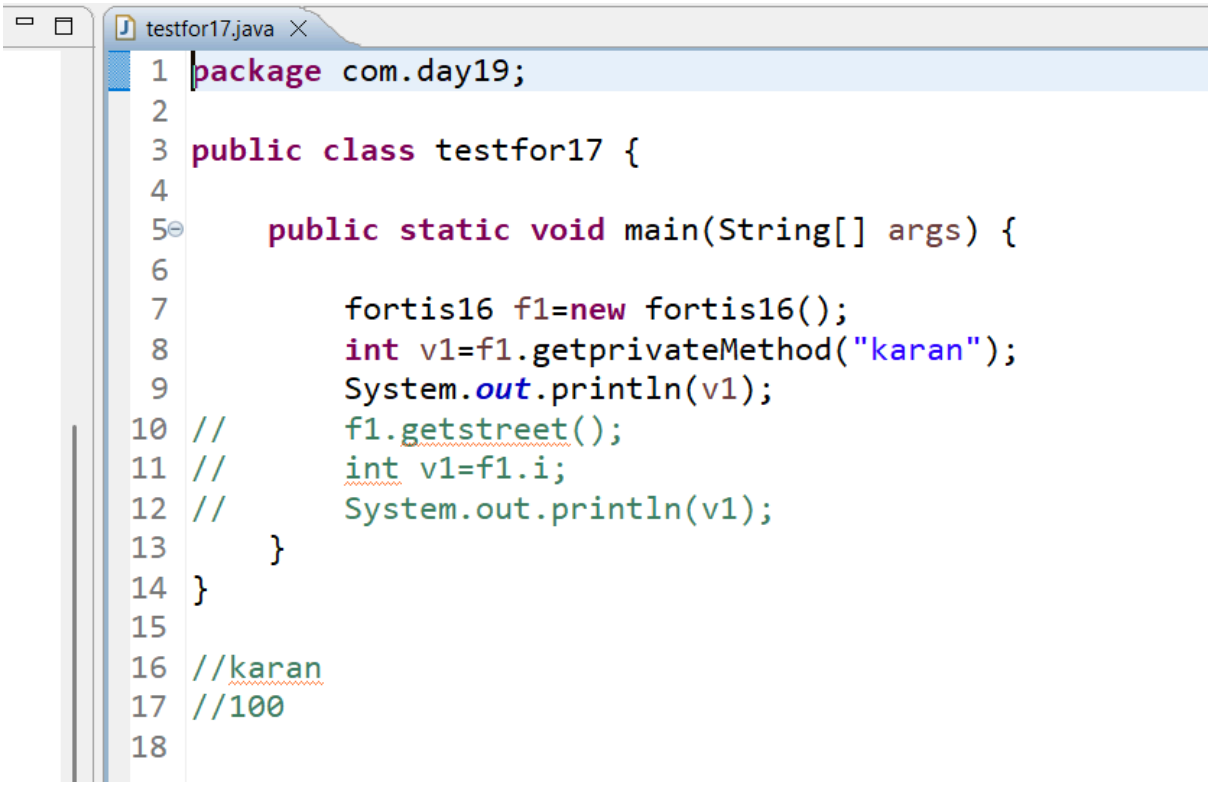

testfor17.java
1 package com.day19;
2
3 public class testfor17 {
4
5 public static void main(String[] args) {
6
7 fortis16 f1=new fortis16();
8 int v1=f1.getprivateMethod("karan");
9 System.out.println(v1);
10 // f1.getstreet();
11 // int v1=f1.i;
12 // System.out.println(v1);
13 }
14 }
15
16 //karan
17 //100
18

usmedical16.java | fortis16.java | testfor17.java

```java
package com.day19;

public interface usmedical16 {

//  private int i=10;//Illegal modifier for the interface field usmedical16.i; only public, static & final are permitted
//  private void getname();//This method requires a body instead of a semicolon
//  public final void getstreet();//Illegal modifier for the interface method getstreet; only public, private, abstract, default, static and strictfp are permitted

    private int getaddress(String name) {
        System.out.println(name);
        return 10;
    }

    default int getprivateMethod(String name) {
        getaddress(name);
        return 100;
    }
}
```

fortis16.java | testfor17.java

```java
package com.day19;

//can implement multiple interface.
public class fortis16 implements usmedical16{

//  @Override
//  public void getstreet() {
//      System.out.println("child class get street");
//  }



}
```



Final method not allowed in interfaces, error-

Us medical.

```
25
26      //final method: abstract?: not allowed
27      //public final void medicalResults();
28
```

//Illegal modifier for the interface method getstreet; only public, private, abstract, default, static and strictfp are permitted

Calling default in fortis-

inheritance concept.

```
66
67      public void pathalogyServices() {
68          System.out.println("Fh -- pathalogyServices");
69          medicalInsurance();
70
71      }
```

Paste usmedical17, fortis17, testfor18-

usmedical17.java | fortis17.java | testfor18.java

```java
package com.day19;

public interface usmedical17 {

    private int getaddress(String name) {
        System.out.println(name);
        return 10;
    }

    default int getprivateMethod(String name) {
        getaddress(name);
        return 100;
    }
}

```

fortis17.java | testfor18.java

```java
package com.day19;

//can implement multiple interface.
public class fortis17 implements usmedical17{


    public void pathservices(String name) {
        System.out.println("pathology services from fortis " +name);
        //another way to call default in child class.
        //directly call the method name.
        getprivateMethod(name);
    }

}

```

testfor18.java

```java
package com.day19;

public class testfor18 {

    public static void main(String[] args) {

        fortis17 f1=new fortis17();
        int v1=f1.getprivateMethod("karan");
        System.out.println(v1);

        f1.pathservices("jigar");

        //Cannot instantiate the type usmedical17
//      usmedical17 u1=new usmedical17();
        usmedical17 u1=new fortis17();
        int v2=u1.getprivateMethod("jumbo");
        System.out.println(v2);

    }

}

//karan
//100
//pathology services from fortis jigar
//jigar
//jumbo
//100
```

Paste usmedical18, fortis18-

usmedical18.java

```java
package com.day19;

public interface usmedical18 {

    private int getaddress(String name) {
        System.out.println(name);
        return 10;
    }

    default int getprivateMethod(String name) {
        getaddress(name);
        return 100;
    }
}

```

usmedical18.java | fortis18.java

```java
package com.day19;

//can implement multiple interface.
public class fortis18 implements usmedical18{


    //cannot have method inside method in interface.
    //we have to define the inner method also in class.
    //else error.
//  @Override
//  public int getprivateMethod(String name) {
//      getaddress(name);//The method getaddress(String) is undefined for the type fortis18
//      return 100;
//  }

}

```

If you override default in fortis then fortis default is called-

```
66
67      public void pathalogyServices() {
68          System.out.println("Fh -- pathalogyServices");
69          medicalInsurance();
70
71      }
72
73      //common
74      @Override
75      public void emergencyServices() {
76          System.out.println("Fh -- emergencyServices");
77      }
78
79
80      //method hiding
81      public static void billing() {
82          System.out.println("FH -- billing");
83
84      }
85
86
87      //trying to override default method of interface with public:
88      @Override
89      public void medicalInsurance() {
90          System.out.println("FH -- medical insurance");
91      }
92
```

Paste usmedical19, testfor20, fortis19 -

usmedical19.java | fortis19.java | testfor20.java

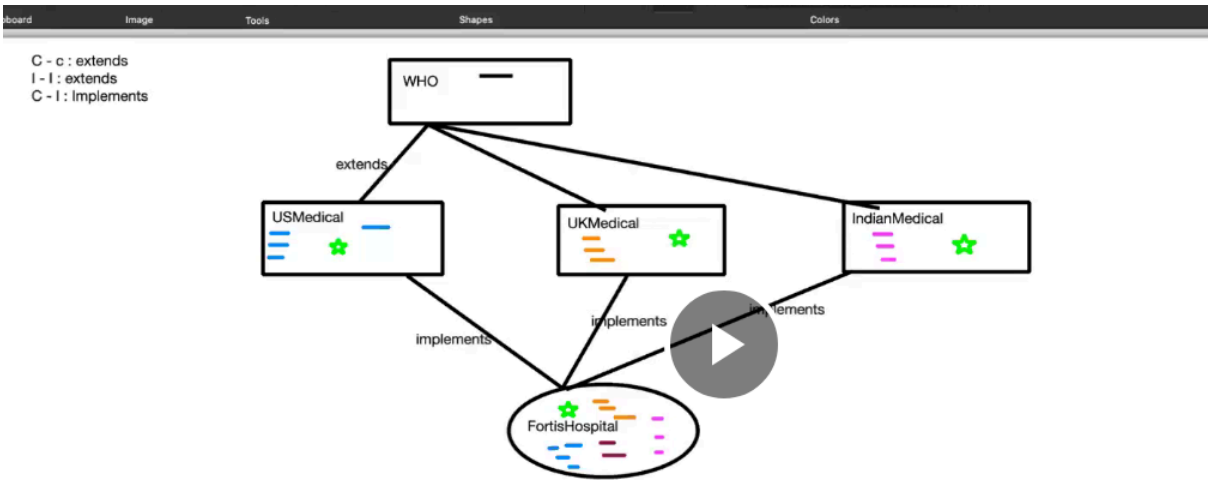
```java
package com.day19;

public interface usmedical19 {

    default int getprivateMethod(String name) {
        System.out.println(name);
        return 100;
    }
}

```

fortis19.java | testfor20.java

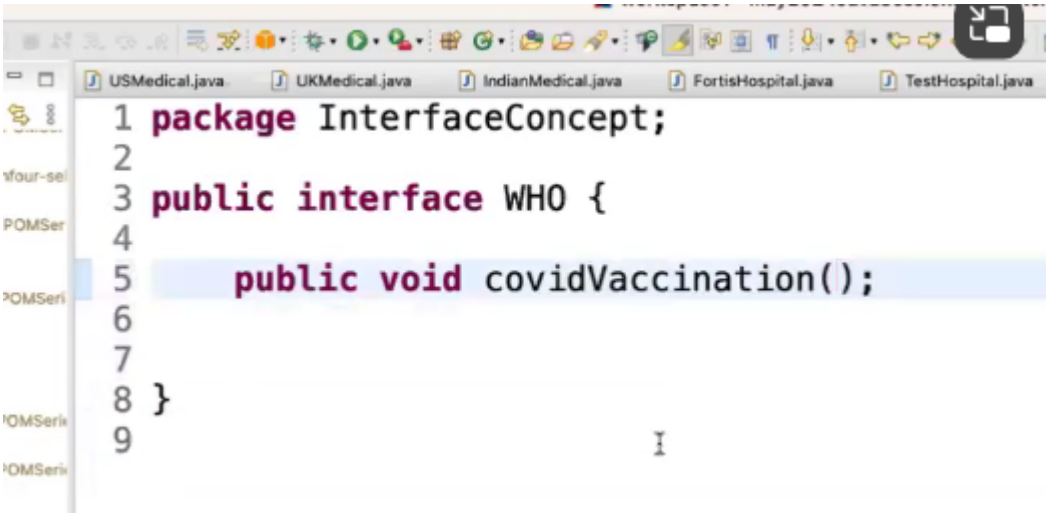
```java
package com.day19;

//can implement multiple interface.
public class fortis19 implements usmedical19{

@Override
public int getprivateMethod(String name) {
    System.out.println(name + " " + " from fortis");
    return 100;
}

}

```

testfor20.java

```java
package com.day19;

public class testfor20 {

    public static void main(String[] args) {

        fortis19 f1=new fortis19();
        int v1=f1.getprivateMethod("jogle");
        System.out.println(v1);

        //Cannot instantiate the type usmedical19
//      usmedical19 u1=new usmedical19();
//      int v2=u1.getprivateMethod("jumbo");
//      System.out.println(v2);

        usmedical19 u1=new fortis19();
        int v2=u1.getprivateMethod("jumbo karan");
        System.out.println(v2);

    }

}

//jogle  from fortis
//100
//jumbo karan  from fortis
//100
```



Who-



Us medical-

Fortis-

```
92
93      //WHO
94      @Override
95      public void covidVaccination() {
96          System.out.println("FH -- covidVaccination");
97      }
98
```

Test-

```
30
31          fh.covidVaccination();
```

Fh -covid.

Us medical-

Can access the who method.

```
45        us.covidVaccination();
```

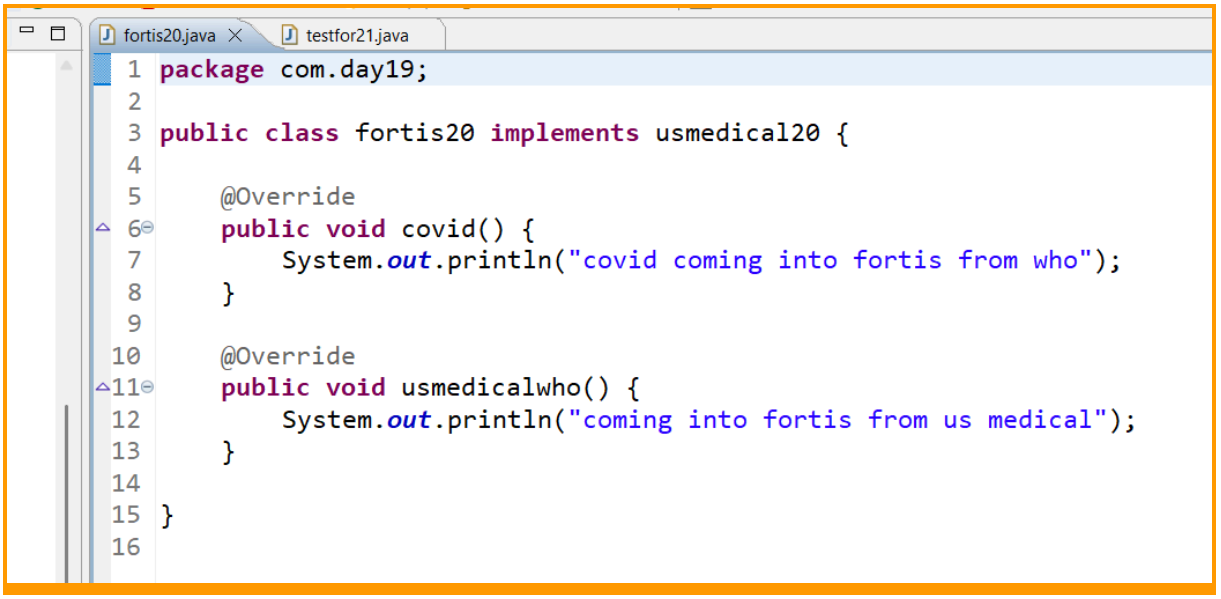
Paste who1, usmedical20, fortis20, testfor21-

who1.java

```
1  package com.day19;
2
3  public interface who1 {
4
5      public void covid();
6
7  }
8
```

usmedical20.java

```
1  package com.day19;
2
3  public interface usmedical20 extends who1{
4
5      public void usmedicalwho();
6
7  }
8
```



fortis20.java | testfor21.java

```
1  package com.day19;
2
3  public class fortis20 implements usmedical20 {
4
5      @Override
6      public void covid() {
7          System.out.println("covid coming into fortis from who");
8      }
9
10     @Override
11     public void usmedicalwho() {
12         System.out.println("coming into fortis from us medical");
13     }
14
15 }
16
```

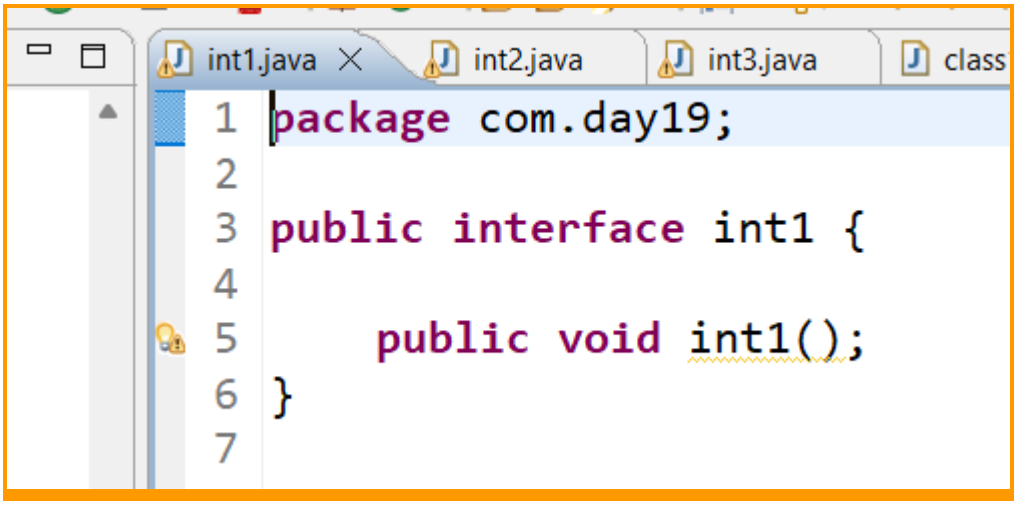
testfor21.java

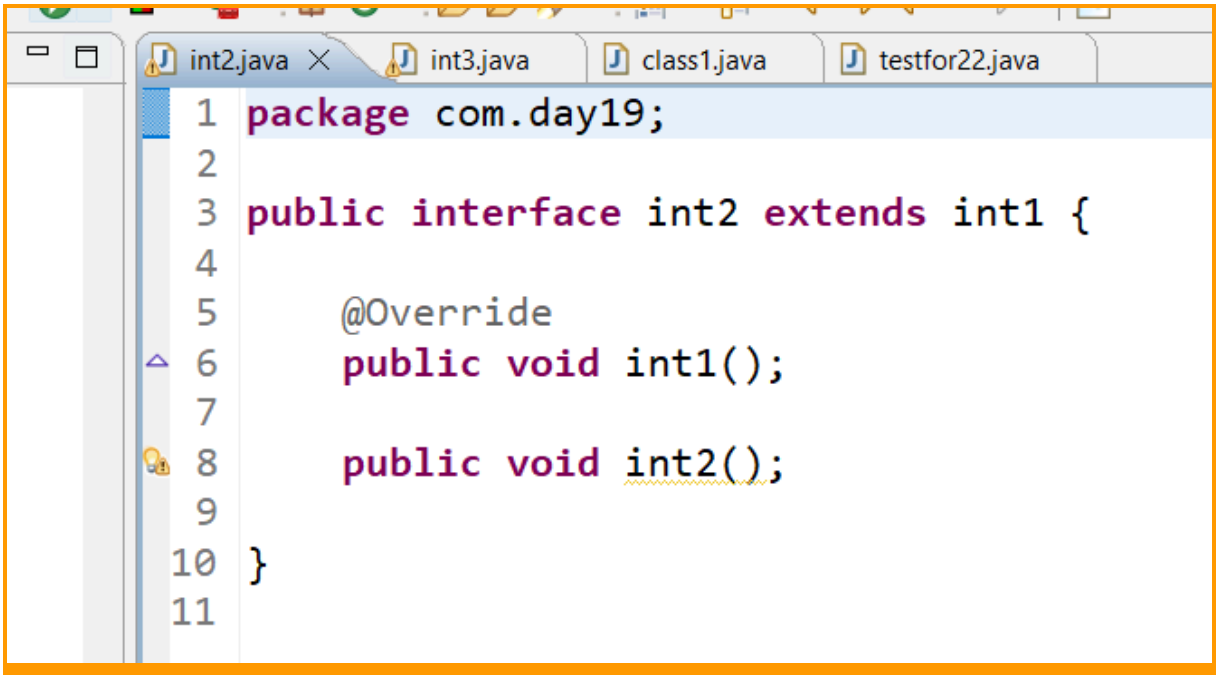
```java
package com.day19;

public class testfor21 {

    public static void main(String[] args) {

        fortis20 f1=new fortis20();
        f1.covid();
        f1.usmedicalwho();

        //Cannot instantiate the type usmedical20
//      usmedical20 u1=new usmedical20();

        usmedical20 u1=new fortis20();
        u1.covid();
        u1.usmedicalwho();

        //Cannot instantiate the type who1
//      who1 w1=new who1();
        who1 w1=new fortis20();
        w1.covid();

    }
```

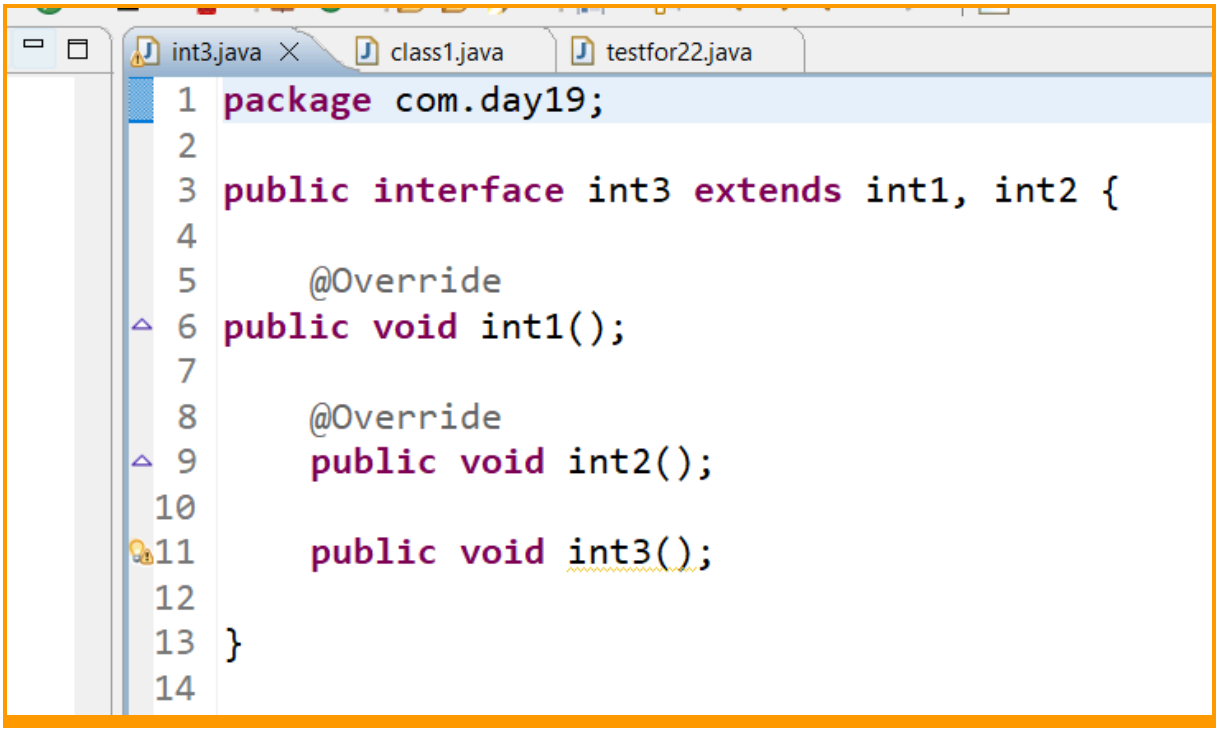
```java
    }

}
//covid coming into fortis from who
//coming into fortis from us medical
//covid coming into fortis from who
//coming into fortis from us medical
//covid coming into fortis from who
```

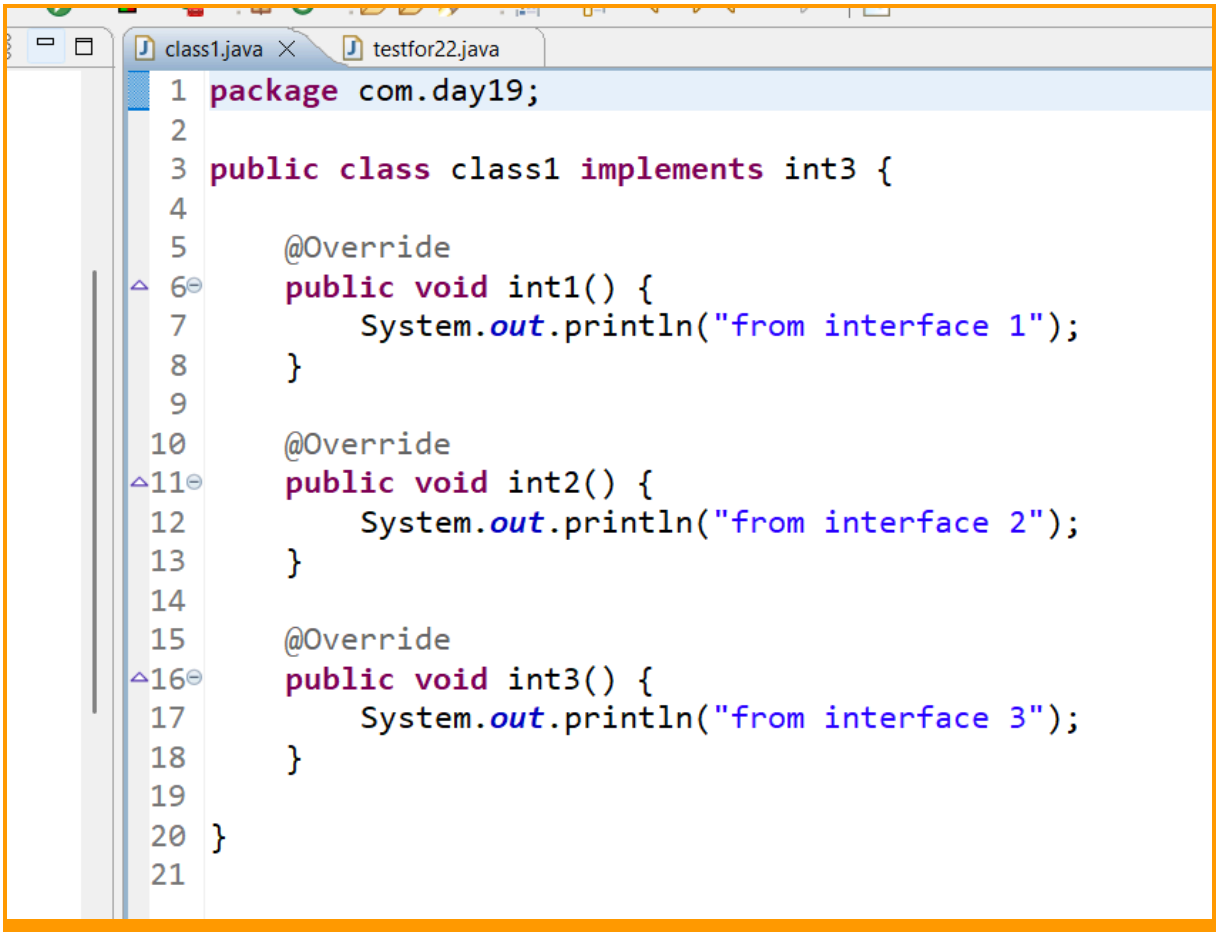
One interface can extend more than one other interface –

Paste int1, int2, int3, class1, testfor22-


int1.java
int2.java
int3.java
class
1 package com.day19;
2
3 public interface int1 {
4
5 public void int1();
6 }
7


int2.java
int3.java
class1.java
testfor22.java
1 package com.day19;
2
3 public interface int2 extends int1 {
4
5 @Override
6 public void int1();
7
8 public void int2();
9
10 }
11


int3.java
class1.java
testfor22.java
1 package com.day19;
2
3 public interface int3 extends int1, int2 {
4
5 @Override
6 public void int1();
7
8 @Override
9 public void int2();
10
11 public void int3();
12
13 }
14


class1.java
testfor22.java
1 package com.day19;
2
3 public class class1 implements int3 {
4
5 @Override
6 public void int1() {
7 System.out.println("from interface 1");
8 }
9
10 @Override
11 public void int2() {
12 System.out.println("from interface 2");
13 }
14
15 @Override
16 public void int3() {
17 System.out.println("from interface 3");
18 }
19
20 }
21

testfor22.java

```java
package com.day19;

public class testfor22 {

    public static void main(String[] args) {

        class1 c1=new class1();
        c1.int1();
        c1.int2();
        c1.int3();

//      int3 i3=new int3(); //Cannot instantiate the type int3
        int3 i3=new class1();
        i3.int1();
        i3.int2();
        i3.int3();

//      int2 i2=new int2(); //Cannot instantiate the type int2
        int2 i2=new class1();
        i2.int1();
        i2.int2();

//      int1 i1=new int1(); //Cannot instantiate the type int1
        int1 i1=new class1();
        i1.int1();

    }
```

```java
    }

}

//from interface 1
//from interface 2
//from interface 3
//from interface 1
//from interface 2
//from interface 3
//from interface 1
//from interface 2
//from interface 1
```

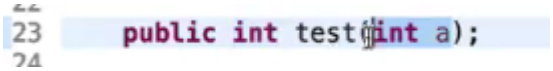
Multiple inheritance –

More than one parent, possible when classes and interfaces involved.

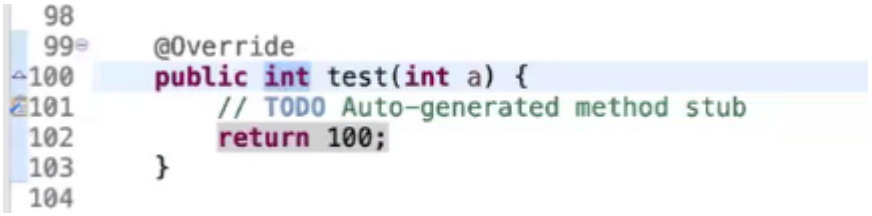
Multi level inheritance-

Grand parent to parent to child etc.

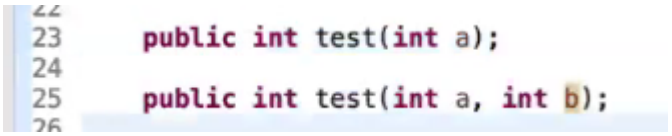
Us medical –



Fortis-



Us medical-



Fortis-

```
98
99      @Override
100     public int test(int a) {
101         // TODO Auto-generated method stub
102         return 100;
103     }
104
105     @Override
106     public int test(int a, int b) {
107         // TODO Auto-generated method stub
108         return 0;
109     }
110
111
```

Like this down cast also not allowed in test-

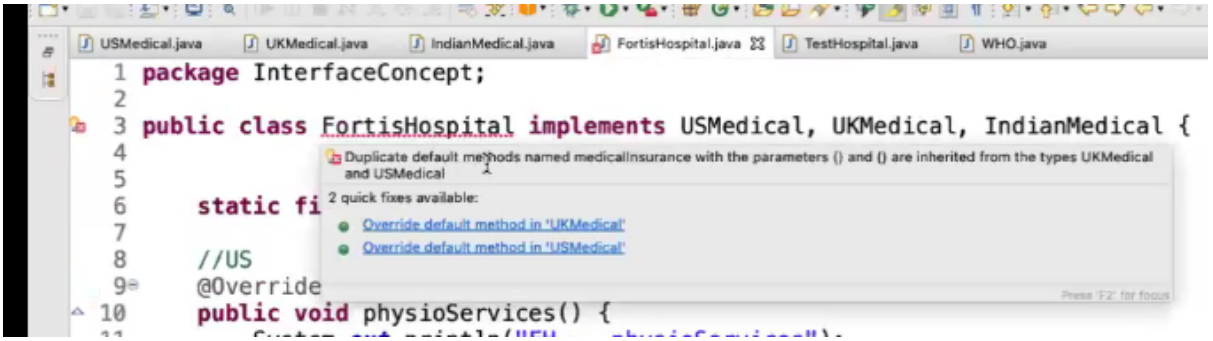
```
46
47          //down casting:
48          FortisHospital f1 = (FortisHospital) new USMedical();
49
```

Same default method in uk interface-

USMedical.java | UKMedical.java | IndianMedical.java | FortisHospital.java | TestHospital.java

```
1  package InterfaceConcept;
2
3  public interface UKMedical {
4
5      public void entServices();
6
7      public void pediaServices();
8
9      public void dermaServices();
10
11     public void emergencyServices();
12
13
14     public static void billing() {
15         System.out.println("UK Medical -- billing");
16     }
17
18
19     default void medicalInsurance() {
20         System.out.println("UK -- medical insurance");
21     }
22
23 }
24
```

Fortis-



Two default methods with same details cant be present in two interfaces which are overridden by a single child class.

Paste default1, default2, class2, testfor23-

default1.java | default2.java | class2.java | testfor23.java

```java
package com.day19;

public interface default1 {

    default int getdetails(String name) {
        System.out.println(name);
        return 100;
    }

}

```

default2.java | class2.java | testfor23.java

```java
package com.day19;

public interface default2 {

    default int getdetails(String name) {
        System.out.println(name);
        return 100;
    }

}

```

class2.java | testfor23.java

```java
package com.day19;

public class class2 implements default1, default2 {

    @Override
    public int getdetails(String name) {
        // TODO Auto-generated method stub
        return default1.super.getdetails(name);
    }
    //Duplicate default methods named getdetails with the parameters (String) and (String) are inherited from the types default2 and default1

    //at a time we can override from one interface.
    //if two interfaces have same default method implementation.

}

```

testfor23.java

```java
package com.day19;

public class testfor23 {

    public static void main(String[] args) {

        class2 c1=new class2();
        int v1=c1.getdetails("karan");
        System.out.println(v1);

//      default2 d2=new default2(); //Cannot instantiate the type default2

        default2 d2=new class2();
        int v3=d2.getdetails("default 2");
        System.out.println(v3);

//      default1 d1=new default1(); //Cannot instantiate the type default1

        default1 d1=new class2();
        int v4=d1.getdetails("jingostan");
        System.out.println(v4);
    }

}
```

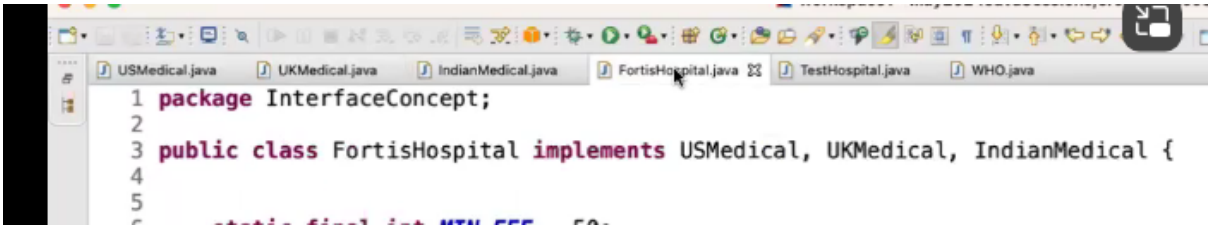
```
}

//karan
//100
//default 2
//100
//jingostan
//100
```

Default method can be present but with some parameter difference-

In two interfaces.

```
16      }
17
18
19      default void medicalInsurance(int a) {
20          System.out.println("UK -- medical insurance");
21      }
22
23  }
```

Child class also works fine.



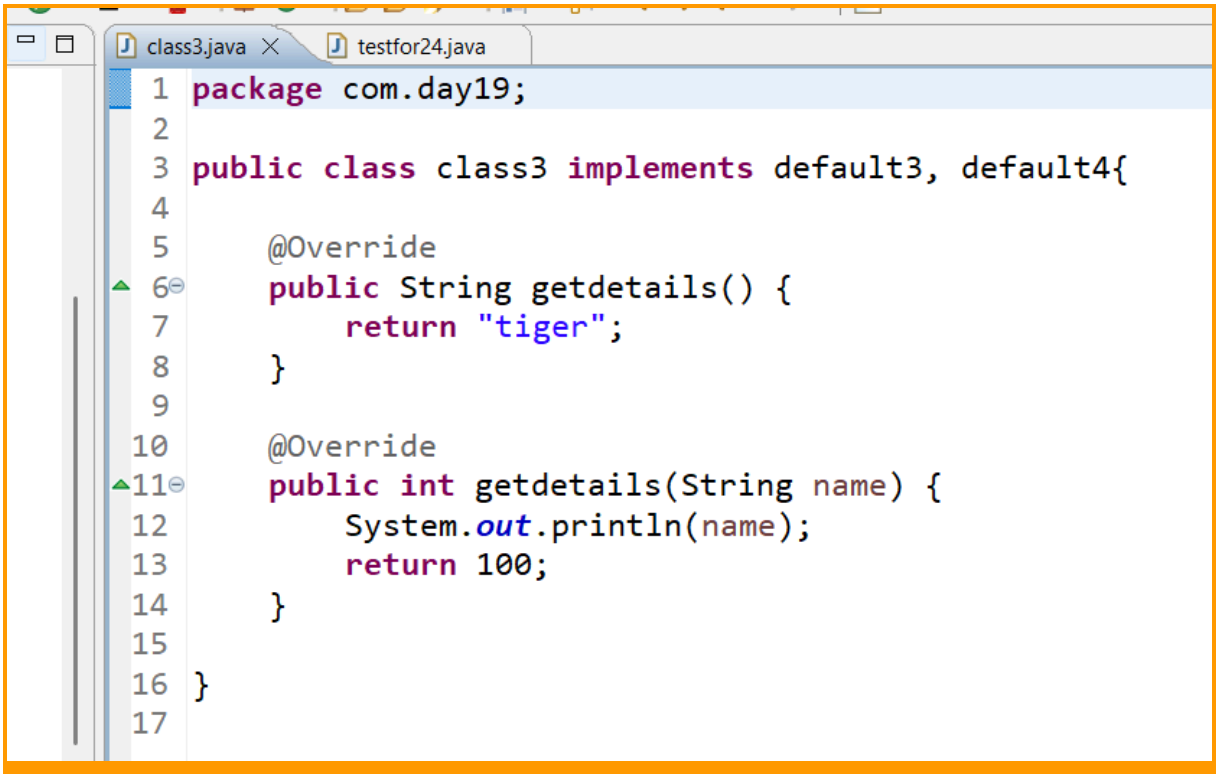
Paste default3, default4, class3, testfor24-

default3.java | default4.java | class3.java | testfor24.java

```java
package com.day19;

public interface default3 {

    default int getdetails(String name) {
        System.out.println(name);
        return 100;
    }

}

```

default4.java | class3.java | testfor24.java

```java
package com.day19;

public interface default4 {

    default String getdetails() {
        return "tiger";
    }

}

```


class3.java
testfor24.java
1 package com.day19;
2
3 public class class3 implements default3, default4{
4
5 @Override
6 public String getdetails() {
7 return "tiger";
8 }
9
10 @Override
11 public int getdetails(String name) {
12 System.out.println(name);
13 return 100;
14 }
15
16 }
17

testfor24.java

```java
package com.day19;

public class testfor24 {

    public static void main(String[] args) {

        class3 c1=new class3();
        String s1=c1.getdetails();
        System.out.println(s1);

        int v1=c1.getdetails("tiger");
        System.out.println(v1);

//      default4 d4=new default4();//Cannot instantiate the type default4
        default4 d4=new class3();
        String s2=d4.getdetails();
        System.out.println(s2);


//      default3 d3=new default3();//Cannot instantiate the type default3

        default3 d3=new class3();
        int v2=d3.getdetails("janadhan");
        System.out.println(v2);

    }
```

```java
        }

}

//tiger
//tiger
//100
//tiger
//janadhan
//100
```