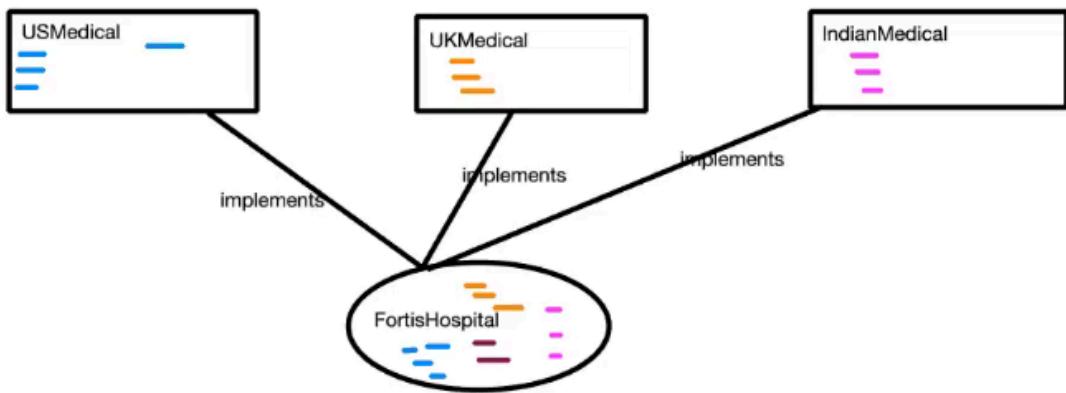


Interface-



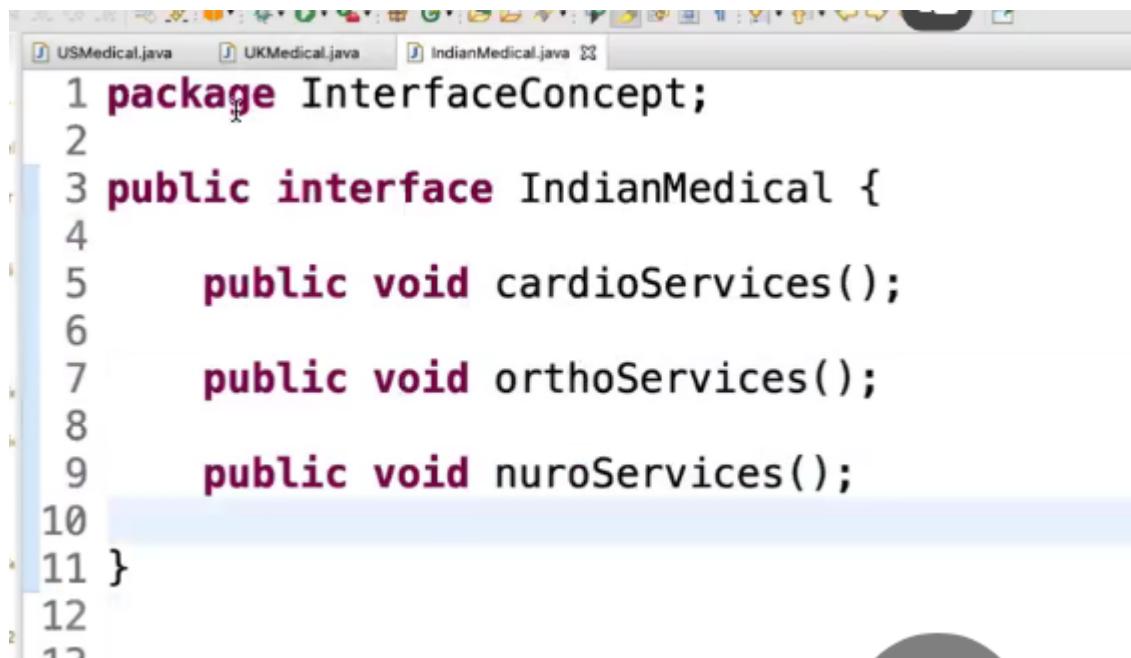
Interface will give error when trying to give method body- //Abstract methods do not specify a body

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



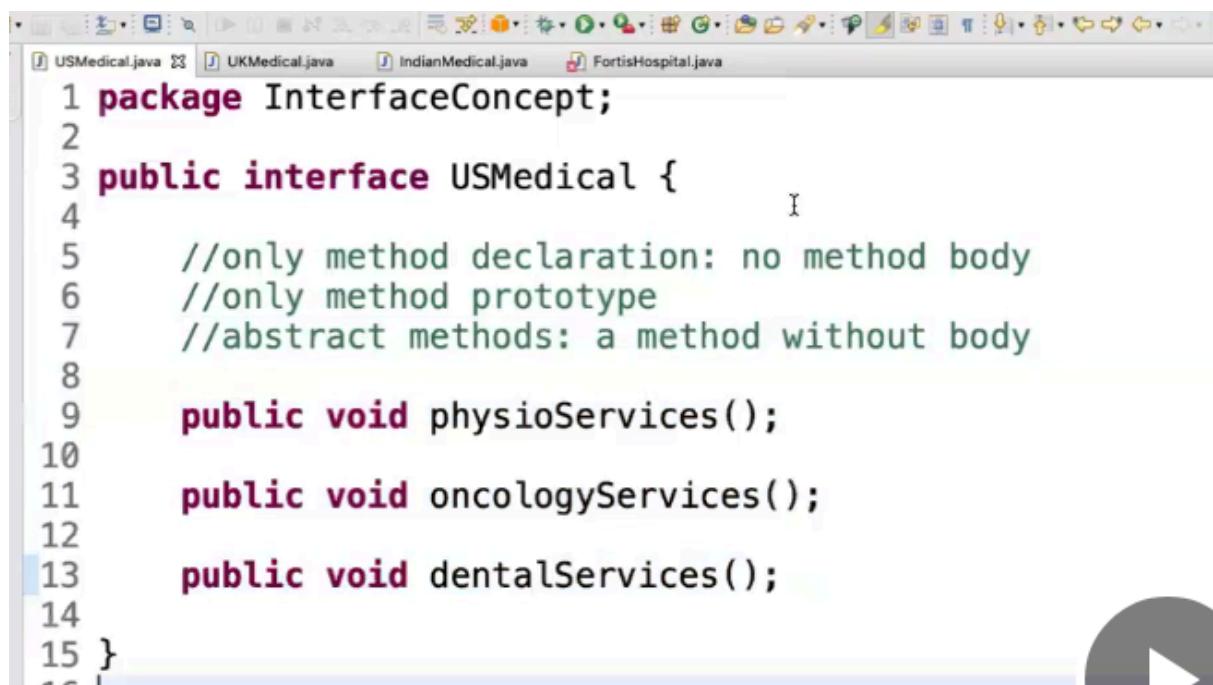
```
1 package InterfaceConcept;
2
3 public interface USMedical {
4
5     //only method declaration: no method body
6     //only method prototype
7
8     public void physioServices();
9
10    public void oncologyServices();
11
12    public void dentalServices();
13
14
15
16
17
18 }
19
```

```
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
```



```
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
13
```

Abstract methods-



```
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 }
```



```

1 package InterfaceConcept;
2
3 public class FortisHospital implements USMedical {
4
5     @Override
6     public void physioServices() {
7         System.out.println("FH -- physioServices");
8     }
9
10    @Override
11    public void oncologyServices() {
12        System.out.println("FH -- oncologyServices");
13    }
14
15    @Override
16    public void dentalServices() {
17        System.out.println("FH -- dentalServices");
18    }
19
20 }

21
22    //UK
23    @Override
24    public void entServices() {
25        System.out.println("FH -- entServices");
26    }
27
28    @Override
29    public void pediaServices() {
30        System.out.println("FH -- pediaServices");
31    }
32
33    @Override
34    public void dermaServices() {
35        System.out.println("FH -- dermaServices");
36    }
37
38
39
40 }

41
42    public class FortisHospital implements USMedical, UKMedical, IndianMedical {

```

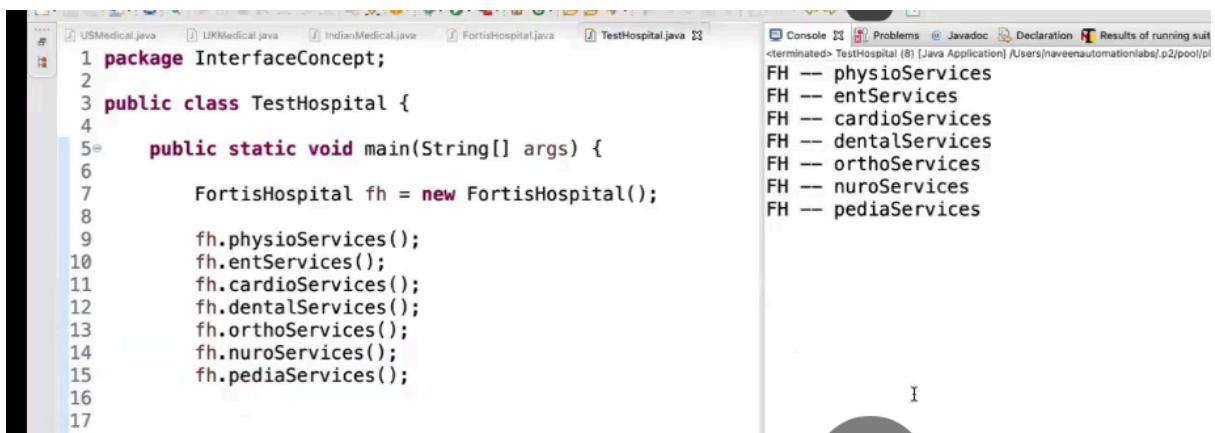
```

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 pathologyServices() {
65      System.out.println("Fh -- pathologyServices");
66
67  }
68

```

Call all methods-



The screenshot shows an IDE interface with multiple tabs at the top: USMedical.java, UKMedical.java, IndianMedical.java, FortisHospital.java, and TestHospital.java. The TestHospital.java tab is active, displaying the following code:

```

1 package InterfaceConcept;
2
3 public class TestHospital {
4
5     public static void main(String[] args) {
6
7         FortisHospital fh = new FortisHospital();
8
9         fh.physioServices();
10        fh.entServices();
11        fh.cardioServices();
12        fh.dentalServices();
13        fh.orthoServices();
14        fh.nuroServices();
15        fh.pediaServices();
16
17    }
18
19

```

To the right of the code, the IDE's console window shows the output of the program's execution:

```

FH -- physioServices
FH -- entServices
FH -- cardioServices
FH -- dentalServices
FH -- orthoServices
FH -- nuroServices
FH -- pediaServices

```

Call individual methods-

```

10
11
12
13
14
15
16
17     fh.medicalTraining();
18     fh.pathologyServices();
19

```

```

FH -- pediaServices
Fh -- medical training
Fh -- pathologyServices

```

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 -- pathologyServices
Fh -- emergencyServices

```

```

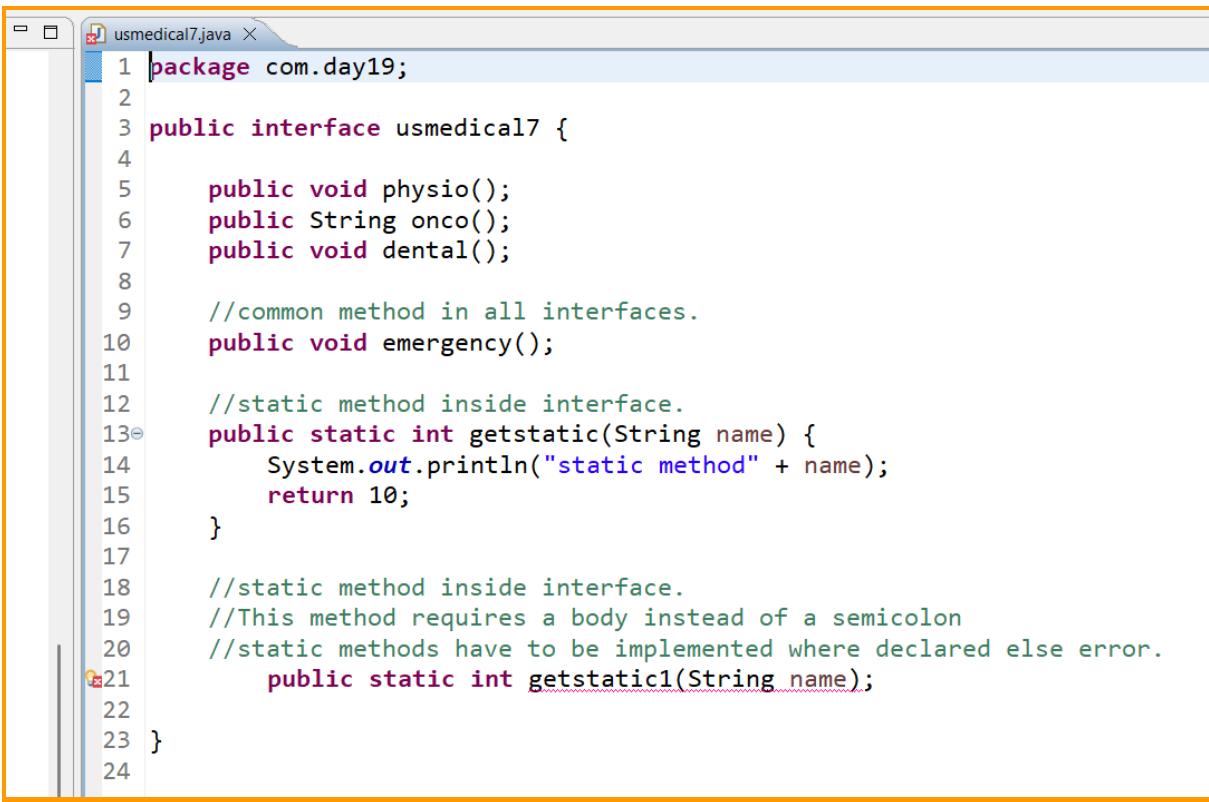
4
5     //only method declaration: no method body
6     //only method prototype
7     //abstract methods: a method without body
8     //can not create the object of interface
9

```

Paste usmedical3-

```
1 package com.day19;
2
3 public interface usmedical3 {
4
5     public void physio();
6     public String onco();
7     public void dental();
8
9     //common method in all interfaces.
10    public void emergency();
11
12    //static method inside interface.
13@    public static int getstatic(String name) {
14        System.out.println("static method" + name);
15        return 10;
16    }
17
18    //final method inside interface.
19    //Illegal modifier for the interface method getfinal; only public, private,
20    //abstract, default, static and strictfp are permitted
21 //   public final int getfinal(String name) {
22 //       System.out.println("static method" + name);
23 //       return 10;
24 //   }
25
26 //   return 10;
27 // }
28@    //Illegal modifier for the interface method getfinalandstatic;
29//only public, private, abstract, default, static and strictfp are permitted
30    public final static int getfinalandstatic(String name) {
31        System.out.println("static method" + name);
32        return 10;
33    }
34
35    //Illegal modifier for the interface method getfinal; only public,
36    //private, abstract, default, static and strictfp are permitted
37    public final int getfinal(String name);
38
39    //This method requires a body instead of a semicolon
40    public final static int getfinalandstatic(String name);
41
42 }
43 }
```

Paste usmedical7-

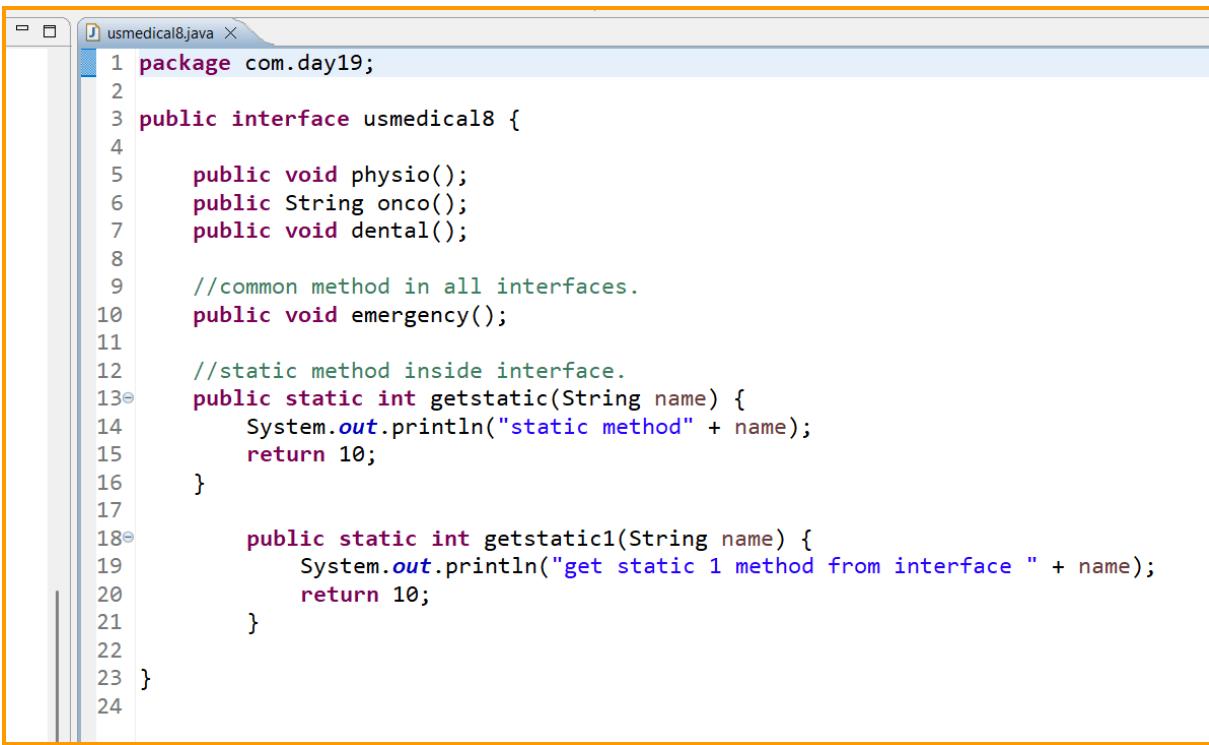


```

1 package com.day19;
2
3 public interface usmedical7 {
4
5     public void physio();
6     public String onco();
7     public void dental();
8
9     //common method in all interfaces.
10    public void emergency();
11
12    //static method inside interface.
13    public static int getstatic(String name) {
14        System.out.println("static method" + name);
15        return 10;
16    }
17
18    //static method inside interface.
19    //This method requires a body instead of a semicolon
20    //static methods have to be implemented where declared else error.
21    public static int getstatic1(String name);
22
23 }
24

```

Paste usmedical8, testfor8, fortis8-



```

1 package com.day19;
2
3 public interface usmedical8 {
4
5     public void physio();
6     public String onco();
7     public void dental();
8
9     //common method in all interfaces.
10    public void emergency();
11
12    //static method inside interface.
13    public static int getstatic(String name) {
14        System.out.println("static method" + name);
15        return 10;
16    }
17
18    public static int getstatic1(String name) {
19        System.out.println("get static 1 method from interface " + name);
20        return 10;
21    }
22
23 }
24

```

The screenshot shows a Java code editor with an orange border. At the top, there are two tabs: "usmedical8.java" and "fortis8.java". The code editor displays the following Java code:

```
1 package com.day19;
2
3 //can implement multiple interface.
4 public class fortis8 implements usmedical8{
5
6     @Override
7     public void physio() {
8         System.out.println("fortis physio");
9     }
10
11    @Override
12    public String onco() {
13        System.out.println("fortis onco");
14        return "tiger";
15    }
16
17    @Override
18    public void dental() {
19        System.out.println("fortis dental");
20    }
21
22    @Override
23    public void emergency() {
24        System.out.println("fortis emergency");
25    }
26
27    //not mandatory to override static method.
28    //if we write override, we get error.
29    //The method getstatic1(String) of type fortis8 must override or implement a supertype method
30 //    @Override
31 //    public static int getstatic1(String name) {
32 //        System.out.println("get static 1 method");
33 //        return 10;
34 //    }
35
36    public static int getstatic1(String name) {
37        System.out.println("get static 1 method from child class");
38        return 190;
39    }
40
41
42
43 }
44
```

The screenshot shows a Java code editor with three tabs at the top: usmedical8.java, fortis8.java, and testfor8.java. The testfor8.java tab is active, displaying the following code:

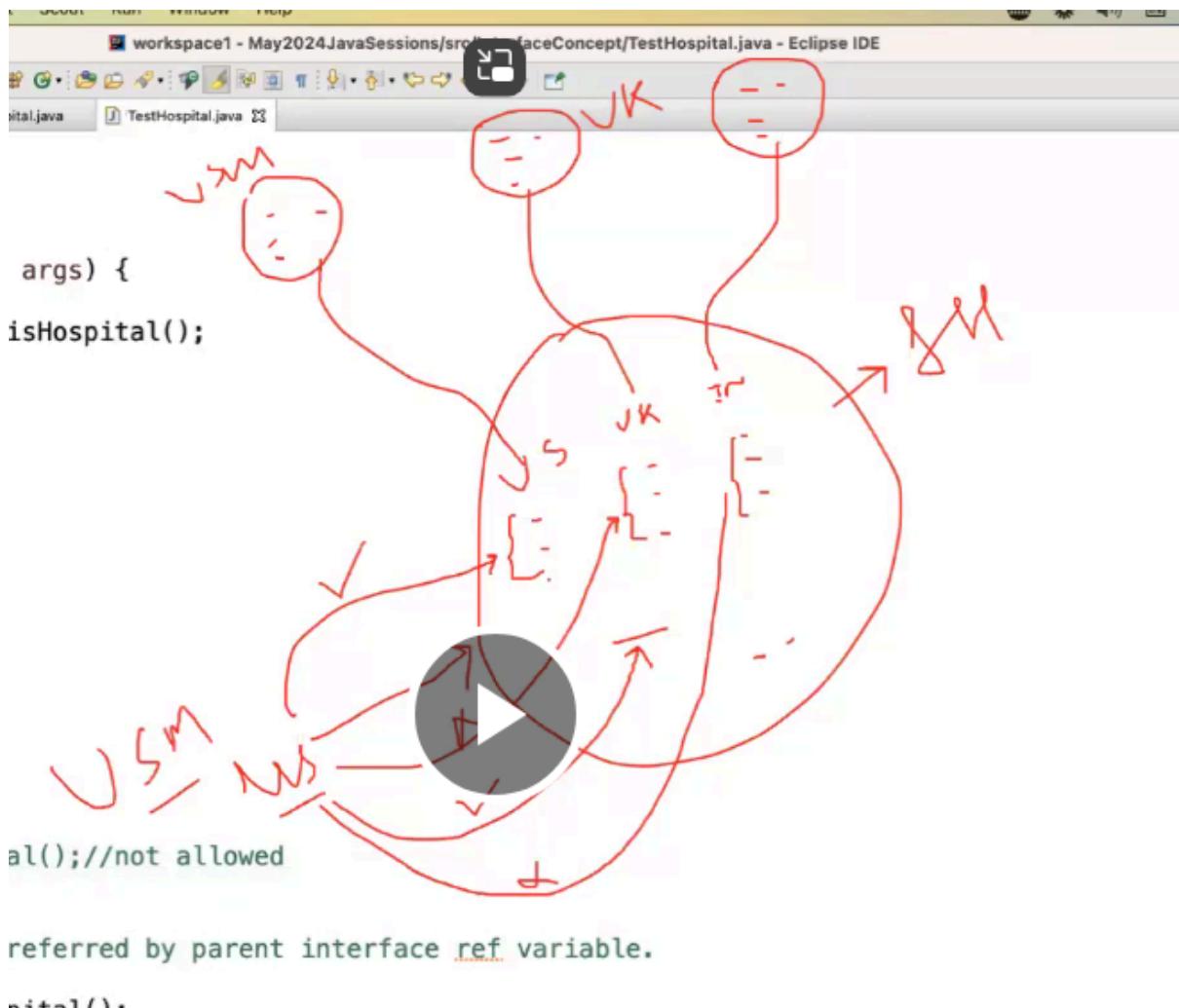
```
1 package com.day19;
2
3 public class testfor8 {
4
5     public static void main(String[] args) {
6
7         //create object of fortis8 and call all methods.
8         //capture all return values and print them also.
9
10        fortis8 f1=new fortis8();
11        f1.physio();
12        String o1=f1.onco();
13        System.out.println(o1);
14        f1.dental();
15        f1.emergency();
16        int i1=f1.getstatic1("karan");
17        //warning.
18        //The static method getstatic1(String) from the type fortis8
19        //should be accessed in a static way
20        System.out.println(i1);
21
22        //access with child class name.
23        int i2=fortis8.getstatic1("lion");
24        System.out.println(i2);
25
26        int i3=usmedical8.getstatic1("cheetah");
27        System.out.println(i3);
28    }
}
```

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



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-

```
usmedical10.java
1 package com.day19;
2
3 public interface usmedical10 {
4
5     public void physio();
6     public String onco();
7     public void dental();
8
9     //common method in all interfaces.
10    public void emergency();
11
12 }
13
```

```
ukmedical10.java
1 package com.day19;
2
3 public interface ukmedical10 {
4
5     public void ent();
6     public int pedia();
7     public void derma();
8
9     //common method in all interfaces.
10    public void emergency();
11
12 }
13
```

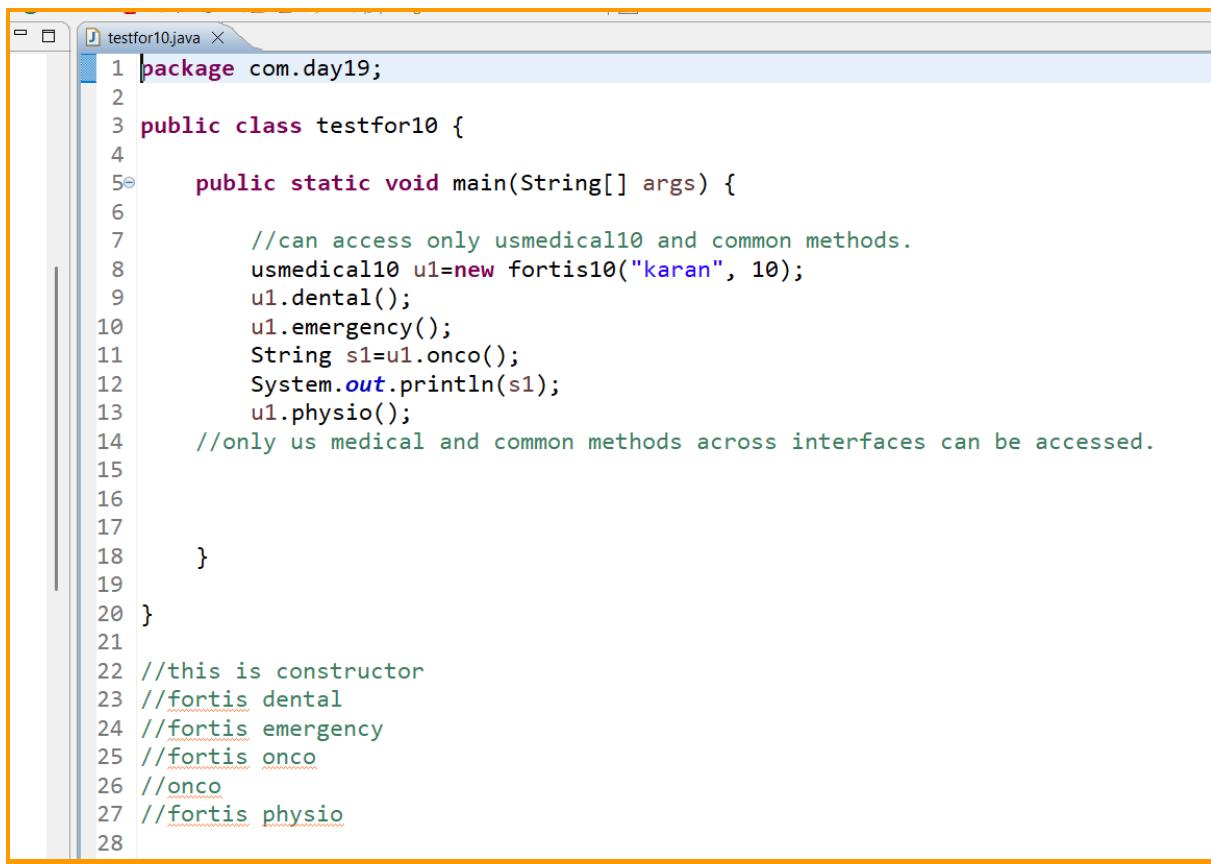
The screenshot shows an IDE interface with two code editors. The top editor contains the code for `indianmedical10.java`, and the bottom editor contains the code for `fortis10.java`. Both files are part of the same project, indicated by the tabs at the top.

```
indianmedical10.java content:
1 package com.day19;
2
3 public interface indianmedical10 {
4
5     public void cardio();
6     public String neuro();
7     public void ortho();
8
9     //common method in all interfaces.
10    public void emergency();
11
12 }
13
```

```
fortis10.java content:
1 package com.day19;
2
3 //can implement multiple interface.
4 public class fortis10 implements usmedical10, ukmedical10, indianmedical10{
5
6     String name;
7     int age;
8
9     public fortis10(String name, int age) {
10         this.name=name;
11         this.age=age;
12         System.out.println("this is constructor");
13     }
14
15     @Override
16     public void physio() {
17         System.out.println("fortis physio");
18     }
19
20
21     @Override
22     public String onco() {
23         System.out.println("fortis onco");
24         return "onco";
25     }
26
```

```
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() {
40         System.out.println("fortis pedia");
41         return 10;
42     }
43
44
45     @Override
△46⊕     public void derma() {
46         System.out.println("fortis derma");
47
48     }
49
50
51     @Override
△52⊕     public void cardio() {
52         System.out.println("fortis cardio");
53
54     }
55
56
57     @Override
△58⊕     public String neuro() {
58         System.out.println("fortis neuro");
59         return "a";
60     }
61
62
```

```
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     }
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
1 package com.day19;
2
3 public class testfor10 {
4
5     public static void main(String[] args) {
6
7         //can access only usmedical10 and common methods.
8         usmedical10 u1=new fortis10("karan", 10);
9         u1.dental();
10        u1.emergency();
11        String s1=u1.onco();
12        System.out.println(s1);
13        u1.physio();
14        //only us medical and common methods across interfaces can be accessed.
15
16
17    }
18
19 }
20
21 //this is constructor
22 //fortis dental
23 //fortis emergency
24 //fortis onco
25 //onco
26 //fortis physio
27
28
```

Downcasting not allowed-

```
33
34
35     //down casting:
36     //FortisHospital f1 = new USMedical();
```

Paste testfor11-

```
testfor11.java X
1 package com.day19;
2
3 public class testfor11 {
4
5     public static void main(String[] args) {
6
7         //when we create object of interface, it asks to implement them.
8         fortis10 f1=new usmedical10();
9
10        @Override
11        public void physio() {
12            // TODO Auto-generated method stub
13        }
14
15        @Override
16        public String onco() {
17            // TODO Auto-generated method stub
18            return null;
19        }
20
21        @Override
22        public void emergency() {
23            // TODO Auto-generated method stub
24
25        }
26
27
28        @Override
29        public void dental() {
30            // TODO Auto-generated method stub
31
32        }
33    };
34
35
36    }
37
38}
39
40
41 //this is constructor
42 //fortis dental
43 //fortis emergency
44 //fortis onco
45 //onco
46 //fortis physio
47
```

us medical interface-

we can have “n” number of static methods.

```

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

The screenshot shows a Java development environment with two code files open in separate tabs:

- usmedical11.java:** This file contains a single-line static method definition. It includes a note about static methods and a comment indicating that the method requires a body instead of a semicolon.

```
1 package com.day19;
2
3 public interface usmedical11 {
4
5     //in interface we can have static method but we have to define it.
6     //create one static method
7     public static String getstatic(String name) {
8         System.out.println("static method" + name);
9         return "tiger";
10    }
11
12    //if no definition then error-
13 // This method requires a body instead of a semicolon
14 // public static String getstatic1(String name);
15
16 }
17
```

- fortis11.java:** This file implements the `usmedical11` interface. It defines three methods: `physio()`, `onco()`, and `dental()`, each printing a specific string to `System.out`.

```
1 package com.day19;
2
3 //can implement multiple interface.
4 public class fortis11 implements usmedical11{
5
6     public void physio() {
7         System.out.println("physio");
8     }
9     public void onco() {
10        System.out.println("onco");
11    }
12     public void dental() {
13         System.out.println("dental");
14     }
15
16 }
17
```

```

1 package com.day19;
2
3 public class testfor12 {
4
5     public static void main(String[] args) {
6
7         String n1=usmedical11.getstatic("karan");
8         System.out.println(n1);
9
10        fortis11.usmedical11.getstatic("karan");
11        //because its not implemented in child class.
12    }
13
14 }
15

```

Same method of interface in fortis class-

```

76     //method hiding
77     public static void billing() {
78         System.out.println("FH -- billing");
79     }
80

```

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-

```

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

```

Paste usmedical12, fortis12, testfor13-

```
usmedical12.java
1 package com.day19;
2
3 public interface usmedical12 {
4
5     //in interface we can have static method but we have to define it.
6     //create one static method
7     public static String getstatic(String name) {
8         System.out.println("static method" + name);
9         return "tiger";
10    }
11
12    //if no definition then error-
13 // This method requires a body instead of a semicolon
14 // public static String getstatic1(String name);
15
16 }
17
```



```
fortis12.java
1 package com.day19;
2
3 //can implement multiple interface.
4 public class fortis12 implements usmedical12{
5
6     public void physio() {
7         System.out.println("physio");
8     }
9     public void onco() {
10        System.out.println("onco");
11    }
12     public void dental() {
13        System.out.println("dental");
14    }
15
16     //The method getstatic(String) of type fortis12 must override or implement a supertype method
17     //cannot use override for static method.
18 // @Override
19 // public static String getstatic(String name) {
20 //     System.out.println("static method" + name);
21 //     return "tiger";
22 // }
```

```

1 //      return "tiger";
2 // }
3
4 //without override it works.
5 //treated like another method.
6 public static String getstatic(String name) {
7 System.out.println("static child class method" + name);
8 return "lion";
9 }
10
11 }
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32

```

```

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
10

```

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();
```

Interface can also call-

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

Paste usmedical13, fortis13, testfor14-

```
usmedical13.java
1 package com.day19;
2
3 public interface usmedical13 {
4
5     //in interface we can have default method.
6     //but it has to have body.
7
8     //create default method.
9     default int getdefault(String name) {
10         System.out.println("default from interface method" + name);
11         return 10;
12     }
13
14
15     //without body error -
16     //This method requires a body instead of a semicolon
17 //    default int getdefault(String name);
18
19 }
20
```

```
fortis13.java
1 package com.day19;
2
3 //can implement multiple interface.
4 public class fortis13 implements usmedical13{
5
6     //cannot override default.
7     //Cannot reduce the visibility of the inherited method from usmedical13
8 //    @Override
9 //    default int getdefault(String name) {
10 //        System.out.println("default method" + name);
11 //        return 10;
12 //    }
13
14
15 }
```

```

1 package com.day19;
2
3 public class testfor14 {
4
5     public static void main(String[] args) {
6
7         fortis13 f1=new fortis13();
8         int v1=f1.getdefault("tiger");
9         System.out.println(v1);
10
11        //Cannot instantiate the type usmedical13
12        //usmedical13 u1=new usmedical13();
13        //int v2=u1.getdefault("gorilla");
14        //System.out.println(v2);
15
16
17        usmedical13 u1=new fortis13();
18        int v2=u1.getdefault("gorilla");
19        System.out.println(v2);
20    }
21
22 }
23
24 //default from interface methodtiger
25 //10
26 //default from interface methodgorilla
27 //10
28

```

How to override default method in class-
Fortis class.

```

81
82     //trying to override default method of interface with public:
83     @Override
84     public void medicalInsurance() {
85         System.out.println("US -- medical insurance");
86     }
87

```

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-

```

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

```

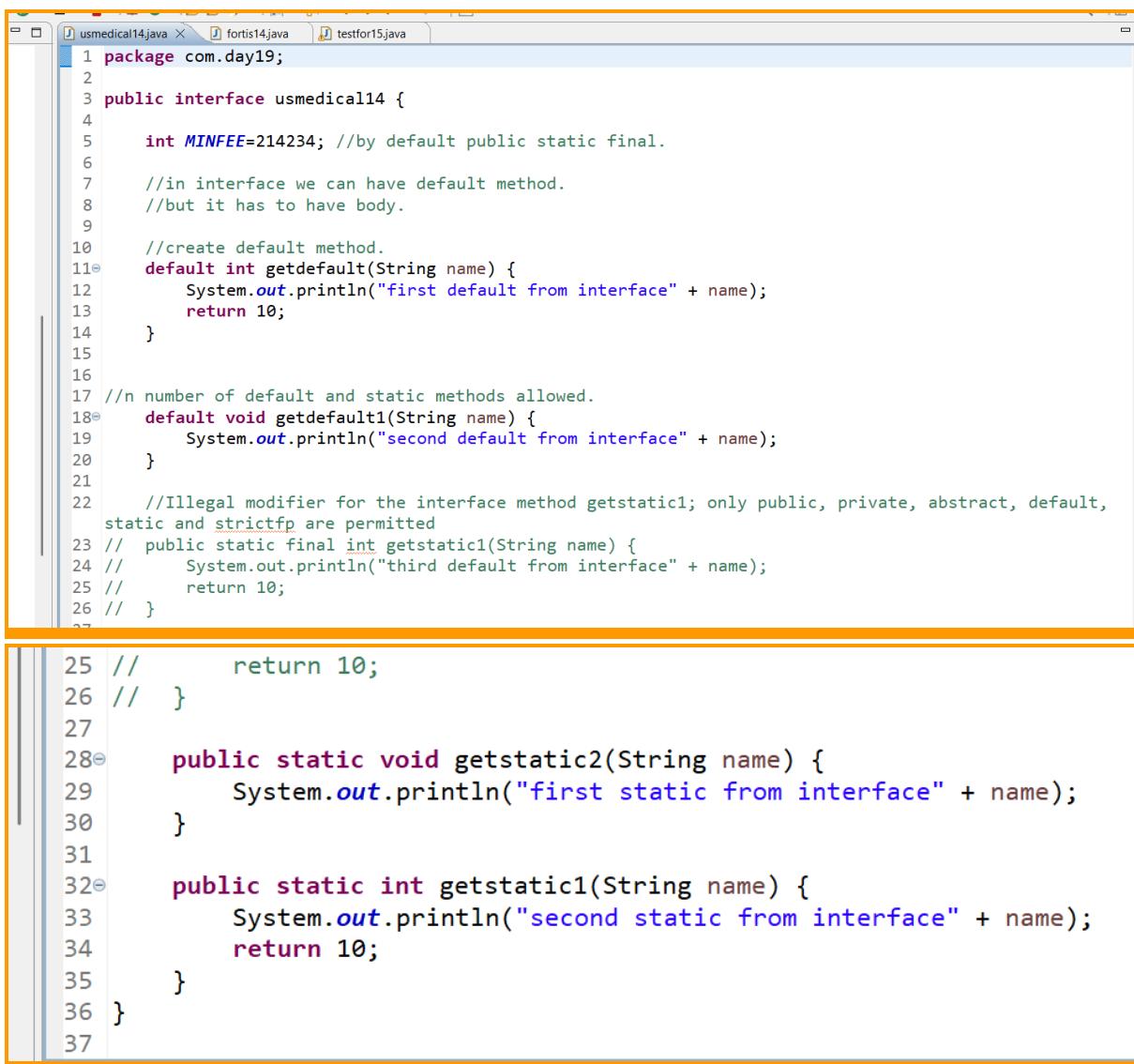
Test-

```

25
26     System.out.println(USMedical.MIN_FEE); //10
27     //USMedical.MIN_FEE = 100;
28
29     System.out.println(FortisHospital.MIN_FEE); //50
30

```

Paste usmedical14, fortis14, testfor15-



```

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,
23    // static and strictfp are permitted
24 //    public static final int getstatic1(String name) {
25 //        System.out.println("third default from interface" + name);
26 //        return 10;
27 //    }
28
29
30
31
32    public static void getstatic2(String name) {
33        System.out.println("first static from interface" + name);
34    }
35
36    public static int getstatic1(String name) {
37        System.out.println("second static from interface" + name);
38        return 10;
39    }
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
317
318
319
319
320
321
322
323
324
325
326
327
327
328
329
329
330
331
332
333
334
335
336
337
337
338
339
339
340
341
342
343
344
345
346
347
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
367
368
369
369
370
371
372
373
374
375
376
377
377
378
379
379
380
381
382
383
384
385
386
387
387
388
389
389
390
391
392
393
394
395
396
397
397
398
399
399
400
401
402
403
404
405
406
407
407
408
409
409
410
411
412
413
414
415
415
416
417
417
418
419
419
420
421
422
423
424
425
425
426
427
427
428
429
429
430
431
432
433
434
435
435
436
437
437
438
439
439
440
441
442
443
444
445
445
446
447
447
448
449
449
450
451
452
453
454
455
455
456
457
457
458
459
459
460
461
462
463
464
464
465
466
466
467
468
468
469
469
470
471
472
473
474
474
475
476
476
477
478
478
479
479
480
481
482
483
483
484
485
485
486
487
487
488
488
489
489
490
491
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452

```

```
1 package com.day19;
2
3 //can implement multiple interface.
4 public class fortis14 implements usmedical14{
5
6     int MINFEE=654657567;
7
8     //Cannot reduce the visibility of the inherited method from usmedical14
9 //    @Override
10 //    default int getdefault(String name) {
11 //        System.out.println("first default from interface" + name);
12 //        return 10;
13 //    }
14
15     //Cannot reduce the visibility of the inherited method from usmedical14
16 //    default int getdefault(String name) {
17 //        System.out.println("first default from interface" + name);
18 //        return 10;
19 //    }
20
21     //Cannot reduce the visibility of the inherited method from usmedical14
22 //    @Override
23 //    default void getdefault1(String name) {
24 //        System.out.println("second default from interface" + name);
25 //    }
26
27 //Cannot reduce the visibility of the inherited method from usmedical14
28 //    default void getdefault1(String name) {
29 //        System.out.println("second default from interface" + name);
30 //    }
31
32     //The method getstatic2(String) of type fortis14 must override or implement a supertype method
33 //    @Override
34 //    public static void getstatic2(String name) {
35 //        System.out.println("first static from interface" + name);
36 //    }
37
38    public static void getstatic2(String name) {
39        System.out.println("first static from child class" + name);
40    }
41
42     //The method getstatic1(String) of type fortis14 must override or implement a supertype method
43 //    @Override
44 //    public static int getstatic1(String name) {
45 //        System.out.println("second static from interface" + name);
46 //        return 10;
47 //    }
48
49
50    public static int getstatic1(String name) {
51        System.out.println("second static from child class" + name);
52        return 10;
53    }
54
55
56
57
58 }
59 }
```

```
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
9         should be accessed in a static way
10        System.out.println(v1);
11        f1.getstatic2("tiger");//The static method getstatic2(String) from the type fortis14 should be
12        accessed in a static way
13
14        int v2=fortis14.getstatic1("karan");
15        System.out.println(v2);
16
17        fortis14.getstatic2("new zealand");
18
19        int v3=f1.MINFEE;
20        System.out.println(v3);
21
22 //        usmedical14 u1=new usmedical14();
23 //        int v4=u1.getstatic1("gorilla");
24 //        System.out.println(v4);
25 //        u1.getstatic2("zebra");
26
27        usmedical14 u1=new fortis14();
28        int b1=u1.getdefault("australisa");
29        System.out.println(b1);
30
31        u1.getdefault1("canada");
32
```

```
29         u1.getdefault1("canada");
30
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 }
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-

The screenshot shows a Java code editor window with the file `usmedical15.java` open. The code defines a public interface `usmedical15` with several methods, some of which are annotated with `@Override`. The code includes comments explaining the use of default methods and static final fields. The code editor has tabs for `usmedical15.java`, `fortis15.java`, and `testfor16.java`.

```
1 package com.day19;
2
3 public interface usmedical15 {
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    @Override 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    @Override 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,
23    // static and strictfp are permitted
24    // public static final int getstatic1(String name) {
25    //     System.out.println("third default from interface" + name);
26    //     return 10;
27    // }
28
29    //public static void getstatic2(String name) {
30    //     System.out.println("first static from interface" + name);
31    //}
32    @Override public static int getstatic1(String name) {
33        System.out.println("second static from interface" + name);
34        return 10;
35    }
36}
37
```

```

1 package com.day19;
2
3 //can implement multiple interface.
4 public class fortis15 implements usmedical15{
5
6     int MINFEE=654657567;
7
8     //change default to another access specifier to use it.
9     @Override
10    public int getdefault(String name) {
11        System.out.println("first default from child class" + name);
12        return 10;
13    }
14
15
16    // protected int getdefault(String name) {
17    //     System.out.println("first default from interface" + name);
18    //     return 10;
19    // }
20
21    //Cannot reduce the visibility of the inherited method from usmedical15
22    //protected not allowed.
23    // @Override
24    // protected void getdefault1(String name) {
25    //     System.out.println("second default from interface" + name);
26    // }
27
28    //System.out.println("second default from interface" + name);
29
30    //Cannot reduce the visibility of the inherited method from usmedical15
31    //cannot have private.
32    // @Override
33    // private void getdefault1(String name) {
34    //     System.out.println("second default from interface" + name);
35    // }
36
37    //only public allowed.
38    @Override
39    public void getdefault1(String name) {
40        System.out.println("second default from child class" + name);
41    }
42
43    //System.out.println("second default from child class" + name);
44
45
46    //The method getstatic2(String) of type fortis14 must override or implement a supertype method
47    //cannot use override with static.
48    // @Override
49    // public static void getstatic2(String name) {
50    //     System.out.println("first static from interface" + name);
51    // }
52
53    //without override works, method hiding.
54    public static void getstatic2(String name) {
55        System.out.println("first static from child class" + name);
56    }
57

```

```

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

```

```

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
19         way
20         System.out.println(v3);
21
22         int v4=fortis15.getstatic1("poland");
23         System.out.println(v4);
24
25         f1.getstatic2("uk");
26         //warning.
27         //The static method getstatic2(String) from the type fortis15 should be accessed in a static
28         way
29
30         //warning.
31         //The static method getstatic2(String) from the type fortis15 should be accessed in a static
32         way
33
34         fortis15.getstatic2("germany");
35
36         //Cannot instantiate the type usmedical15
37         usmedical15 u1=new usmedical15();
38
39         usmedical15 u1=new fortis15();
40         int b1=u1.getdefault("canada");
41         System.out.println(b1);
42
43         u1.getdefault1("australisa");
44
45         int b2=u1.MINFEE;
46         System.out.println(b2);
47
48         //cannot access static from interface using objects.
49
50         //using interface name we can access.
51         int c1=usmedical15.getstatic1("hungary");
52         System.out.println(c1);
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76

```

```
//using interface name we can access.  
45 int c1=usmedical15.getstatic1("hungary");  
46 System.out.println(c1);  
47  
48 usmedical15.getstatic2("budapest");  
49  
50 }  
51  
52 }  
53 }  
  
54 }  
55 //654657567  
56 //first default from child classkaran  
57 //10  
58 //second default from child classpawan  
59 //second static from child classranjani  
60 //10  
61 //second static from child classpoland  
62 //10  
63 //first static from child classuk  
64 //first static from child classgermany  
65 //first default from child classcanada  
66 //10  
67 //second default from child classaustralisa  
68 //214234  
69 //second static from interfacehungary  
70 //10  
71 //first static from interfacebudapest  
72  
73  
74
```

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 -

The screenshot displays a Java development environment with two code editors. The top editor contains the file `usmedical16.java`, which defines an interface `usmedical16` with several methods, some of which have illegal modifiers like `private` or `final`. The bottom editor contains the file `fortis16.java`, which implements the `usmedical16` interface. The code in `fortis16.java` includes a comment indicating it can implement multiple interfaces and shows a partially implemented method `getstreet()`.

```
1 package com.day19;
2
3 public interface usmedical16 {
4
5 // private int i=10;//Illegal modifier for the interface field usmedical16.i; only public, static &
6 // final are permitted
7 // private void getname();//This method requires a body instead of a semicolon
8 // public final void getstreet();//Illegal modifier for the interface method getstreet; only public,
9 // private, abstract, default, static and strictfp are permitted
10
11     private int getAddress(String name) {
12         System.out.println(name);
13         return 10;
14     }
15
16     default int getprivateMethod(String name) {
17         getAddress(name);
18         return 100;
19     }
}
1 package com.day19;
2
3 //can implement multiple interface.
4 public class fortis16 implements usmedical16{
5
6 // @Override
7 // public void getstreet() {
8 //     System.out.println("child class get street");
9 // }
10
11
12
13
14
15
16 }
17
```

```
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
```

The screenshot displays a Java development environment with two code editors side-by-side.

Top Editor (usmedical16.java):

```
1 package com.day19;
2
3 public interface usmedical16 {
4
5 // private int i=10;//Illegal modifier for the interface field usmedical16.i; only public, static &
6 // final are permitted
7 // public final void getstreet();//Illegal modifier for the interface method getstreet; only public,
8 // private, abstract, default, static and strictfp are permitted
9
10 private int getAddress(String name) {
11     System.out.println(name);
12     return 10;
13 }
14
15 default int getprivateMethod(String name) {
16     getAddress(name);
17     return 100;
18 }
19 }
```

Bottom Editor (fortis16.java):

```
1 package com.day19;
2
3 //can implement multiple interface.
4 public class fortis16 implements usmedical16{
5
6 // @Override
7 // public void getstreet() {
8 //     System.out.println("child class get street");
9 // }
```

```

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

```

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.

```
 65      }
 66
 67  public void pathologyServices() {
 68      System.out.println("Fh -- pathologyServices");
 69      medicalInsurance();
 70
 71 }
```

Paste usmedical17, fortis17, testfor18-

```
usmedical17.java
1 package com.day19;
2
3 public interface usmedical17 {
4
5     private int getAddress(String name) {
6         System.out.println(name);
7         return 10;
8     }
9
10    default int getprivateMethod(String name) {
11        getAddress(name);
12        return 100;
13    }
14 }
15
```

```
fortis17.java
1 package com.day19;
2
3 //can implement multiple interface.
4 public class fortis17 implements usmedical17{
5
6
7     public void pathservices(String name) {
8         System.out.println("pathology services from fortis " +name);
9         //another way to call default in child class.
10        //directly call the method name.
11        getprivateMethod(name);
12    }
13
14 }
```

```
1 package com.day19;
2
3 public class testfor18 {
4
5     public static void main(String[] args) {
6
7         fortis17 f1=new fortis17();
8         int v1=f1.getprivateMethod("karan");
9         System.out.println(v1);
10
11         f1.pathservices("jigar");
12
13         //Cannot instantiate the type usmedical17
14         //usmedical17 u1=new usmedical17();
15         //usmedical17 u1=new fortis17();
16         //int v2=u1.getprivateMethod("jumbo");
17         //System.out.println(v2);
18
19     }
20
21 }
22
23 //karan
24 //100
25 //pathology services from fortis jigar
26 //jigar
27 //jumbo
28 //100
```

Paste usmedical18, fortis18-

The screenshot shows a Java IDE interface with two code editors. The top editor contains the file `usmedical18.java` with the following code:

```
1 package com.day19;
2
3 public interface usmedical18 {
4
5     private int getAddress(String name) {
6         System.out.println(name);
7         return 10;
8     }
9
10    default int getprivateMethod(String name) {
11        getAddress(name);
12        return 100;
13    }
14 }
15
```

The bottom editor contains the file `fortis18.java` with the following code:

```
1 package com.day19;
2
3 //can implement multiple interface.
4 public class fortis18 implements usmedical18{
5
6
7     //cannot have method inside method in interface.
8     //we have to define the inner method also in class.
9     //else error.
10 //    @Override
11 //    public int getprivateMethod(String name) {
12 //        getAddress(name); //The method getAddress(String) is undefined for the type fortis18
13 //        return 100;
14 //    }
15
16 }
```

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

```
66  public void pathologyServices() {
67      System.out.println("Fh -- pathologyServices");
68      medicalInsurance();
69  }
70
71  //common
72  @Override
73  public void emergencyServices() {
74      System.out.println("Fh -- emergencyServices");
75  }
76
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 -

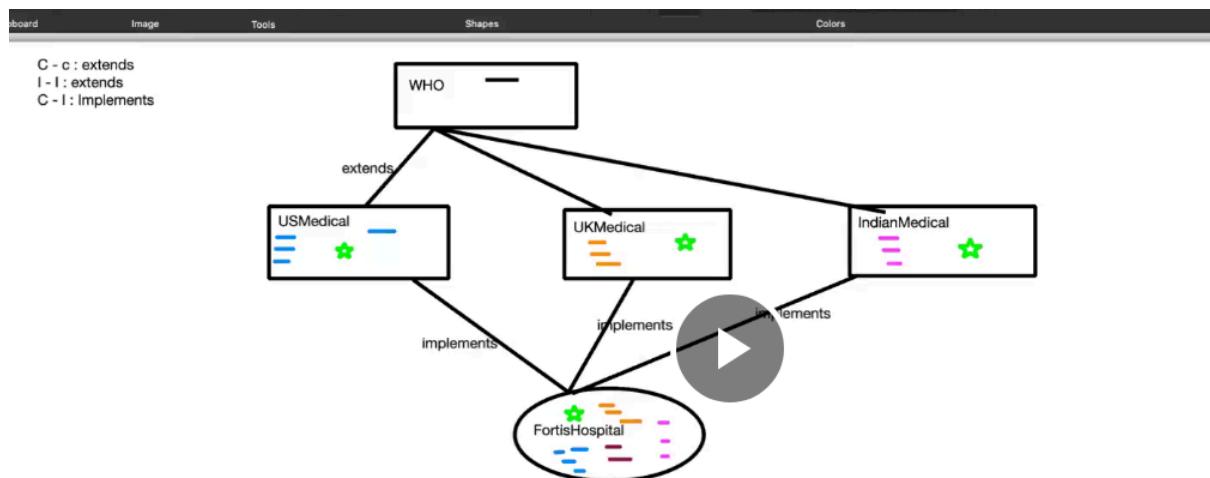
```
usmedical19.java
1 package com.day19;
2
3 public interface usmedical19 {
4
5     default int getprivateMethod(String name) {
6         System.out.println(name);
7         return 100;
8     }
9 }
10

fortis19.java
1 package com.day19;
2
3 //can implement multiple interface.
4 public class fortis19 implements usmedical19{
5
6     @Override
7     public int getprivateMethod(String name) {
8         System.out.println(name + " " + " from fortis");
9         return 100;
10    }
11
12 }
13
```

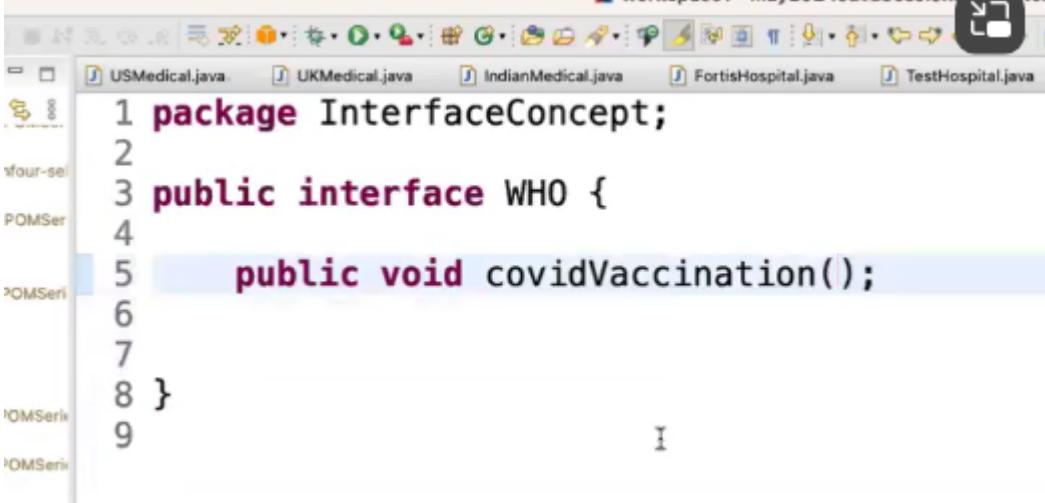
```

testfor20.java ×
1 package com.day19;
2
3 public class testfor20 {
4
5     public static void main(String[] args) {
6
7         fortis19 f1=new fortis19();
8         int v1=f1.getprivateMethod("jogle");
9         System.out.println(v1);
10
11        //Cannot instantiate the type usmedical19
12        usmedical19 u1=new usmedical19();
13        //int v2=u1.getprivateMethod("jumbo");
14        //System.out.println(v2);
15
16        usmedical19 u1=new fortis19();
17        int v2=u1.getprivateMethod("jumbo karan");
18        System.out.println(v2);
19
20    }
21
22 }
23
24 //jogle from fortis
25 //100
26 //jumbo karan from fortis
27 //100
28
29
30

```



Who-

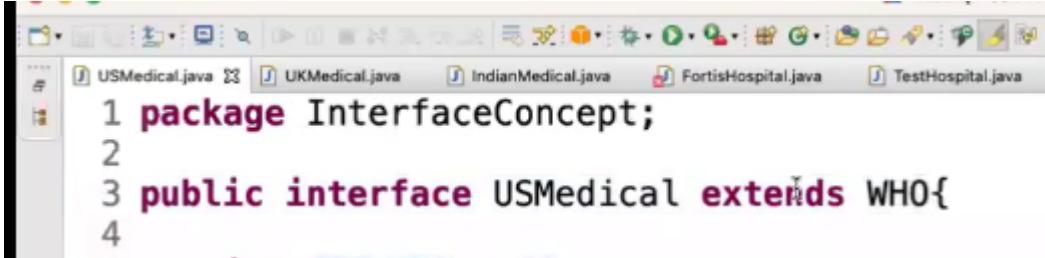


```

1 package InterfaceConcept;
2
3 public interface WHO {
4
5     public void covidVaccination();
6
7
8 }
9

```

Us medical-



```

1 package InterfaceConcept;
2
3 public interface USMedical extends WHO{
4
5
6
7
8
9

```

Fortis-

```

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

```

Test-

```

30
31     fh.covidVaccination();

```

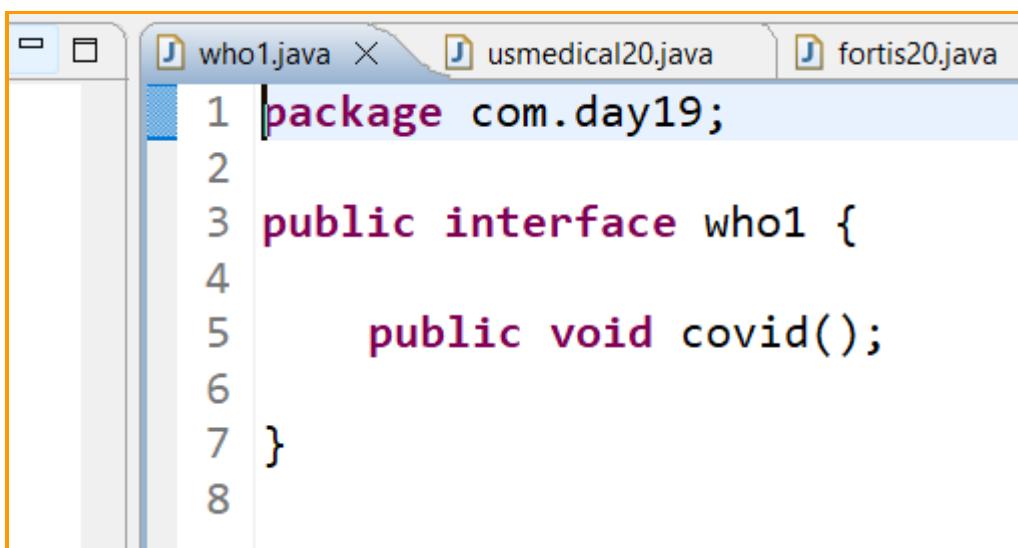
Fh -covid.

Us medical-

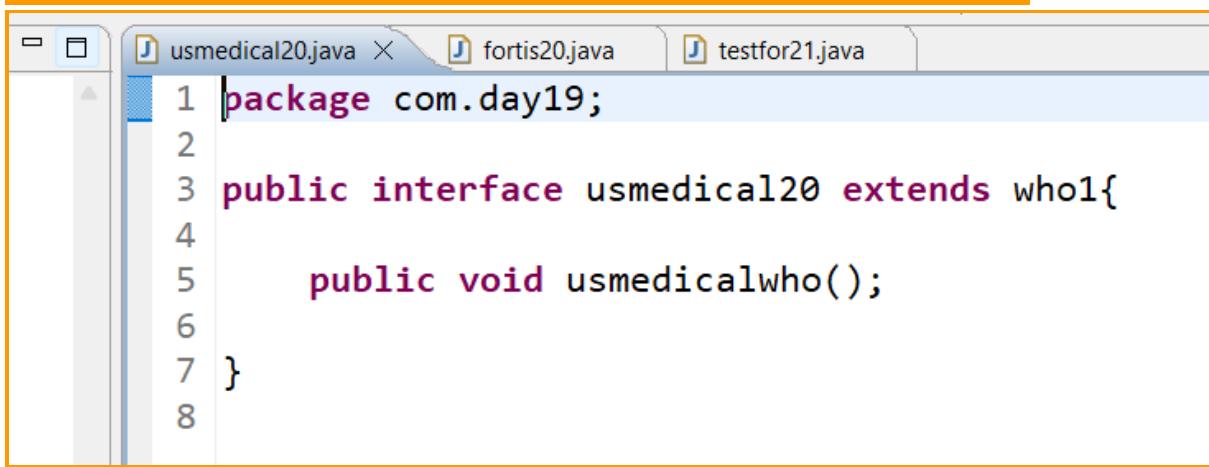
Can access the who method.

```
44      us.medicalTreatment();  
45      us.covidVaccination();  
46
```

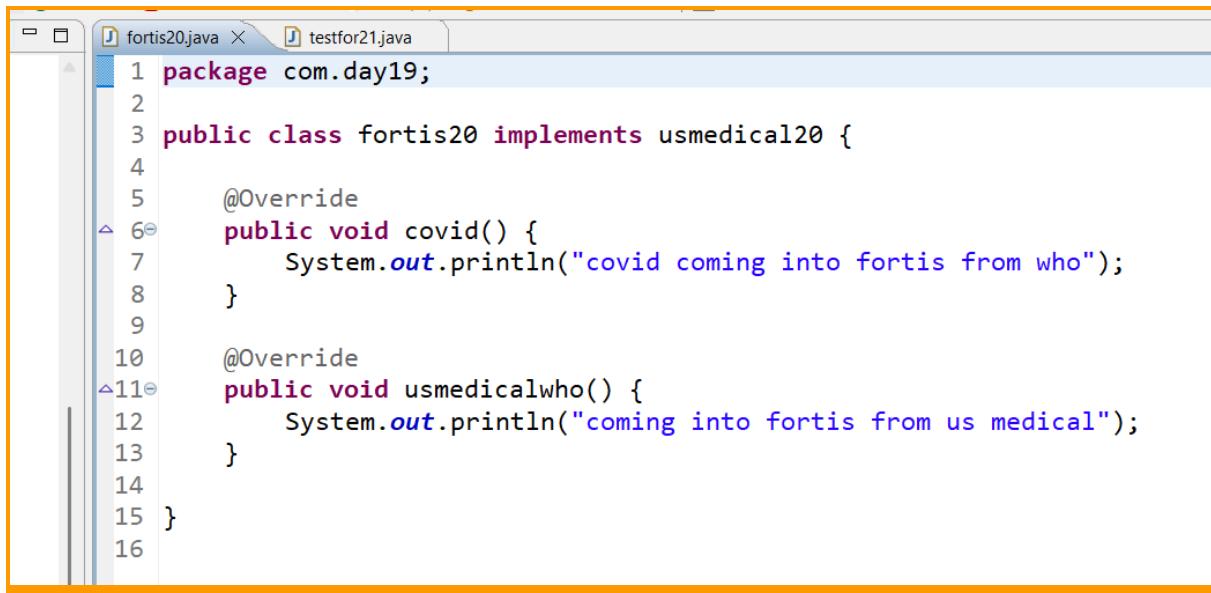
Paste who1, usmedical20, fortis20, testfor21-



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



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



```
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
```

```

1 package com.day19;
2
3 public class testfor21 {
4
5     public static void main(String[] args) {
6
7         fortis20 f1=new fortis20();
8         f1.covid();
9         f1.usmedicalwho();
10
11        //Cannot instantiate the type usmedical20
12        //usmedical20 u1=new usmedical20();
13
14        usmedical20 u1=new fortis20();
15        u1.covid();
16        u1.usmedicalwho();
17
18        //Cannot instantiate the type who1
19        //who1 w1=new who1();
20        who1 w1=new fortis20();
21        w1.covid();
22
23    }
24
25 }
26 //covid coming into fortis from who
27 //coming into fortis from us medical
28 //covid coming into fortis from who
29 //coming into fortis from us medical
30 //covid coming into fortis from who
31
32

```

One interface can extend more than one other interface –

Paste int1, int2, int3, class1, testfor22-

The screenshot shows a Java IDE interface with a central code editor window. The code editor has a light blue header bar with tabs for "int1.java", "int2.java", "int3.java", and "class". The "int1.java" tab is active. The code itself is a Java interface definition:

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

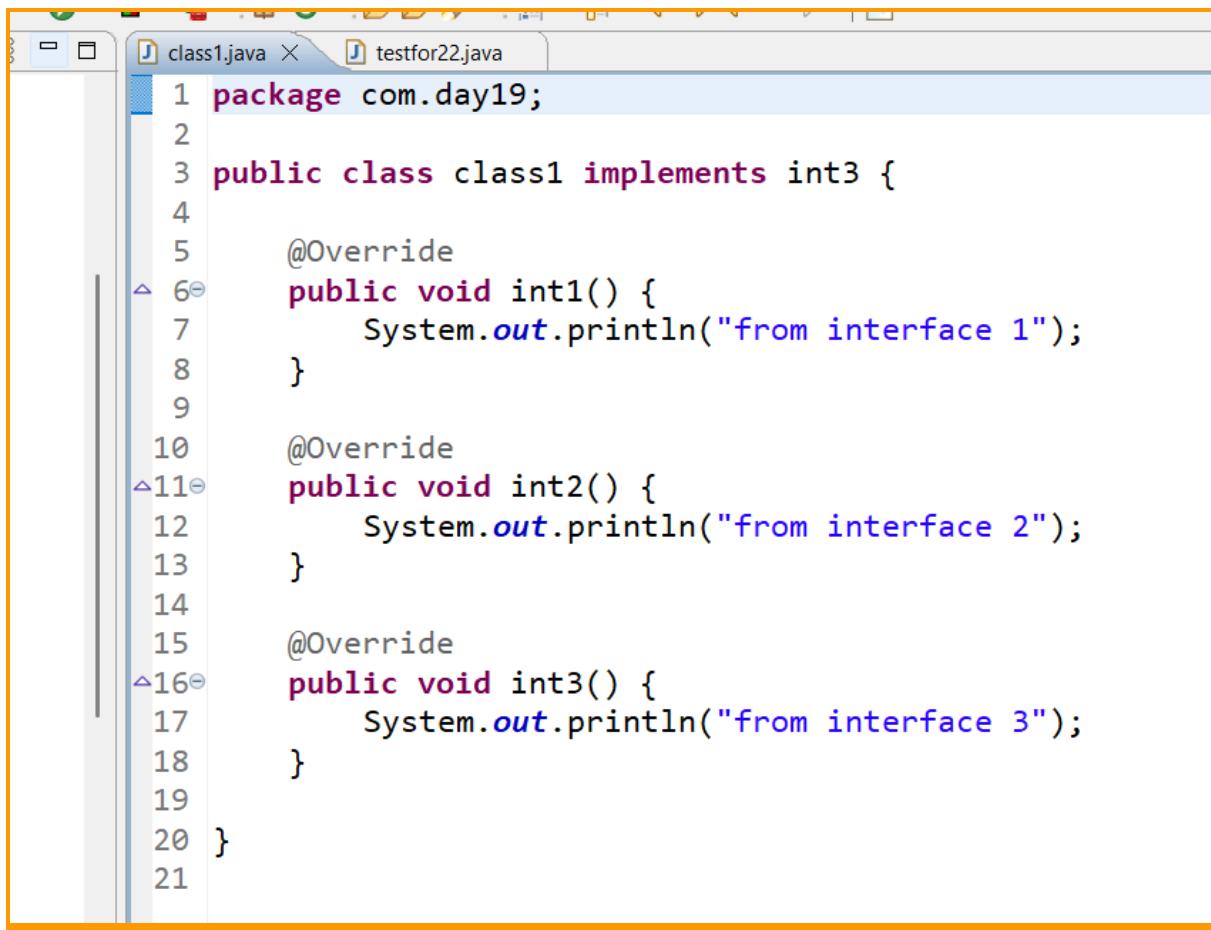
The code is numbered from 1 to 7 on the left. The word "int1" in the interface name and the method name is underlined with a yellow squiggly line, indicating a potential spelling error or a warning from the IDE.

The screenshot shows a Java development environment with two code editors side-by-side. Both editors have tabs at the top labeled 'int2.java X' and 'int3.java'. The left editor contains the following code:

```
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
```

The right editor contains the following code:

```
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
```



The screenshot shows a Java code editor window with the file "class1.java" open. The code defines a class named "class1" that implements an interface named "int3". The class contains three methods: "int1", "int2", and "int3", each printing a specific message to the console using System.out.println.

```
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 X
1 package com.day19;
2
3 public class testfor22 {
4
5     public static void main(String[] args) {
6
7         class1 c1=new class1();
8         c1.int1();
9         c1.int2();
10        c1.int3();
11
12 //        int3 i3=new int3(); //Cannot instantiate the type int3
13 //        int3 i3=new class1();
14 //        i3.int1();
15 //        i3.int2();
16 //        i3.int3();
17
18 //        int2 i2=new int2(); //Cannot instantiate the type int2
19 //        int2 i2=new class1();
20 //        i2.int1();
21 //        i2.int2();
22
23 //        int1 i1=new int1(); //Cannot instantiate the type int1
24 //        int1 i1=new class1();
25 //        i1.int1();
26
27    }
28
29 }
30
31 //from interface 1
32 //from interface 2
33 //from interface 3
34 //from interface 1
35 //from interface 2
36 //from interface 3
37 //from interface 1
38 //from interface 2
39 //from interface 1
40
41
```

Multiple inheritance –

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

Multi level inheritance-

Grand parent to parent to child etc.

Us medical –

```
23  public int test(int a);
24
```

Fortis-

```
98
99=     @Override
100    public int test(int a) {
101        // TODO Auto-generated method stub
102        return 100;
103    }
104
```

Us medical-

```
22
23  public int test(int a);
24
25  public int test(int a, int b);
26
```

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
48     //down casting:
49     FortisHospital f1 = (FortisHospital) new USMedical();
50

```

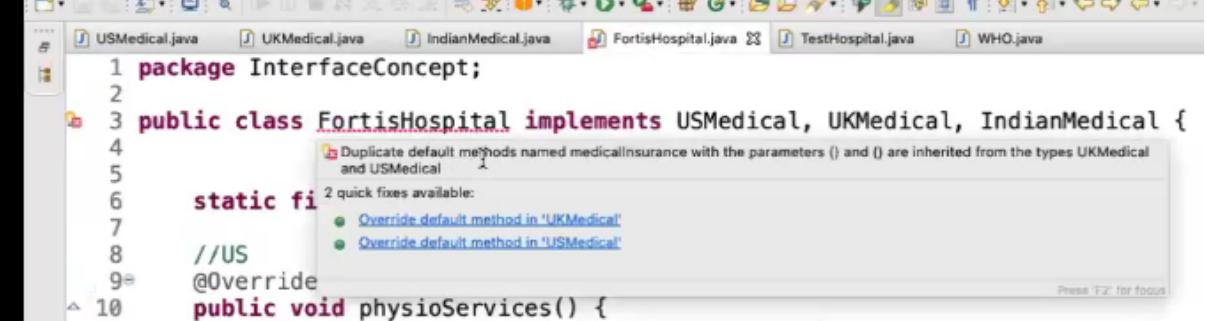
Same default method in uk interface-

```

1 package InterfaceConcept;
2
3 public interface UKMedical {
4
5     public void entServices();
6
7     public void pediaServices();      I
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-



A screenshot of an IDE showing a Java code editor. The code implements three interfaces: USMedical, UKMedical, and IndianMedical. It contains a static field and a method physioServices(). A tooltip appears over the method, stating: "Duplicate default methods named medicalInsurance with the parameters () and () are inherited from the types UKMedical and USMedical". It also says "2 quick fixes available:" with options "Override default method in 'UKMedical'" and "Override default method in 'USMedical'".

```
1 package InterfaceConcept;
2
3 public class FortisHospital implements USMedical, UKMedical, IndianMedical {
4
5     static fi
6     //US
7     @Override
8     public void physioServices() {
```

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-

The screenshot shows an IDE interface with three tabs open:

- default1.java**: Contains the following code:


```

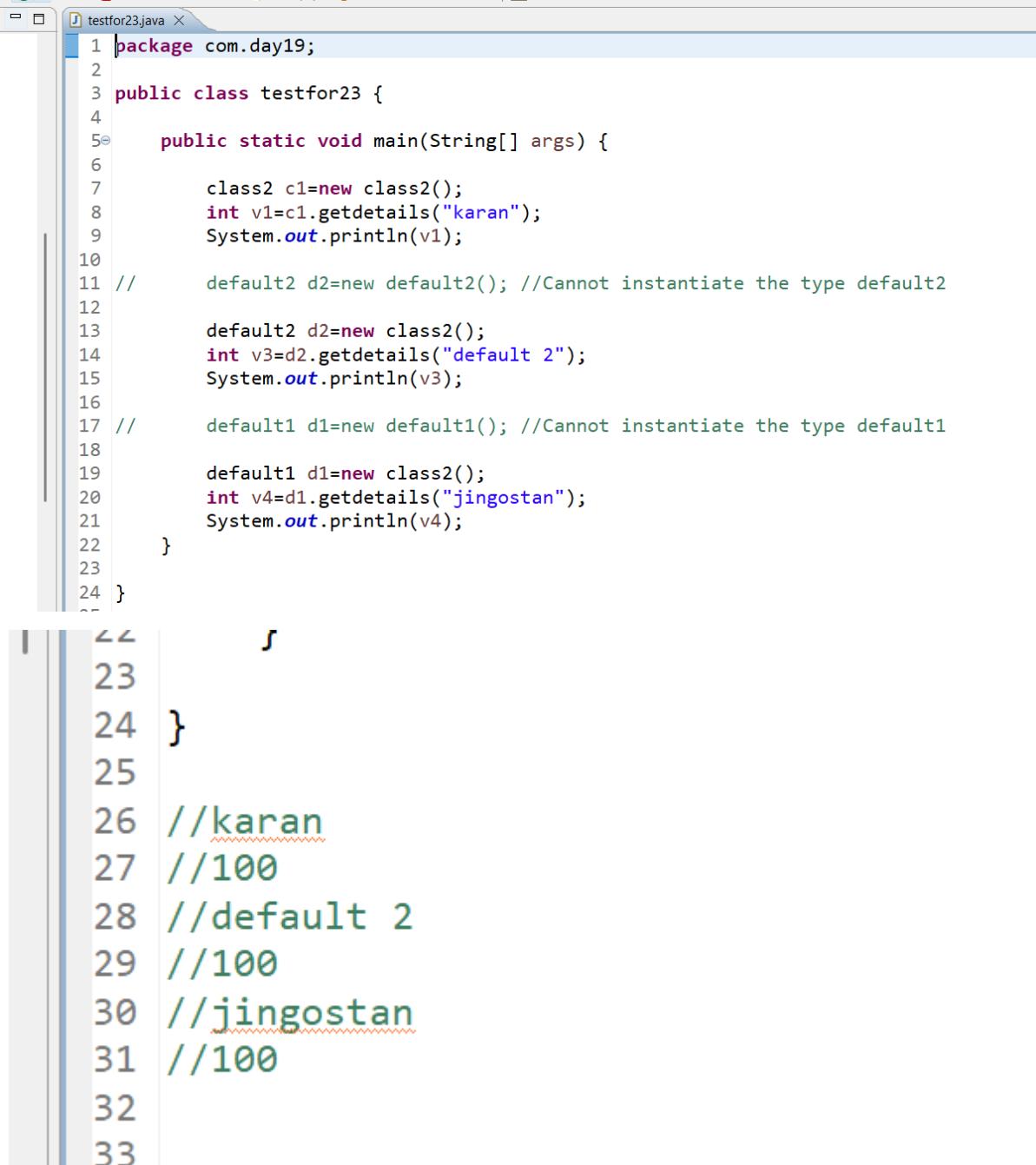
1 package com.day19;
2
3 public interface default1 {
4
5     default int getdetails(String name) {
6         System.out.println(name);
7         return 100;
8     }
9
10 }
11
      
```
- default2.java**: Contains the following code:


```

1 package com.day19;
2
3 public interface default2 {
4
5     default int getdetails(String name) {
6         System.out.println(name);
7         return 100;
8     }
9
10 }
11
      
```
- class2.java**: Contains the following code:


```

1 package com.day19;
2
3 public class class2 implements default1, default2 {
4
5     @Override
6     public int getdetails(String name) {
7         // TODO Auto-generated method stub
8         return default1.super.getdetails(name);
9     }
10    //Duplicate default methods named getdetails with the parameters (String) and (String) are
11    //inherited from the types default2 and default1
12    //at a time we can override from one interface.
13    //if two interfaces have same default method implementation.
14
15 }
16
      
```



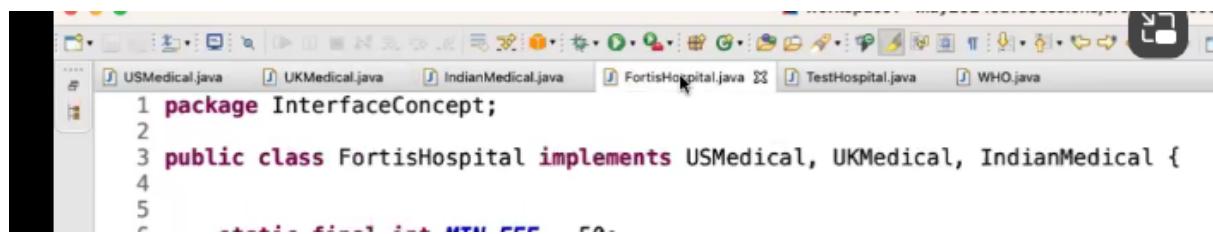
```
1 package com.day19;
2
3 public class testfor23 {
4
5     public static void main(String[] args) {
6
7         class2 c1=new class2();
8         int v1=c1.getdetails("karan");
9         System.out.println(v1);
10
11 //        default2 d2=new default2(); //Cannot instantiate the type default2
12
13        default2 d2=new class2();
14        int v3=d2.getdetails("default 2");
15        System.out.println(v3);
16
17 //        default1 d1=new default1(); //Cannot instantiate the type default1
18
19        default1 d1=new class2();
20        int v4=d1.getdetails("jingostan");
21        System.out.println(v4);
22    }
23
24 }
25
26 //karan
27 //100
28 //default 2
29 //100
30 //jingostan
31 //100
32
33
```

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.



A screenshot of an IDE showing a Java code editor. The code implements the `USMedical`, `UKMedical`, and `IndianMedical` interfaces. The code is as follows:

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

Paste default3, default4, class3, testfor24-

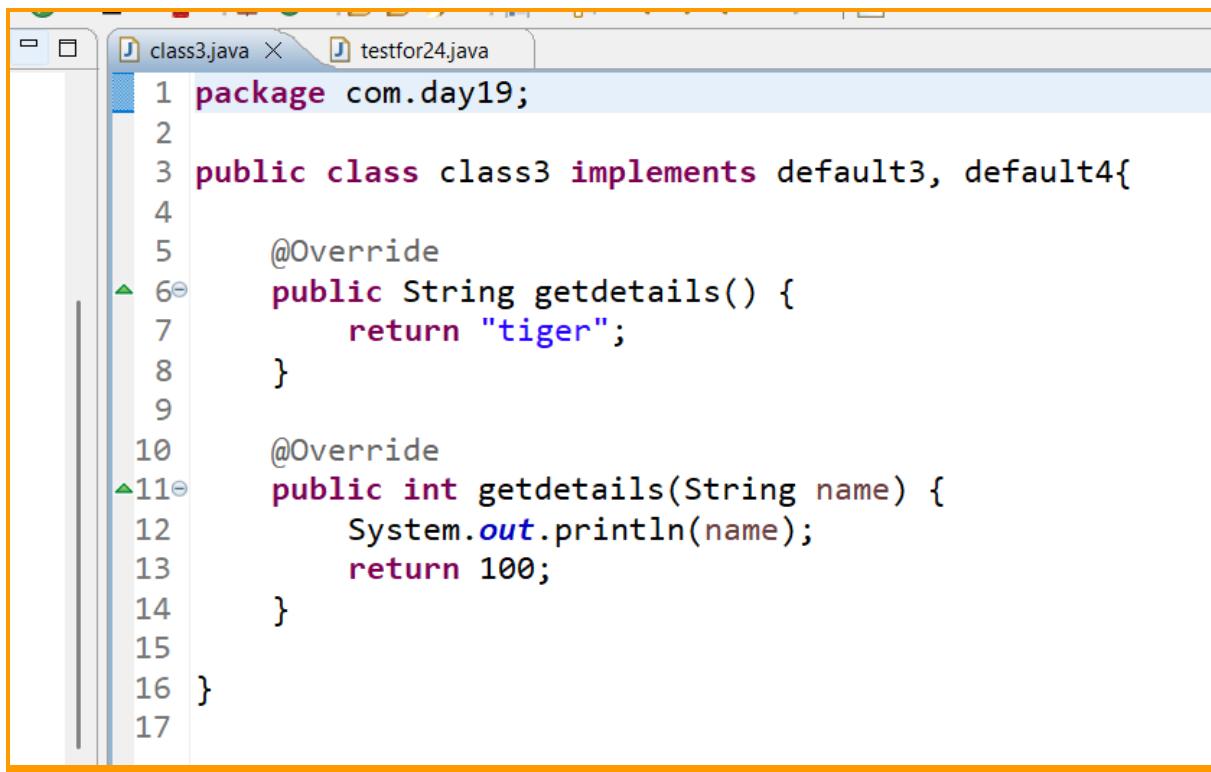
The image shows a Java development environment with two code editors side-by-side, both highlighted with an orange border.

Top Editor (default3.java):

```
1 package com.day19;
2
3 public interface default3 {
4
5     default int getdetails(String name) {
6         System.out.println(name);
7         return 100;
8     }
9
10 }
11
```

Bottom Editor (default4.java):

```
1 package com.day19;
2
3 public interface default4 {
4
5     default String getdetails() {
6         return "tiger";
7     }
8
9 }
10
```



The screenshot shows a Java code editor window with the file "class3.java" open. The code implements two interfaces: default3 and default4. It contains two overridden methods: getdetails() returning a string "tiger" and getdetails(String name) which prints the name to the console and returns 100. The code is numbered from 1 to 17.

```
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
1 package com.day19;
2
3 public class testfor24 {
4
5     public static void main(String[] args) {
6
7         class3 c1=new class3();
8         String s1=c1.getdetails();
9         System.out.println(s1);
10
11        int v1=c1.getdetails("tiger");
12        System.out.println(v1);
13
14 //        default4 d4=new default4(); //Cannot instantiate the type default4
15 //        default4 d4=new class3();
16 //        String s2=d4.getdetails();
17 //        System.out.println(s2);
18
19
20 //        default3 d3=new default3(); //Cannot instantiate the type default3
21
22        default3 d3=new class3();
23        int v2=d3.getdetails("janadhan");
24        System.out.println(v2);
25
26    }
27
28 }
29
30 //tiger
31 //tiger
32 //100
33 //tiger
34 //janadhan
35 //100
36
37
```