

```
1 Interface
2 =====
3 program 1
4 // weaker privileges
5 interface itf1    // abstract
6 {
7     void m1();    //public abstract
8     void m2();
9     void m3();
10 }
11 class Test implements itf1
12 {
13
14     void m1()
15     {
16         System.out.println("m1 method ");
17     }
18     void m2()
19     {
```

```
20      System.out.println("m2 method ");
21  }
22  void m3()
23  {
24      System.out.println("m2 method ");
25  }
26  public static void main(String [] args )
27  {
28      Test t = new Test();
29      t.m1();
30      t.m2();
31      t.m3();
32  }
33 }
34 D:\AY 2023-24\SEM-I\JAVA\DIVB>javac interface.java
35 interface.java:17: error: m3() in Test cannot implement m3() in
36     void m3()
37         ^
38     attempting to assign weaker access privileges; was public
```

```
39 interface.java:13: error: m2() in Test cannot implement m2() in i
40     void m2()
41         ^
42     attempting to assign weaker access privileges; was public
43 interface.java:9: error: m1() in Test cannot implement m1() in
44     void m1()
45         ^
46     attempting to assign weaker access privileges; was public
47 3 errors
48 =====
49 program 2
50 // declare all overriding method as public
51 interface itf1    // abstract
52 {
53     void m1();    //public abstract
54     void m2();
55     void m3();
56 }
57 class Test implements itf1
```

```
58 {
59     public void m1()
60     {
61         System.out.println("m1 method ");
62     }
63     public void m2()
64     {
65         System.out.println("m2 method ");
66     }
67     public void m3()
68     {
69         System.out.println("m2 method ");
70     }
71     public static void main(String [] args )
72     {
73         Test t = new Test();
74         t.m1();
75         t.m2();
76         t.m3();
```

```
77     }
78 }
79 D:\AY 2023-24\SEM-I\JAVA\DIVB>java Test
80 m1 method
81 m2 method
82 m2 method
83 =====
84 program 3:
85 // multiple classes implementing overridden methods
86 interface itf1    // abstract
87 {
88     void m1();    //public abstract
89     void m2();
90     void m3();
91 }
92 abstract class Test implements itf1
93 {
94     public void m1()
95     {
```

```
96         System.out.println("m1 method ");
97     }
98     public void m2()
99     {
100         System.out.println("m2 method ");
101     }
102
103 }
104 class Test1 extends Test
105 {
106     public void m3()
107     {
108         System.out.println("m2 method ");
109     }
110     public static void main(String [] args )
111     {
112         Test1 t = new Test1();
113         t.m1();
114         t.m2();
```

```
115         t.m3();
116     }
117 }
118 D:\AY 2023-24\SEM-I\JAVA\DIVB>java Test1
119 m1 method
120 m2 method
121 m2 method
122 =====
123 program 4:
124 // accesssing through interface reference variable
125 interface itf1    // abstract
126 {
127     void m1();    //public abstract
128     void m2();
129     void m3();
130 }
131 abstract class Test implements itf1
132 {
133     public void m1()
```

```
134     {
135         System.out.println("m1 method ");
136     }
137     public void m2()
138     {
139         System.out.println("m2 method ");
140     }
141
142 }
143 class Test1 extends Test
144 {
145     public void m3()
146     {
147         System.out.println("m2 method ");
148     }
149     public static void main(String [] args )
150     {
151         Test1 t = new Test1();
152         t.m1();
```



```
153         t.m2();
154         t.m3();
155
156         itf1 i = new Test1(); // accesssing through interface
157         i.m1();
158         i.m2();
159         i.m3();
160     }
161 }
162 D:\AY 2023-24\SEM-I\JAVA\DIVB>java Test1
163 m1 method
164 m2 method
165 m2 method
166 m1 method
167 m2 method
168 m2 method
169 =====
170 program 5
171 // Adapter class
```

```
172 interface itf1    // abstract
173 {
174     void m1();      //public abstract
175     void m2();
176     void m3();
177 }
178 class Demo implements itf1    // Adampter class
179 {
180     public void m1(){}
181     public void m2(){}
182     public void m3(){}
183 }
184 class Test1 extends Demo
185 {
186     public void m1()
187     {
188         System.out.println("m1 method ");
189     }
190     public static void main(String [] args )
```

```
191     {
192         Test1 t = new Test1();
193         t.m1();
194         t.m2();
195         t.m3();
196
197
198     }
199 }
200 D:\AY 2023-24\SEM-I\JAVA\DIVB>java Test1
201 m1 method
202 =====
203 program 6
204 // nested interface
205 interface itf1          // nested interface
206 {
207     interface itf2
208     {
209         void m1();
```

```
210     }
211 }
212
213 class Test1 implements itf1.itf2
214 {
215     public void m1()
216     {
217         System.out.println("m1 method ");
218     }
219     public static void main(String [] args )
220     {
221         Test1 t = new Test1();
222         t.m1();
223     }
224 }
225 }
226 D:\AY 2023-24\SEM-I\JAVA\DIVB>java Test1
227 m1 method
228 =====
```

```
229 program 7
230 // interface inside a class
231 class Demo
232 {
233     interface itf2
234     {
235         void m1();
236     }
237 }
238
239 class Test1 implements Demo.itf2
240 {
241     public void m1()
242     {
243         System.out.println("m1() method ");
244     }
245     public static void main(String [] args )
246     {
247         Test1 t = new Test1();
```

```
248         t.m1();
249
250     }
251 }
252 D:\AY 2023-24\SEM-I\JAVA\DIVB>java Test1
253 m1() method
254
255 =====
256 program 8
257 // final variables
258 interface itf1
259 {
260     int x =100;    // public static final
261     void m1();    // public abstract
262 }
263
264 class Test1 implements itf1
265 {
266     public void m1()
```

```
267     {
268         x = x+200;
269         System.out.println(x);
270     }
271     public static void main(String [] args )
272     {
273         Test1 t = new Test1();
274         t.m1();
275     }
276 }
277 }
```

```
278
279 D:\AY 2023-24\SEM-I\JAVA\DIVB>javac interface.java
280 interface.java:11: error: cannot assign a value to final variable
281         x = x+200;
282         ^
283 1 error
284
285 =====
```

```
286 program 9
287 // ambiguous error
288 interface itf1
289 {
290     int x =100;    // public static final
291
292 }
293 interface itf2
294 {
295     int x = 200;
296 }
297
298 class Test1 implements itf1, itf2
299 {
300     public void m1()
301     {
302         System.out.println(x); // ambiguous
303     }
304     public static void main(String [] args )
```



```
305     {
306         Test1 t = new Test1();
307         t.m1();
308
309     }
310 }
311
312 D:\AY 2023-24\SEM-I\JAVA\DIVB>javac interface.java
313 interface.java:15: error: reference to x is ambiguous
314         System.out.println(x);
315                             ^
316     both variable x in itf1 and variable x in itf2 match
317 1 error
318
319 =====
320 program 10
321 //
322 interface itf1
323 {
```

```
324     int x =100;    // public static final
325
326 }
327 interface itf2
328 {
329     int x = 200;
330 }
331
332 class Test1 implements itf1, itf2
333 {
334     public void m1()
335     {
336         System.out.println(itf1.x);
337         System.out.println(itf2.x);
338     }
339     public static void main(String [] args )
340     {
341         Test1 t = new Test1();
342         t.m1();
```

```
343
344     }
345 }
346 D:\AY 2023-24\SEM-I\JAVA\DIVB>java Test1
347 100
348 200
349 =====
350 program 11
351 // calling variables with corresponding interfaces to avoid am
352 interface itf1
353 {
354     int x =100;    // public static final
355
356 }
357 interface itf2
358 {
359     int x = 200;
360 }
361
```

```
362 class Test1 implements itf1, itf2
363 {
364     public void m1()
365     {
366         System.out.println(itf1.x);
367         System.out.println(itf2.x);
368     }
369     public static void main(String [] args )
370     {
371         Test1 t = new Test1();
372         t.m1();
373     }
374 }
375 }
376 D:\AY 2023-24\SEM-I\JAVA\DIVB>java Test1
377 100
378 200
379
380 =====
```

```
381 program 12
382 // cloing
383 class Demo implements Cloneable
384 {
385     int x = 100;
386     int y =200;
387
388     public static void main(String [] args ) throws Exception
389     {
390         Demo d = new Demo();
391         System.out.println(d.x);
392         System.out.println(d.y);
393
394         d.x = 500;
395         d.y=600;
396         System.out.println(d.x);
397         System.out.println(d.y);
398         Demo d1= (Demo)d.clone();    // cloning /duplicating o
399         ;;;;
```

```
400      ;;;;  
401      ;;;;  
402      ;;;;  
403      ;;;;  
404      ;;;  
405      ;;;;  
406      d.x = 700;  
407      d.y=800;  
408      System.out.println(d.x);  
409      System.out.println(d.y);  
410      // i want to print 500 and 600  
411      System.out.println(d1.x);  
412      System.out.println(d1.y);  
413  
414  
415  }  
416 }  
417 D:\AY 2023-24\SEM-I\JAVA\DIVB>java Demo  
418 100
```

```
419 200
420 500
421 600
422 700
423 800
424 500
425 600
426
427 =====
428 Marker Interface
429
430 D:\AY 2023-24\SEM-I\JAVA\DIVB>javap java.io.Serializable
431 Compiled from "Serializable.java"
432 public interface java.io.Serializable {
433 }
434
435 D:\AY 2023-24\SEM-I\JAVA\DIVB>javap java.Lang.Cloneable
436 Compiled from "Cloneable.java"
437 public interface java.Lang.Cloneable {
```

438 }

439

440 D:\AY 2023-24\SEM-I\JAVA\DIVB>javap java.util.*RandomAccess*

441 Compiled from "*RandomAccess.java*"

442 *public interface* java.util.*RandomAccess* {

443 }

444

445

446

447

448

449

450