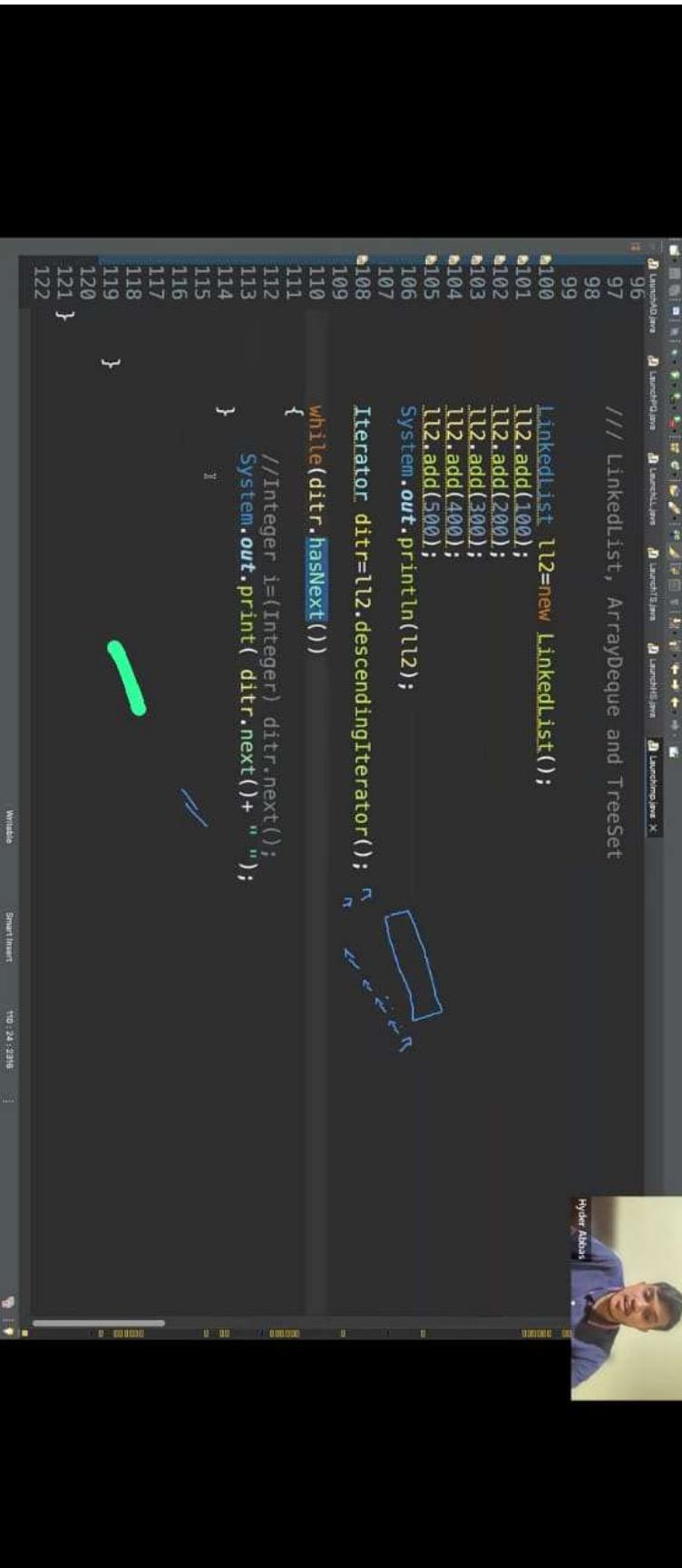


Java Collection Part-3



The image shows a composite screen with two main sections. On the left is a dark-themed Java code editor with white text. On the right is a video player window displaying a video of a man speaking. A green highlighter has been used to mark several lines of code in the editor.

```
96 // // LinkedList, ArrayDeque and TreeSet
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 }
```

A blue highlighter has also been used to draw a small rectangle around the word "descendingIterator" in the code.

A screenshot of a Java IDE interface. The code editor shows the following Java code:

```
1 import java.util.Vector;
2
3 public class LaunchV {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7
8         Vector v=new Vector();
9         v.add(100);
10        v.add(200);
11        v.add(300);
12        v.add(400);
13        v.add(500);
14    }
15
16    v.
17
18    v.
19    v.
20 }
```

The cursor is positioned at the end of the line starting with 'v.' in the code editor. A tooltip window is open, providing information about the `v.add` method:

v.add(Double d) Returns: A `Double` element. Specified by `add(D)` in `Object`. Overrides: `add(D)` in `Vector`.

Parameters: `d` - the element to be added to this `Vector`.

Returns: `True` if the element was successfully added; `false` if this `Vector` is full.

Since: 1.2

The tooltip also includes a note: "Press Shift+Space to open previous method or click for more".

A screenshot of a Java IDE interface. At the top, there's a toolbar with various icons. Below it is a package explorer showing several Java files like 'Launch.java', 'LaunchList.java', 'LaunchListImpl.java', etc. The main area shows a Java code editor with the following content:

```
1 import java.util.Vector;
2
3 public class LaunchV {
4
5     public static void main(String[] args) {
6
7         Vector v=new Vector();
8
9         v.add(100);
10        v.add(200);
11        v.add(300);
12        v.add(400);
13        v.add(500);
14
15    }
16
17 }
18
19 }
20 }
```

The cursor is positioned at the end of the line 'v.elementAt'. A tooltip is displayed over the word 'elementAt', providing the following information:

v.elementAt

MethodDeclaration [Vector] Object <: Vector
↳ elementAt (int index) : Object
↳ addElement (Object obj, int index) : void <: Vector
↳ removeElement (Object obj) : void <: Vector
↳ removeElementAt (int index) : void <: Vector
↳ setElementAt (Object obj, int index) : void <: Vector

Return value: Returns the component at the specified index.
This method is identical in functionality to the `get(int)` method (which is part of the `List` interface).

Parameters:
index an index into this vector.

Returns:
the component at the specified index.

Throws:
`IndexOutOfBoundsException` - if the index is out of range (`index < 0` || `index > size()`)

Below the code editor, there's a preview pane showing a small video thumbnail of a person speaking. The bottom status bar shows 'Writable' and 'Smart indent'.

A screenshot of a Java development environment. On the left, the package explorer shows several projects and files, including 'Launchy.java' which is currently selected. In the center, a code editor displays the following Java code:

```
1 import java.util.Enumeration;
2 import java.util.Vector;
3
4 public class Launchy {
5
6     public static void main(String[] args) {
7         // TODO Auto-generated method stub
8
9             Vector v=new Vector();
10            v.add(100);
11            v.add(200);
12            v.add(300);
13            v.add(400);
14            v.add(500);
15
16            Enumeration em=v.elements();
17            while(em.hasMoreElements()){
18                System.out.println(em.nextElement());
19            }
20        }
21    }
22
23 }
24
25 }
```

A large green number '4' is drawn over the code editor area. In the bottom right corner of the screen, there is a video player interface showing a person's face.

```
25
26     while(itr.hasNext())
27     {
28         System.out.println(itr.next());
29     }
30     //al.add(123);
31
32     //failsafe
33
34     CopyOnWriteArrayList cal=new CopyOnWriteArrayList();
35     cal.add(1000);
36     cal.add(2000);
37     cal.add(3000);
38     cal.add(4000);
39
40
41     Iterator itr=cal.iterator();
42     while(itr.hasNext())
43     {
44         System.out.println(itr.next());
45     }
46     cal.add(12345);
47
48
49
50
51
```

S

A screenshot of a Java IDE interface. On the left, there is a code editor with the following Java code:

```
1 package Explorer;
2
3 import java.util.*;
4
5 public class Explorer {
6     public static void main(String[] args) {
7         ArrayList<String> al = new ArrayList<String>();
8
9         al.add("50");
10        al.add(150);
11        al.add(25);
12        al.add(75);
13        al.add(125);
14        al.add(175);
15
16        System.out.println(al);
17
18        Collections.sort(al);
19
20        System.out.println(al);
21
22        ArrayList<String> al2 = new ArrayList<String>();
23        al2.add("28");
24        al2.add("Hyderabad");
25        al2.add("Mumbai");
26        al2.add("Chennai");
27        al2.add("Kolkata");
28        al2.add("Bengaluru");
29        al2.add("Pune");
30        al2.add("Delhi");
31        al2.add("Jaipur");
32        al2.add("Chennai");
33        al2.add("Mumbai");
34        al2.add("Kolkata");
35        al2.add("Bengaluru");
36        al2.add("Pune");
37        al2.add("Hyderabad");
38        al2.add("Chennai");
39        al2.add("Delhi");
40        al2.add("Jaipur");
41        al2.add("Kolkata");
42        al2.add("Mumbai");
43        al2.add("Bengaluru");
44        Collections.sort(al2);
45        System.out.println(al2);
46
47
48
49
50 }
```

The code uses two ArrayLists, one for odd integers and one for city names, and prints them out after sorting. A large red number '6' is drawn over the code editor area.

On the right side of the IDE, there is a video player window showing a man speaking. The video player has controls for play, pause, volume, and other media functions. The video is titled "Byder Abbas".

The screenshot shows a Java development environment with a code editor and a terminal window. A video player is overlaid on the terminal window, displaying a man speaking.

```
2.04 PM
```

```
File Edit View Insert Run Tools Window Help
```

```
Package Explorer X
```

```
LaunchConfigurations LaunchConfigurations LaunchConfigurations LaunchConfigurations LaunchConfigurations LaunchConfigurations
```

```
36     ArrayList<String> al2=new ArrayList<String>()
37     // al2.add(28); error
38     al2.add("Hyder");
39     al2.add("ineuron");
40     al2.add("Najafi code");
41     //error al2.add(560025);
42
43
44
45     Collections.sort(al2);
46     System.out.println(al2);
47
48     ArrayList<Integer> al3=new ArrayList<Integer>();
49     al3.add(1000);
50     // al3.add("GF"); error
51     Collections.sort(al3);
52     System.out.println(al3);
53
54
55
56
57
58
59
60
61
62 }
```

Hyder Abbas

```
Wrinkle SmartInnate 52:32:1011
```

```
1. package Explorer X;
2. 
3. import java.util.*;
4. import java.io.*;
5. import java.awt.*;
6. import javax.swing.*;
7. import javax.swing.table.*;
8. import javax.swing.tree.*;
9. import javax.swing.border.*;
10. import javax.swing.GroupLayout.*;
11. import javax.swing.GroupLayout.Alignment.*;
12. import javax.swing.LayoutStyle.ComponentPlacement.*;
13. import javax.swing.JPanel.*;
14. import javax.swing.JButton.*;
15. import javax.swing.JList.*;
16. import javax.swing.JTable.*;
17. import javax.swing.JTable.*;
18. import javax.swing.JScrollPane.*;
19. import javax.swing.JTextPane.*;
20. import javax.swing.JTextPane.*;
21. import javax.swing.JTextPane.*;
22. import javax.swing.JTextPane.*;
23. import javax.swing.JTextPane.*;
24. import javax.swing.JTextPane.*;
25. import javax.swing.JTextPane.*;
26. import javax.swing.JTextPane.*;
27. import javax.swing.JTextPane.*;
28. import javax.swing.JTextPane.*;
29. import javax.swing.JTextPane.*;
30. import javax.swing.JTextPane.*;
31. import javax.swing.JTextPane.*;
32. import javax.swing.JTextPane.*;
33. import javax.swing.JTextPane.*;
34. import javax.swing.JTextPane.*;
35. import javax.swing.JTextPane.*;
36. import javax.swing.JTextPane.*;
37. import javax.swing.JTextPane.*;
38. import javax.swing.JTextPane.*;
39. import javax.swing.JTextPane.*;
40. import javax.swing.JTextPane.*;
41. import javax.swing.JTextPane.*;
42. import javax.swing.JTextPane.*;
43. import javax.swing.JTextPane.*;
44. import javax.swing.JTextPane.*;
45. import javax.swing.JTextPane.*;
46. import javax.swing.JTextPane.*;
47. import javax.swing.JTextPane.*;
48. import javax.swing.JTextPane.*;
49. import javax.swing.JTextPane.*;
50. import javax.swing.JTextPane.*;
51. import javax.swing.JTextPane.*;
52. import javax.swing.JTextPane.*;
53. import javax.swing.JTextPane.*;
54. import javax.swing.JTextPane.*;
55. import javax.swing.JTextPane.*;
56. import javax.swing.JTextPane.*;
57. import javax.swing.JTextPane.*;
58. import javax.swing.JTextPane.*;
59. import javax.swing.JTextPane.*;
60. import javax.swing.JTextPane.*;
61. import javax.swing.JTextPane.*;
62. import javax.swing.JTextPane.*;
63. import javax.swing.JTextPane.*;
64. import javax.swing.JTextPane.*;
65. import javax.swing.JTextPane.*;
66. import javax.swing.JTextPane.*;
67. import javax.swing.JTextPane.*;
```

//few more important inbuilt methods of Collections clas:

```
ArrayList al4=new ArrayList();
al4.add(10);
al4.add(20);
al4.add(30);
al4.add(40);
al4.add(50);
Collections.binarySearch(al4, 40);
```

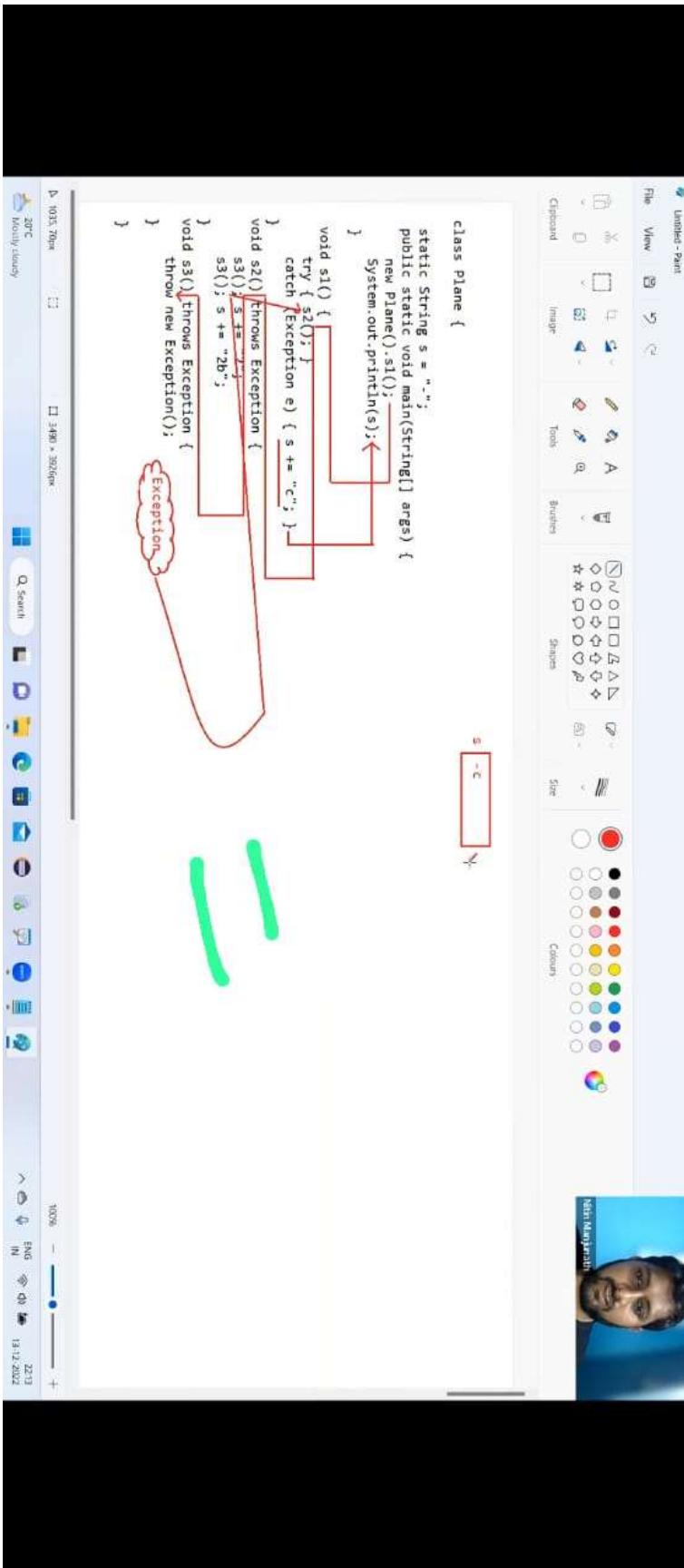


```
//error at l2.add(3600025);
42
43 Collections.sort(al2);
44 System.out.println(al2);
45
46 ArrayList<Integer> al3=new ArrayList<Integer>();
47 al3.add(1000);
48 al3.add(200);
49 // al3.add("GF"); error
50 Collections.sort(al3);
51 System.out.println(al3);
52
53
54 //few more important inbuilt methods of Collections clas:
55
56 ArrayList al4=new ArrayList();
57 al4.add(10);
58 al4.add(20);
59 al4.add(30);
60 al4.add(40);
61 al4.add(50);
62
63 int index= Collections.binarySearch(al4, 40);
64 System.out.println("Index " + index);
65
66
67 }
68 }
```

Writable Smart Insert 65 : 9 : 1280



```
1 package Explorer;
2
3 import java.util.*;
4
5 public class Explorer {
6     public static void main(String[] args) {
7         ArrayList<Integer> al3=new ArrayList<Integer>();
8         al3.add(1000);
9         al3.add(200);
10        // al3.add("GF"); error
11        Collections.sort(al3);
12        System.out.println(al3);
13
14        //few more important inbuilt methods of Collections cla:
15        ArrayList al4=new ArrayList();
16        al4.add(10);
17        al4.add(20);
18        al4.add(30);
19        al4.add(40);
20        al4.add(50);
21
22        int index= Collections.binarySearch(al4, 40);
23        System.out.println(al4);
24
25        Collections.shuffle(al4);
26        System.out.println(al4);
27
28        System.out.println(Collections.frequency(al4, 40));
29
30    }
31
32    System.out.println(Collections.frequency(al4, 40));
33
34    System.out.println(Collections.frequency(al4, 40));
35
36    System.out.println(Collections.frequency(al4, 40));
37
38    System.out.println(Collections.frequency(al4, 40));
39
40    System.out.println(Collections.frequency(al4, 40));
41
42    System.out.println(Collections.frequency(al4, 40));
43
44    System.out.println(Collections.frequency(al4, 40));
45
46    System.out.println(Collections.frequency(al4, 40));
47
48    System.out.println(Collections.frequency(al4, 40));
49
50    System.out.println(Collections.frequency(al4, 40));
51
52    System.out.println(Collections.frequency(al4, 40));
53
54    System.out.println(Collections.frequency(al4, 40));
55
56    System.out.println(Collections.frequency(al4, 40));
57
58    System.out.println(Collections.frequency(al4, 40));
59
60    System.out.println(Collections.frequency(al4, 40));
61
62    System.out.println(Collections.frequency(al4, 40));
63
64    System.out.println(Collections.frequency(al4, 40));
65
66    System.out.println(Collections.frequency(al4, 40));
67
68    System.out.println(Collections.frequency(al4, 40));
69
70    System.out.println(Collections.frequency(al4, 40));
71
72    System.out.println(Collections.frequency(al4, 40));
73 }
```



13.12.2022 snippet session - Notepad

```
File Edit View
try {s2();}

}
catch (Exception e) { s += "c"; }

void s2() throws Exception {
    s3();
    s += "2";
    s3();
    s += "2b";
}

void s3() throws Exception {
    throw new Exception();
}
```

What is the result?

A. -
B. -c
C. -c2
D. -2c
E. -c22b
F. -2c2b
G. -2c2bc
H. Compilation fails

Answer: B

13.12.2022, Calif 1 20°C, mostly cloudy

Q. Search

Nitin Mangalath

99% Windows (CEP) UTF-8

~ ⌂ ⌂ ENG ⌂ ⌂ IN 13.12.2022

13.12.2022 (snippets_session - Notepad)

File Edit View

C. -c2
D. -2c
E. -c22b
F. -2c2b
G. -2c2bc
H. Compilation fails

Answer: B

Given:

```
try { int x = Integer.parseInt("two"); }
```

Which could be used to create an appropriate catch block? (Choose all that apply.)

A. ClassCastException
B. IllegalStateException
C. NumberFormatException
D. IllegalArgumentException
E. ExceptionInInitializerError
F. ArrayIndexOutOfBoundsException

answer: C,D

1



Ur 44 Col 1 99% Windows (CEP) | UTF-8

20°C Nitesh Mampath ENG ⌂ IN 13.12.2022 22:18

13.12.2022 - snippetsession - Notepad

File Edit View

```
1. class Loopy{  
2.     public static void main(String[] args){  
3.         int[] x = {7,6,5,4,3,2,1};  
4.         // insert code here  
5.         System.out.print(y + " ");  
6.     }  
7. }
```

Which, inserted independently at line 4, compiles? (Choose all that apply.)

A. for(int y : x) {
B. for(x : int y) {
C. int y = 0; for(y : x) {
D. for(int y=0, z=0; z<x.length; z++) { y = x[z];
E. for(int y=0, int z=0; z<x.length; z++) { y = x[z];
F. int y = 0; for(int z=0; z<x.length; z++) { y = x[z];

4

Un 65 Cpl 1
20°C
Mostly cloudy



100% Windows (CEP)

UTF-8

^ ⌂ ⇧ ENG ⇨ DE IN

13.12.2022 22:25



Q>
Given:

```
class Emu {
    static String s = "-";// -ic mc mf of
    public static void main(String[] args) {
        try {
            throw new Exception();
        }
        catch (Exception e) {
            try {
                throw new Exception();
            }
            catch (Exception ex) {
                s += "ic ";
            }
        }
        throw new Exception();
    }
}
```





```
13.12.2022_201955_GeForce_Notebook  
File Edit View  
  
{  
    throw new Exception();  
}  
catch (Exception e){  
    try  
    {  
        try  
        {  
            throw new Exception();  
        }  
        catch (Exception ex){  
            s+="lc";  
        }  
        throw new Exception();  
    }  
    catch (Exception x)  
    {  
        s+="mc";  
    }  
} finally  
  
16
```



Un 113.64.1

20°C
Mostly cloudy



100%

Windows (C:\EF7)

UTF-8

~ ⌂ ⌄ ENG ⌄ DE 22:33

IN 13.12.2022

```
13-12-2022 (Windows version) Notepad  
File Edit View  
}  
System.out.println(s);  
}  
}  
What is the result?  
A. -ic of  
B. -mf of  
C. -mc mf  
D. -ic mf of  
E. -ic mc mf of  
F. -ic mc of mf  
G. Compilation fails
```

Answer: E

→ 7

13-12-2022 (Windows version) Notepad
File Edit View
{
System.out.println(s);
}
}
}
What is the result?
A. -ic of
B. -mf of
C. -mc mf
D. -ic mf of
E. -ic mc mf of
F. -ic mc of mf
G. Compilation fails

13-12-2022 (Windows version) Notepad
File Edit View
{
System.out.println(s);
}
}
}

13-12-2022 (Windows version) Notepad
File Edit View
{
System.out.println(s);
}
}
}

13-12-2022 (Windows version) Notepad
File Edit View
{
System.out.println(s);
}
}
}



13-12-2022 (Windows version) Notepad
File Edit View
{
System.out.println(s);
}
}
}

11:14 AM

File Edit View

Given:

3. class SubException extends Exception {} //parentexception type
4. class SubSubException extends SubException {} //Childexception type
- 5.
6. public class CC { void doStuff() throws SubException {} } //Parent
- 7.
8. class CC2 extends CC { void doStuff() throws SubSubException {} } //Child
- 9.
10. class CC3 extends CC { void doStuff() throws Exception {} } //Child::CE(violate the rule of overriding)
- 11.
12. class CC4 extends CC { void doStuff(int x) throws Exception {} } //child
- 13.
14. class CC5 extends CC { void doStuff() {} } //Child

What is the result? (Choose all that apply)

- A. Compilation succeeds
- B. Compilation fails due to an error on line 8
- C. Compilation fails due to an error on line 10
- D. Compilation fails due to an error on line 12
- E. Compilation fails due to an error on line 14

Answer: C

39

Unsubscribe

20°C
Mostly cloudy

Q. Search

99% Windows CE/EF

UTF-8

ENG

IN

22:40

13/12/2022

Given:

```

3. public class Ebb {
4.     static int x = 7; // x = 7,8,9,10,11
5.     public static void main(String[] args) {
6.         String s = ""; // s = 9 10 10 d 13
7.         for(int y = 0; y < 3; y++) // y = 0,1,2,3
8.             x++;
9.         switch(x) {
10.             case 8: s += "8 ";
11.             case 9: s += "9 ";
12.             case 10: {s+= "10 ";break;}
13.             default: s += "d ";
14.             case 13: s+= "13 ";
15.         }
16.     }
17.     System.out.println(s);
18. }
19. static {x++;}
20. }

What is the result?
A. 9 10 d
B. 8 9 10 d
C. 9 10 10 d
D. 9 10 10 d 13

```

19

13.12.2022_snippets_session - Notepad

File Edit View

```
13.12.2022_snippets_session - Notepad
File Edit View
9.     switch(x) {
10.         case 8: s += "8 ";
11.         case 9: s += "9 ";
12.         case 10: { s+= "10 ";break; }
13.         default: s += "d ";
14.         case 13: s+= "13 ";
15.     }
16. }
System.out.println(s);
18. }
static {x++;}
20. }
```

What is the result?

A. 9 10 d

B. 8 9 10 d

C. 9 10 10 d

D. 9 10 10 d 13

E. 8 9 10 10 d 13

F. 8 9 10 9 10 10 d 13

G. Compilation fails

answer: D

RP

Ln 168 Col 1
20°C Cloudy

Q_Searh

Windows CE/RTI
UTF-8
^< > ENG ⌂ ⌂ 22:50
IN 13.12.2022

Given:

```
3. class Infinity {}  
4. public class Beyond extends Infinity {  
5.     static Integer i; // i = null  
6.     public static void main(String[] args){  
7.         int sw = (int)(Math.random() * 3);  
8.         switch(sw){  
9.             case 0: {for(int x = 10; x > 5; x++)  
10.                 if(x > 10000000) x = 10;  
11.                 break;}  
12.             case 1: {int v = 7 * i; break;}  
13.             case 2: {Infinity inf = new Beyond();  
14.             Beyond b = (Beyond)inf;  
15.         }  
16.     }  
17. }
```

And Given that line 7 will assign the value 0, 1, or 2 to sw, which are true? (Choose all that apply.)

- A. Compilation fails.
- B. A ClassCastException might be thrown.
- C. A StackOverflowError might be thrown.
- D. A NullPointerException might be thrown.
- E. An IllegalStateException might be thrown.

File Edit View
answer: D



22

13.12.2022, 19:09:51, Notepad

Q. Search

20°C

Cloudy

99%

Windows CE/RT

^

~

4

ENG

IN

22:57

13.12.2022

13.12.2022_snippetsession - Notepad

File Edit View

```
6. public static void main(String[] args){  
7.     int sw = (int)(Math.random() * 3);  
8.     switch(sw){  
9.         case 0: { for(int x = 10; x > 5; x++)  
10.             if(x > 10000000) x = 10;  
11.             break; }  
12.         case 1: { int y = 7 * i; break; }  
13.         case 2: { Infinity inf = new Beyond();  
14.             Beyond b = (Beyond)inf; }  
15.     }  
16. }  
17. }
```

And given that line 7 will assign the value 0, 1, or 2 to sw, which are true? (Choose all that apply.)

A. Compilation fails
B. A ClassCastException might be thrown
C. A StackOverflowError might be thrown
D. A NullPointerException might be thrown
E. An IllegalStateException might be thrown
F. The program might hang without ever completing
G. The program will always complete without exception

Answer: D,F

1



13.12.2022_snippetsession - Notepad

File Edit View

```
6. public static void main(String[] args){  
7.     int sw = (int)(Math.random() * 3);  
8.     switch(sw){  
9.         case 0: { for(int x = 10; x > 5; x++)  
10.             if(x > 10000000) x = 10;  
11.             break; }  
12.         case 1: { int y = 7 * i; break; }  
13.         case 2: { Infinity inf = new Beyond();  
14.             Beyond b = (Beyond)inf; }  
15.     }  
16. }  
17. }
```

And given that line 7 will assign the value 0, 1, or 2 to sw, which are true? (Choose all that apply.)

A. Compilation fails
B. A ClassCastException might be thrown
C. A StackOverflowError might be thrown
D. A NullPointerException might be thrown
E. An IllegalStateException might be thrown
F. The program might hang without ever completing
G. The program will always complete without exception

Answer: D,F

1



13.12.2022_snippetsession - Notepad

File Edit View

```
6. public static void main(String[] args){  
7.     int sw = (int)(Math.random() * 3);  
8.     switch(sw){  
9.         case 0: { for(int x = 10; x > 5; x++)  
10.             if(x > 10000000) x = 10;  
11.             break; }  
12.         case 1: { int y = 7 * i; break; }  
13.         case 2: { Infinity inf = new Beyond();  
14.             Beyond b = (Beyond)inf; }  
15.     }  
16. }  
17. }
```

And given that line 7 will assign the value 0, 1, or 2 to sw, which are true? (Choose all that apply.)

A. Compilation fails
B. A ClassCastException might be thrown
C. A StackOverflowError might be thrown
D. A NullPointerException might be thrown
E. An IllegalStateException might be thrown
F. The program might hang without ever completing
G. The program will always complete without exception

Answer: D,F

1