Keypoint in Exception Handling:

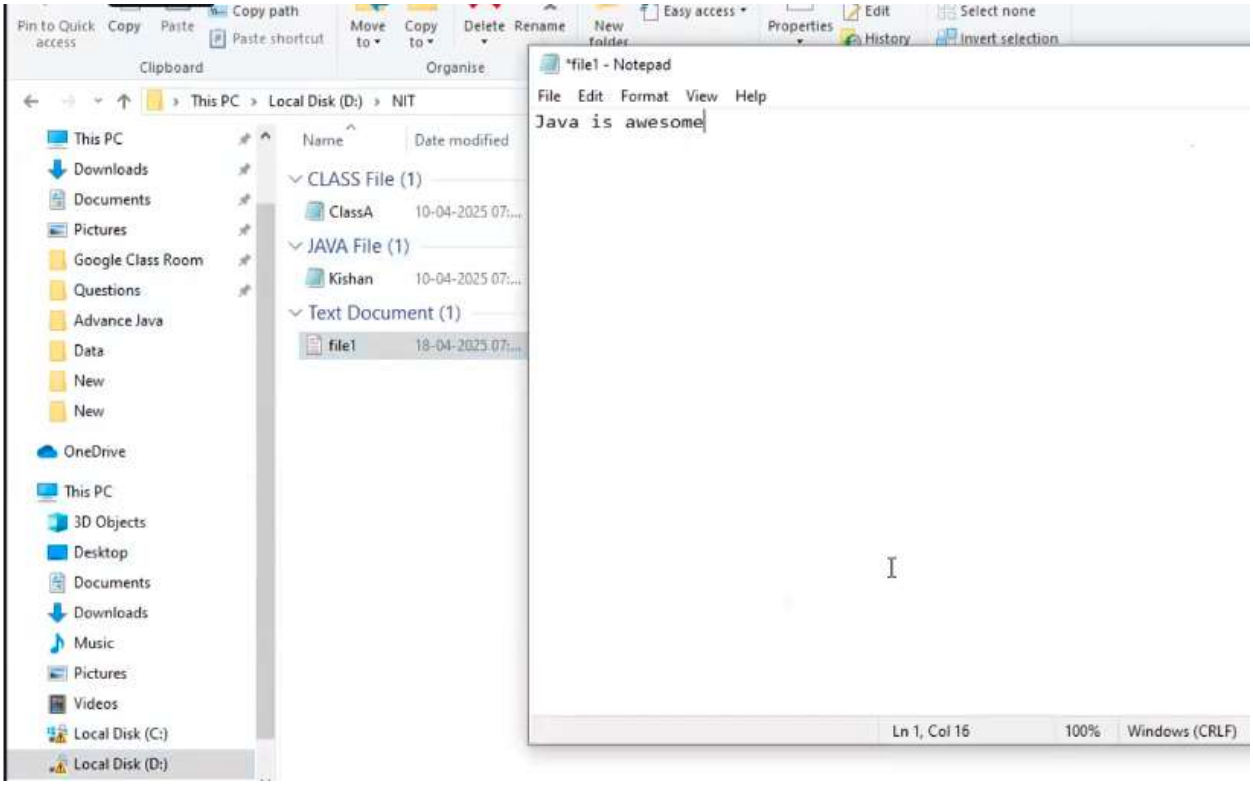
try-finally-catch ⇨ Invalid
finally-try-catch ⇨ Invalid
catch-try-finally ⇨ Invalid
try-catch-finally ⇨ Valid

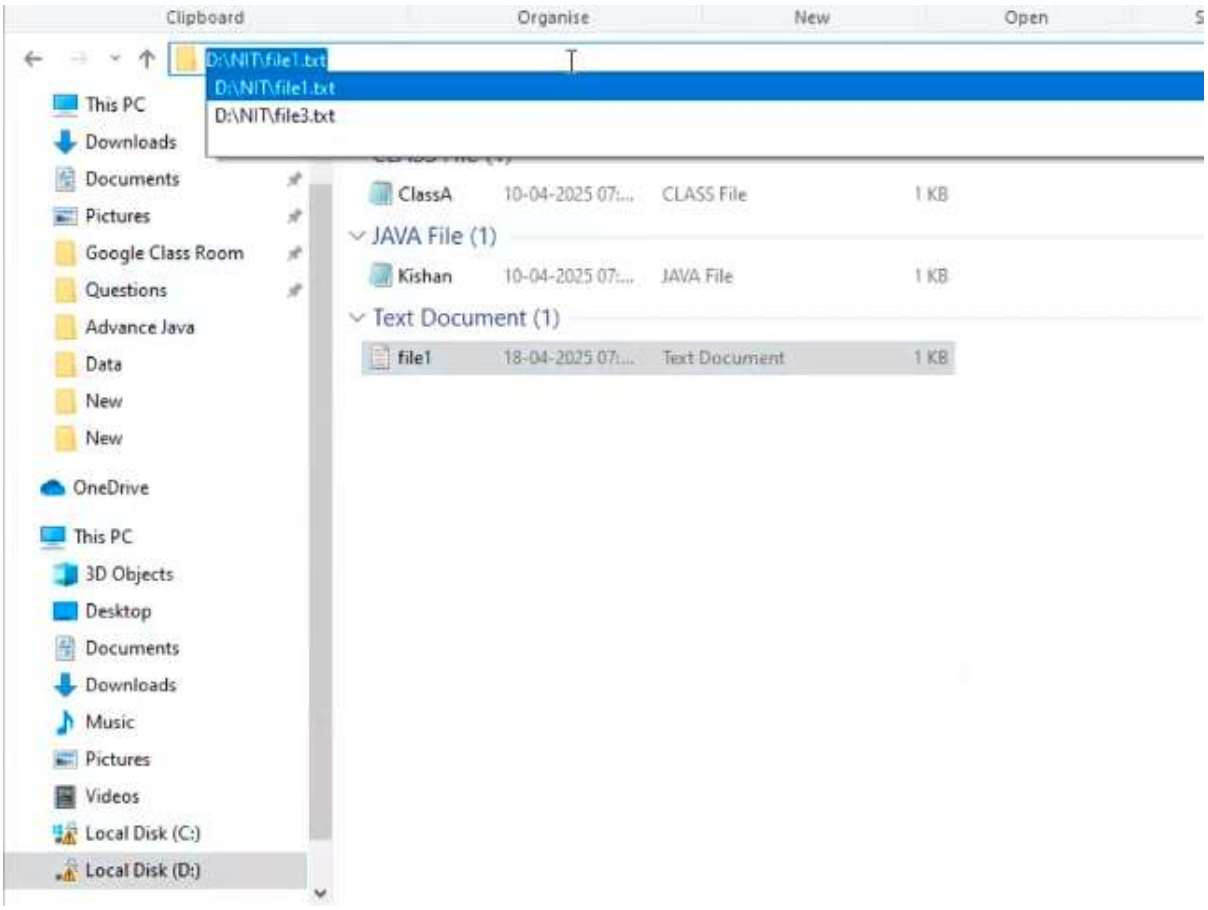
1) We can handle an exception by using try, catch, finally blocks.
2) Whenever we are using all the **three** blocks we should 100% maintain the order.
3) Inside try block always we need **to write minimum code**. [Write **only** suspecious code inside the try block]
4) If there is an exception occurred in the try block then immediately the compiler will be coming to **its respective catch block**. Remaining code which is present inside the try block will not be executed.
5)A catch block will be executed only if there is an exception occurred in the try block and we are catching that respective exception.
6) If we are catching the parent Exception of all the exception classes ie., **[Exception]** then every Exception will be handled.
7) **A single try block never exists.**
8) try block should be followed with either catch block (or) finally block (or) both.

try-catch ⇨ Valid | try-finally ⇨ Valid | catch-finally ⇨ Invalid | try-catch-finally ⇨ Valid | try ⇨ Invalid

9) If we are not writing catch block in our program then we will not be having any error but if there is an exception occurred in our program it will not be handled.
10) finally block is used to close the existing database/server connections.
11) Between **try-catch-finally** blocks there should not be any individual statements.
12) **We can handle multiple exceptions by using multiple catch blocks.**
13) For a single try block we can write multiple catch blocks but we need to write a single finally block
14) Multiple catch blocks are allowed but multiple finally blocks are not allowed.
15) **Whenever we are using multiple catch blocks always parent exception should be handled in the last catch block.**
16) If we are using multiple catch blocks duplicate exception handling is not allowed. [We will be getting an Compile time error]
17) From Java 1.7v onwards we can write a single try block also. try(Resources){ }
18) From Java 1.7v onwards we can handle multiple exceptions by using a single cacth block. But those exceptions should not have any parent child relation ship.

```
    void meth2()throws Exception
    {
        System.out.println("meth2() called");
        try(FileReader fr=new FileReader("filePath"))
        {
            System.out.println("Connection created");
        }
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```


Pin to Quick access
Copy
Paste
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
New folder
Organise
Easy access
Properties
Edit
History
Select none
Invert selection
This PC › Local Disk (D:) › NIT
This PC
Downloads
Documents
Pictures
Google Class Room
Questions
Advance Java
Data
New
New
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)
Name
Date modified
CLASS File (1)
ClassA
10-04-2025 07:...
JAVA File (1)
Kishan
10-04-2025 07:...
Text Document (1)
file1
18-04-2025 07:...
*file1 - Notepad
File Edit Format View Help
Java is awesome
Ln 1, Col 16
100%
Windows (CRLF)


Clipboard
Organise
New
Open
D:\NIT\file1.txt
D:\NIT\file1.txt
D:\NIT\file3.txt
This PC
Downloads
Documents
Pictures
Google Class Room
Questions
Advance Java
Data
New
New
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)
ClassA
10-04-2025 07:...
CLASS File
1 KB
JAVA File (1)
Kishan
10-04-2025 07:...
JAVA File
1 KB
Text Document (1)
file1
18-04-2025 07:...
Text Document
1 KB

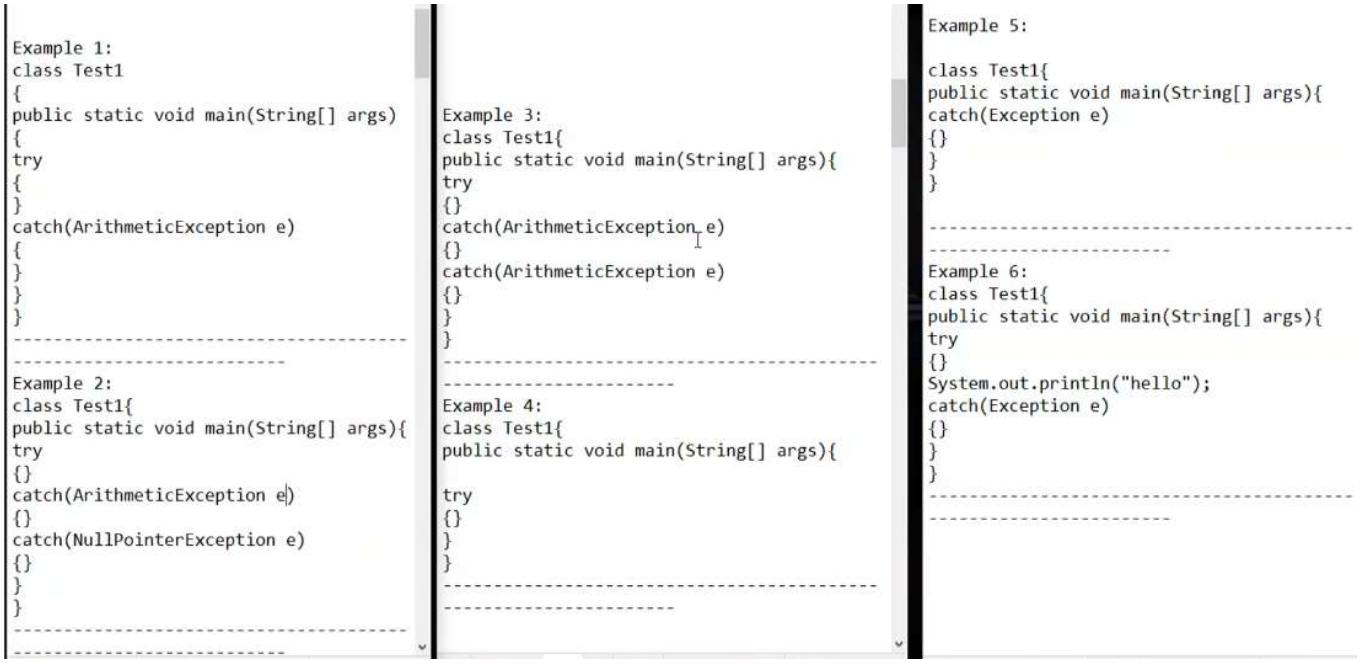
```
53      void meth2()throws Exception
54      {
55          System.out.println("meth2() called");
56          try(FileReader fr=new FileReader("D:\\NIT\\file1.txt"))
57          {
58              System.out.println("Connection created");
59          }
60      }
61      public static void main(String[] args)
62      {
63          ClassA aobj=new ClassA();
64          aobj.meth1();
65      }
66  }
```

```
46          finally
47          {
48              System.out.println("finally Block executed");
49              sc.close();
50          }
51          System.out.println(30);
52      }
53      void meth2()throws Exception
54      {
55          System.out.println("meth2() called");
56          try(FileReader fr=new FileReader("D:\\NIT\\file1.txt"))// Java 1.7v
57          {
58              System.out.println("Connection created");
59          }
60      }
61      public static void main(String[] args) throws Exception
62      {
63          ClassA aobj=new ClassA();
64          //aobj.meth1();
65          aobj.meth2();
66      }
67  }
```

```
meth2() called
Connection created
```

```java
import java.io.FileReader;
import java.util.Scanner;

public class ClassA
{
    void meth1()
    {
        System.out.println("meth1() called");
        Scanner sc=new Scanner(System.in);

        String arr[]=new String[5];
        arr[1]="Kishan";
        arr[2]="Java";

        System.out.println(10);
        try
        {
            System.out.println("try block executed");
            System.out.println("Enter a number");
            System.out.println("1)====>"+(20/sc.nextInt()));
            System.out.println("Enter index postion of array");
            System.out.println("2)===>"+arr[sc.nextInt()].toUpperCase());
            System.out.println("Hello world!!!");
        }
/*
        //System.out.println("hi"); // C.E
        catch(NullPointerException e)
        {
            System.out.println("1st catch Block executed");
            e.printStackTrace();
        }
        catch(Exception e)
        {
            System.out.println("2nd catch Block executed");
            e.printStackTrace();
        }
        //System.out.println("hi"); // C.E
*/
        catch(ArithmeticException | NullPointerException e) // Java 1.7v
        {
            System.out.println("catch block executed (Java 1.7v)");
            e.printStackTrace();
        }
        finally
        {
            System.out.println("finally Block executed");
            sc.close();
        }
        System.out.println(30);
```

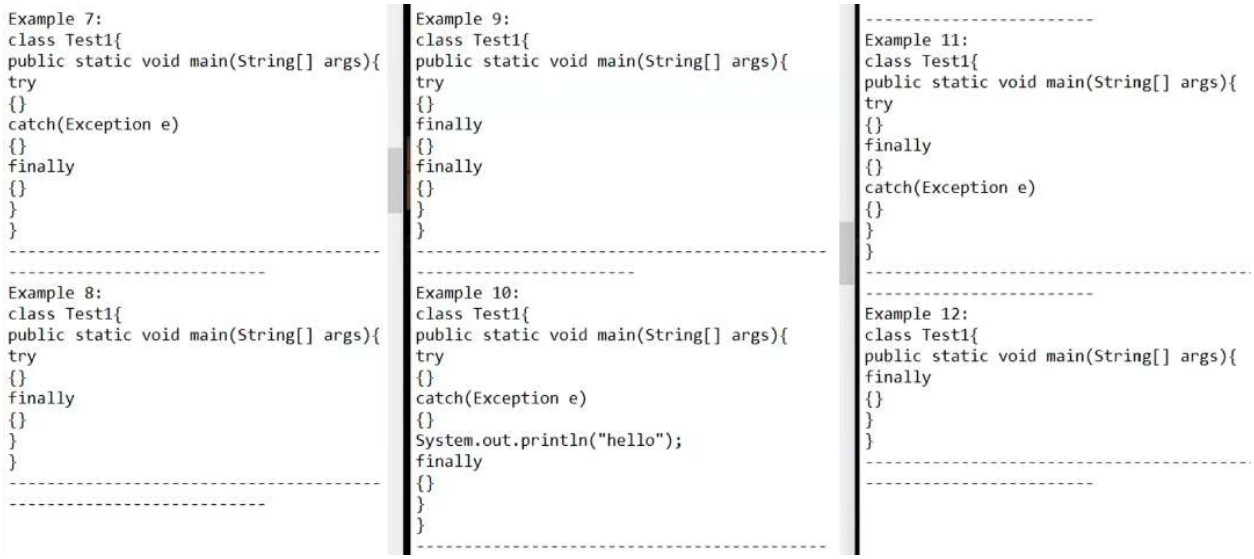
```
52      }
53      void meth2()throws Exception
54      {
55          System.out.println("meth2() called");
56          try(FileReader fr=new FileReader("D:\\NIT\\file1.txt"))// Java 1.7v
57          {
58              System.out.println("Connection created");
59          }
60      }
61      public static void main(String[] args) throws Exception
62      {
63          ClassA aobj=new ClassA();
64          //aobj.meth1();
65          aobj.meth2();
66      }
67  }
```



```
Example 1:
class Test1
{
public static void main(String[] args)
{
try
{
}
catch(ArithmeticException e)
{
}
}
}
-----------------------------------------
---------------------------
Example 2:
class Test1{
public static void main(String[] args){
try
{}
catch(ArithmeticException e)
{}
catch(NullPointerException e)
{}
}
}
-----------------------------------------
---------------------------
```

```
Example 3:
class Test1{
public static void main(String[] args){
try
{}
catch(ArithmeticException e)
{}
catch(ArithmeticException e)
{}
}
}
-----------------------------------------
----------------------
Example 4:
class Test1{
public static void main(String[] args){

try
{}
}
}
-----------------------------------------
----------------------
```

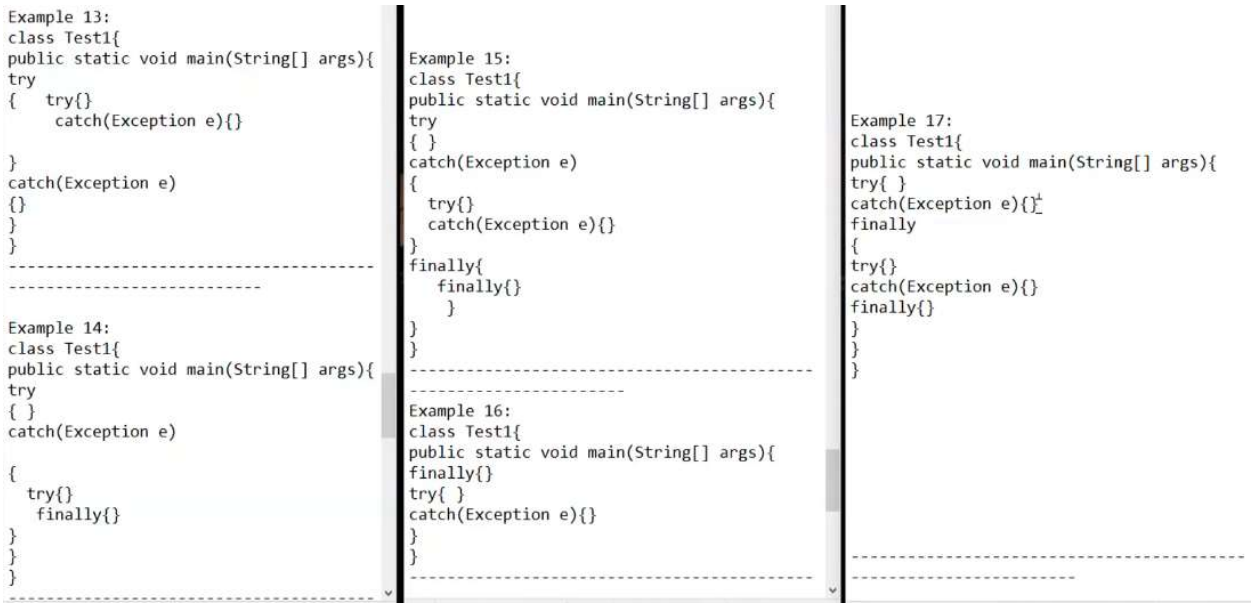
```
Example 5:

class Test1{
public static void main(String[] args){
catch(Exception e)
{}
}
}

-----------------------------------------
----------------------
Example 6:
class Test1{
public static void main(String[] args){
try
{}
System.out.println("hello");
catch(Exception e)
{}
}
}
-----------------------------------------
----------------------
```



```
Example 7:
class Test1{
public static void main(String[] args){
try
{}
catch(Exception e)
{}
finally
{}
}
}
-----------------------------------------
---------------------------
Example 8:
class Test1{
public static void main(String[] args){
try
{}
finally
{}
}
}
-----------------------------------------
---------------------------
```

```
Example 9:
class Test1{
public static void main(String[] args){
try
{}
finally
{}
finally
{}
}
}
-----------------------------------------
----------------------
Example 10:
class Test1{
public static void main(String[] args){
try
{}
catch(Exception e)
{}
System.out.println("hello");
finally
{}
}
}
-----------------------------------------
```

```
----------------------
Example 11:
class Test1{
public static void main(String[] args){
try
{}
finally
{}
catch(Exception e)
{}
}
}
-----------------------------------------
----------------------
Example 12:
class Test1{
public static void main(String[] args){
finally
{}
}
}
-----------------------------------------
----------------------
```



```
Example 13:
class Test1{
public static void main(String[] args){
try
{   try{}
     catch(Exception e){}

}
catch(Exception e)
{}
}
}
-----------------------------------------
---------------------------

Example 14:
class Test1{
public static void main(String[] args){
try
{ }
catch(Exception e)

{
  try{}
   finally{}
}
}
}
-----------------------------------------
```

```
Example 15:
class Test1{
public static void main(String[] args){
try
{ }
catch(Exception e)
{
  try{}
  catch(Exception e){}
}
finally{
   finally{}
    }
}
}
-----------------------------------------
-----------------------
Example 16:
class Test1{
public static void main(String[] args){
finally{}
try{ }
catch(Exception e){}
}
}
-----------------------------------------
```

```
Example 17:
class Test1{
public static void main(String[] args){
try{ }
catch(Exception e){}
finally
{
try{}
catch(Exception e){}
finally{}
}
}
}
-----------------------------------------
-----------------------
```

```
Example 1:    // Valid
class Test1
{
public static void main(String[] args)
{
try
{
}
catch(ArithmeticException e)
{
}
}
}
-----------------------------------------------
Example 2:      // Valid
class Test1{
public static void main(String[] args){
try
{}
catch(ArithmeticException e)
{}
catch(NullPointerException e)
{}
}
}
-----------------------------------------------
```

```
Example 3:     // In-Valid
class Test1{
public static void main(String[] args){
try
{}
catch(ArithmeticException e)
{}
catch(ArithmeticException e)
{}
}
}
-----------------------------------------------
Example 4:    // In-Valid
class Test1{
public static void main(String[] args){
try
{}
}
}
-----------------------------------------------
```

3.invalid because of two arithmetic Exception.

4.Invalid because there should not be single try block

But it is valid for try with resource.

```
Example 6:   // In-Valid
class Test1{
public static void main(String[] args){
try
{}
System.out.println("hello");
catch(Exception e)
{}
}
}
```

Because of statements between blocks

```
Example 7:   // Valid
class Test1{
public static void main(String[] args){
try
{}
catch(Exception e)
{}
finally
{}
}
}
```

```
Example 8:  // Valid
class Test1{
public static void main(String[] args){
try
{}
finally
{}
}
}
----------------------------------------
```

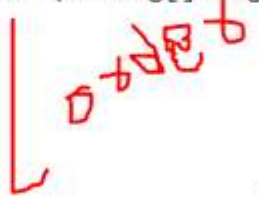
8.Valid, but if there is any exception occurred cannot be handled why because there is no catch block.

```
Example 9:    // In-Valid
class Test1{
public static void main(String[] args){
try
{}
finally
{}
finally
{}
}
}
```

9.invalid because multiple finally blocks are not allowed, but multiple catch blocks are allowed.

```
Example 10:   // In-Valid
class Test1{
public static void main(String[] args){
try
{}
catch(Exception e)
{}
System.out.println("hello");
finally
{}
}
}
```

```
Example 11:  // In-Valid
class Test1{
public static void main(String[] args){
try
{}
finally
{}
catch(Exception e)
{}
}
}
```



```
Example 12:   // In-Valid
class Test1{
public static void main(String[] args){
finally
{}
}
}
```

12.Invalid because single try, single catch, single final never exists.

```
Example 13: //  Valid
class Test1{
public static void main(String[] args){
try
{   try{}
     catch(Exception e){}
}
catch(Exception e)
{}
}
}
```

```
Example 14:   // Valid
class Test1{
public static void main(String[] args){
try
{ }
catch(Exception e)
{
  try{}
   finally{}
}
}
}
```

```
Example 15:   // In-Valid
class Test1{
public static void main(String[] args){
try
{ }
catch(Exception e)
{
  try{}
  catch(Exception e){}
}
finally{
   finally{}
    }
}
}
```

```
Example 16:  // In-Valid
class Test1{
public static void main(String[] args){
finally{}
try{ }
catch(Exception e){}
}
}
----------------------------------------
```

16.Invalid because of order

```
Example 17:    // Valid
class Test1{
public static void main(String[] args){
try{ }
catch(Exception e){}
finally
{
try{}
catch(Exception e){}
finally{}
}
}
}
```

17.valid