**Exception interview Question 26 .**

Exception **Output** interview question **1.**

|  |
| --- |
| **public** **class** MyClass {  **static** String *str* = "a";  **public** **static** **void** main(String[] args) {  **new** MyClass().method1();            System.*out*.println(*str*);     }  **void** method1() {  **try** {                   method2();            } **catch** (Exception e) {  *str* += "b";            }     }  **void** method2() **throws** Exception {  **try**{                   method3();  *str* += "c";            }**catch**(Exception e){  **throw** **new** Exception();            }**finally**{  *str* += "d";            }            method3();  *str* += "e";     }  **void** method3() **throws** Exception {  **throw** **new** Exception();     }  } |

**Answer**.  adb

**Exception interview Question 27.**

Exception **Output** interview question **2.**

|  |
| --- |
| /\*\* Copyright (c), AnkitMittal JavaMadeSoEasy.com \*/  **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) {  *m*(); //call recursive method m()            System.*out*.println("Code after exception handling");     }    **static** **void** m() {  **try** {  *m*();            } **catch** (StackOverflowError e) {                   e.printStackTrace();            }     }  } |

**Answer**. method m() calls itself recursively so [StackOverflowError](http://www.javamadesoeasy.com/2015/05/javalangerror-in-exception-handling-in.html) will be thrown in java.

Output=

java.lang.StackOverflowError

   at ExceptionTest.m(ExceptionTest.java:10)

   .

   .

   .

   .

   .

   .

   .

   .

Code after exception handling

**Exception interview Question 28.**

Exception **Output** interview question **3.**

|  |
| --- |
| **/\*\* Copyright (c), AnkitMittal JavaMadeSoEasy.com \*/**  **public class ExceptionTest {**  **public static void main(String[] args) {**  **int i=10/0;**  **System.*out*.println("Did this line execute?");**  **}**  **}** |

**Answer**. **int i=10/0;  will throw ArithmeticException in java**

Output =

**Exception in thread "main" java.lang.ArithmeticException: / by zero**

**At ExceptionTest.main(ExceptionTest.java:4)**

**Exception interview Question 29.**

Exception **Output** interview question **4.**

|  |
| --- |
| **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) {    **try**{  **int** i=10/0;            }**catch**(Exception e){                   System.*out*.println("Exception handled  properly in catch block");            }            System.*out*.println("Code after exception handling");     }  } |

**Answer**.

**int** i=10/0; will throw ArithmeticException, Exception is superclass of ArithmeticException, so catch block will handle ArithmeticException in java.

Output=

Exception handled  properly

Code after exception handling

[*finally*](http://www.javamadesoeasy.com/2015/05/try-catch-finally-block-in-java.html) *related output questions >*

**Exception interview Question 30.**

Exception **Output** interview question **5.**

|  |
| --- |
| **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) {    **try**{  **int** i=10/0; //will throw ArithmeticException            }**catch**(ArithmeticException e){                   System.*out*.println("ArithmeticException handled in catch block");            }  **finally**{                   System.*out*.println("finally block executed");            }            System.*out*.println("code after try-catch-finally block");     }  } |

**Answer**. **int** i=10/0; will throw ArithmeticException, Exception is superclass of ArithmeticException, so catch block will handle ArithmeticException and [finally](http://www.javamadesoeasy.com/2015/05/finally-block-in-java.html) is always executed in java.

OUTPUT =

ArithmeticException handled in catch block

finally block executed

code after try-catch-finally block

**Exception interview Question 31.**

Exception **Output** interview question **6.**

|  |
| --- |
| **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) {  **try**{                   System.*out*.println("in try block");                   System.*exit*(0);            }**finally**{                   System.*out*.println("finally block executed");            }     }  } |

**Answer**. finally block is not executed when System.exit is called in java.

OUTPUT =

in try block

**Exception interview Question 32.**

Exception **Output** interview question **7.**

|  |
| --- |
| **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) {  **try**{  **int** i=10/0;            }**catch**(IndexOutOfBoundsException e){                   System.*out*.println("IndexOutOfBoundsException handled in catch block");            }  **finally**{                   System.*out*.println("finally block executed");            }            System.*out*.println("code after try-catch-finally block");     }  } |

**Answer**. **int** i=10/0; will throw ArithmeticException, IndexOutOfBoundsException is not super class of ArithmeticException, so catch block won’t be able to handle ArithmeticException but [finally](http://www.javamadesoeasy.com/2015/05/finally-block-in-java.html) is always executed in java.

OUTPUT =

finally block executed

Exception in thread "main" java.lang.ArithmeticException: / by zero

   at ExceptionTest.main(ExceptionTest.java:4)

**Exception interview Question 33.**

Exception **Output** interview question **8.**

|  |
| --- |
| **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) {            System.*out*.println("method return -> "+*m*());     }    **static** String m(){  **try**{  **int** i=10/0;            }**catch**(ArithmeticException e){  **return** "catch";            }**finally**{  **return** "finally";            }       }  } |

**Answer**. In above program, i=10/0 will throw ArithmeticException and enter catch block to return "catch", but ultimately control will enter finally block to return "finally" in java.

OUTPUT =

method return -> finally

[*Multiple exception handling*](http://www.javamadesoeasy.com/2015/05/multiple-catch-block-in-java.html) *related output questions >*

**Exception handling interview Question 34.**

Exception **Output** interview question **9.**

|  |
| --- |
| public class ExceptionTest {     public static void main(String[] args) {              try{                   int i=10/0;            }**catch**(**ArithmeticException** ae){                   System.*out*.println("Exception handled - ArithmeticException");            }**catch**(**RuntimeException** re){                   System.*out*.println("Exception handled - RuntimeException");            }**catch**(**Exception** e){                   System.*out*.println("Exception handled - Exception");            }     }  } |

**Answer**. Yes, program will compile successfully in java.

In the above above >

i=10/0 will throw ArithmeticException and will be handled in first catch block.

**ArithmeticException** has been used in **first** catch block

**RuntimeException** has been used in **second** catch block

**Exception** has been used in **third** catch block

**Exception** is superclass of **RuntimeException** and

**RuntimeException** is superclass of **ArithmeticException in java.**

OUTPUT =

Exception handled - ArithmeticException

**Exception handling interview Question 35.**

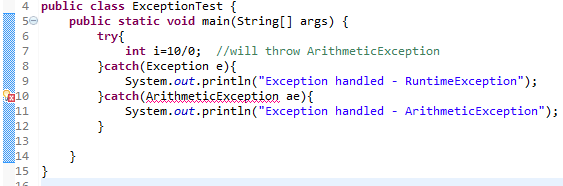
Exception **Output** interview question **10.**

|  |
| --- |
| **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) {  **try**{  **int** i=10/0;            }**catch**(Exception e){                   System.*out*.println("Exception handled - RuntimeException");            }**catch**(ArithmeticException ae){                   System.*out*.println("Exception handled - ArithmeticException");            }     }  } |

**Answer**. No, program will not compile.

**Exception** is superclass of **ArithmeticException.** Exception class handled in starting catch block must be subclass of Exception class handled in following catch blocks (otherwise we will face compilation error) in java.

In above program we will compilation error at line 10



**Exception handling interview Question 36.**

Exception **Output** interview question **11.**

|  |
| --- |
| **public** **class** MyClass {  **static** String *s* = "";    **public** **static** **void** main(String[] args) {            //try-catch-finally  **try** {  **throw** **new** Exception();            } **catch** (Exception e) {                   //1st nested try-catch-finally  **try** {                         //2nd nested try-catch-finally  **try** {  **throw** **new** Exception();                         } **catch** (Exception ex) {  *s* += "a";                         } **finally**{  *s* += "b";                         }  **throw** **new** Exception();                   } **catch** (Exception x) {  *s* += "c";                   } **finally** {  *s* += "d";                   }            } **finally** {  *s* += "e";            }            System.*out*.println(*s*);     }  } |

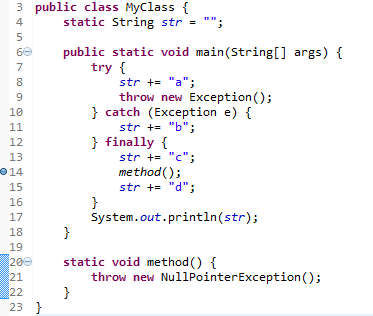
**Answer**.

OUTPUT in java =

abcde

**Exception handling interview Question 37.**

Exception **Output** interview question 12 **.**

****

**Answer**. Exception is thrown at line 14, at that time value of str is “abc”

Output in java=

Exception in thread "main" java.lang.NullPointerException

   at MyClass.method(MyClass.java:21)

   at MyClass.main(MyClass.java:14)

**Exception handling interview Question 38.**

Exception **Output** interview question **13.**



**Answer**. Yes, program will compile because **UserDefinedException is RuntimeException** and we are **free not to handle Runtime exceptions in java**.

Output in java=

Exception in thread "main" UserDefinedException: user defined exception

   at UserDefinedExceptionTest.main(UserDefinedExceptionTest.java:16)

[*Multi catch syntax*](http://www.javamadesoeasy.com/2015/05/catch-block-and-automatic-resource.html) *related output questions  in java >*

**Exception handling interview Question 39.**

Exception **Output** interview question **14.**

**will this program compile?**

|  |
| --- |
| **import** java.io.IOException;  /\*\* Copyright (c), AnkitMittal JavaMadeSoEasy.com \*/  **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) {  **try**{  **throw** **new** IOException();            }**catch**(IOException | Exception ex){                   System.*out*.println(ex + " handled ");            }     }  } |

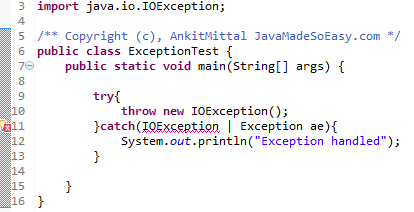
**Answer**. No, program will not compile in java.

Multi catch syntax have been used in above program,

IOException is subclass of Exception in java.

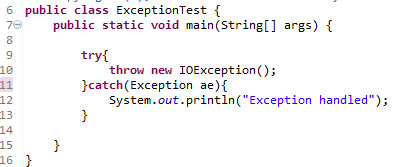
if **multi catch syntax** is used to catch subclass and its superclass than compilation error will be thrown.

IOException and Exception in **multi catch syntax** will cause compilation error “The exception **IOException** is already caught by the alternative **Exception**”.



**Solution >**

We must use only **Exception** to catch its subclass like this >



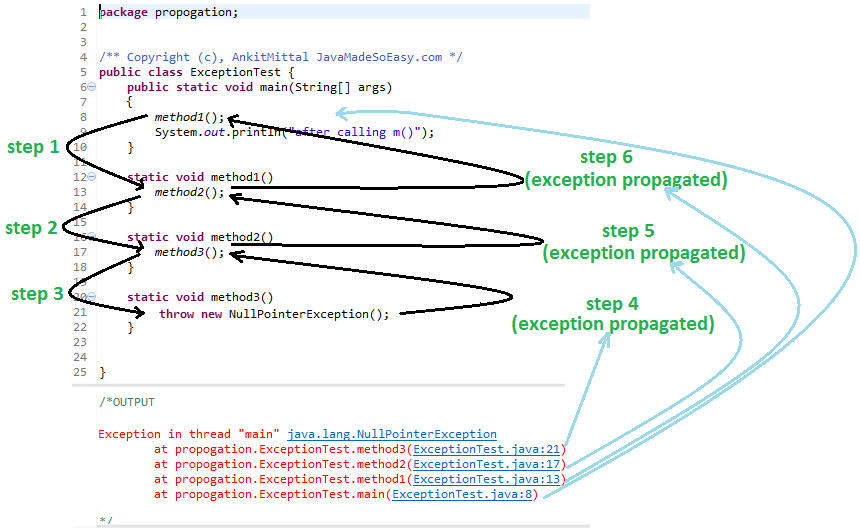
[*Propagating checked and unchecked exceptions*](http://www.javamadesoeasy.com/2015/05/exception-propagation-in-java-deep.html) *related output questions in java >*

**Exception handling interview Question 40.**

Exception **Output** interview question **15.**

|  |
| --- |
| **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args)     {  *method1*();            System.*out*.println("after calling m()");     }    **static** **void** method1(){  *method2*();     }    **static** **void** method2(){  *method3*();     }    **static** **void** method3(){  **throw** **new** NullPointerException();     }      } |

**Answer**. [***unchecked***](http://www.javamadesoeasy.com/2015/05/checked-compile-time-exceptions-and.html)exceptions are **automatically propagated** in java.

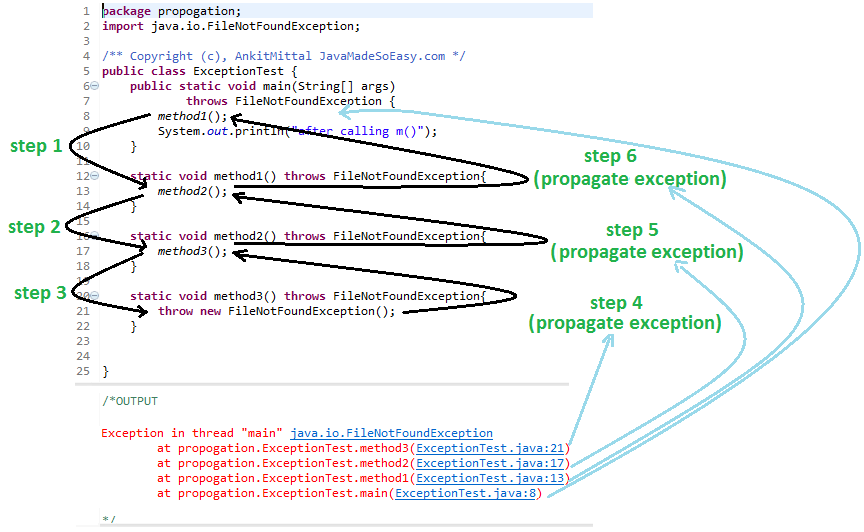


**Exception handling interview Question 41.**

Exception **Output** interview question **16.**

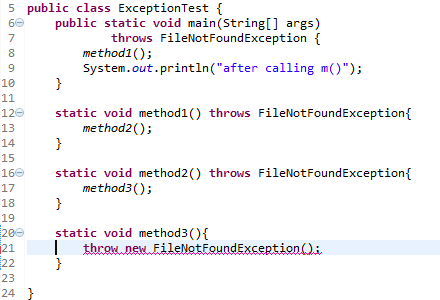
|  |
| --- |
| **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args)  **throws** FileNotFoundException {  *method1*();            System.*out*.println("after calling m()");     }    **static** **void** method1() **throws** FileNotFoundException{  *method2*();     }    **static** **void** method2() **throws** FileNotFoundException{  *method3*();     }    **static** **void** method3() **throws** FileNotFoundException{  **throw** **new** FileNotFoundException();     }    } |

**Answer**. For [**propagating checked** exceptions](http://www.javamadesoeasy.com/2015/05/exception-propagation-in-java-deep.html) method must throw exception by using [**throws**](http://www.javamadesoeasy.com/2015/05/throws-exception-in-java.html)keyword in java.



**Exception handling interview Question 42 .**

Exception **Output** interview question **17.**



**Answer**. Compilation of program will fail at line 21 because for **propagating checked** exceptions method must throw exception by using [**throws**](http://www.javamadesoeasy.com/2015/05/throws-exception-in-java.html)keyword in java.

[*try-with-resource*](http://www.javamadesoeasy.com/2015/05/try-with-resources-in-java.html) *related output questions in java >*

**Exception handling interview Question 43.**

Exception **Output** interview question **18.**

|  |
| --- |
| **import** java.io.FileInputStream;  **import** java.io.IOException;  **import** java.io.InputStream;  /\*\* Copyright (c), AnkitMittal JavaMadeSoEasy.com  \* Main class \*/  **public** **class** TryWithResourseTest {  **public** **static** **void** main(String[] args) **throws** IOException {  **try** (InputStream inputStream = **new** FileInputStream("c:/txtFile.txt")) {                   //code...            }     }  } |

**Answer**. Above program will execute properly provided file is found at specified directory. In java 7, using [Try-with-resources](http://www.javamadesoeasy.com/2015/05/try-with-resources-in-java.html)we need not to write **explicit code for closing file in java**.

Now, question comes why we need not to close file when we are using **Try-with-resources**?

Because **FileInputStream** implements java.lang.**AutoCloseable** **interface** (**AutoCloseable** interface’s close method automatically closes resources which are no longer needed.) in java.

**Exception handling interview Question 44.**

Exception **Output** interview question **19.**

|  |
| --- |
| **import** java.io.BufferedInputStream;  **import** java.io.FileInputStream;  **import** java.io.IOException;  **import** java.io.InputStream;  /\*\* Copyright (c), AnkitMittal JavaMadeSoEasy.com  \* Main class \*/  **public** **class** TryWithResourseTest {  **public** **static** **void** main(String[] args) **throws** IOException {  **try** (InputStream inputStream = **new** FileInputStream("c:/txtFile.txt") ;                   InputStream bInputStream = **new** BufferedInputStream(inputStream) ) {                   //code...            }     }  } |

**Answer**. Above program will execute properly provided file is found at specified directory, **Try-with-resources** allows us to use multiple resources inside it, all that we need to do is separate resources by semicolon (;)

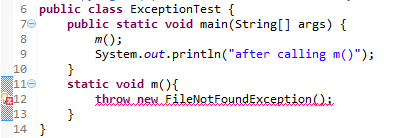
[*throw*](http://www.javamadesoeasy.com/2015/05/throw-exception-in-java.html) *and* [*throws*](http://www.javamadesoeasy.com/2015/05/throws-exception-in-java.html) *related output questions in java >*

**Exception handling interview Question 45.**

Exception **Output** interview question **20.**

|  |
| --- |
| **import** java.io.FileNotFoundException;  **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) {  *m*();            System.*out*.println("after calling m()");     }  **static** **void** m(){  **throw** **new** FileNotFoundException();     }  } |

**Answer**. If checked Exception is not handled either by try-catch or throws, we will face compilation error in java.



**Exception handling interview Question 46.**

Exception **Output** interview question **21.**

|  |
| --- |
| **import** java.io.FileNotFoundException;  /\*\* Copyright (c), AnkitMittal JavaMadeSoEasy.com \*/  **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) {  *m*();            System.*out*.println("after calling m()");     }  **static** **void** m(){  **try {**  **throw new FileNotFoundException();**  **} catch (FileNotFoundException e) {**  **System.*out*.println("FileNotFoundException handled in try-catch block");**  **}**     }  } |

**Answer**. In above program, We throwed **FileNotFoundException** (checked exception) by using **throw** keyword and handled it in try-catch block in java.

OUTPUT =

FileNotFoundException handled in try-catch block

after calling m()

**Exception handling interview Question 47.**

Exception **Output** interview question **22.**

|  |
| --- |
| **import** java.io.FileNotFoundException;  /\*\* Copyright (c), AnkitMittal JavaMadeSoEasy.com \*/  **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) {  **try {**  ***m*();**  **} catch (FileNotFoundException e) {**  **System.*out*.println("FileNotFoundException handled in try-catch block");**  **}**            System.*out*.println("after calling m()");     }  **static** **void** m() **throws** FileNotFoundException{  **throw** **new** FileNotFoundException();     }  } |

**Answer**. In the above program, method m() [**propagated** exception](http://www.javamadesoeasy.com/2015/05/exception-propagation-in-java-deep.html) to calling method (i.e. main method) using **throws in java**.

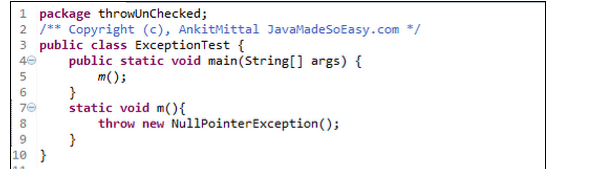
OUTPUT of program =

FileNotFoundException handled in try-catch block

after calling m()

**Exception handling interview Question 48.**

Exception **Output** interview question **23.**



**Answer**.

We throw NullPointerException (unChecked exception) and didn’t handled it, no compilation error was thrown.

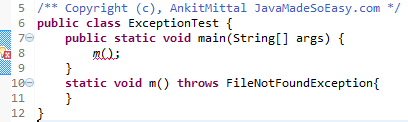
We need **not to handle** unChecked exception either by catching it or throwing it in java.

Output of program -

https://lh5.googleusercontent.com/zeQcBKW6Dq1FLF_--3XydVNPZbuEbSW38pCz-FxyoouNvsxbCD4cJlto4hu9Y7MZUqmORCGtGT2AUXlD7g3VNBCTsyGCmyo5mEjWvUaH3oNk0st-RNUIyZ67W57ltZTOGvb5xYc

**Exception handling interview Question 49.**

Exception **Output** interview question **24.**



**Answer**. We need **to handle** checked exception either by catching it or throwing it further, if not handled we will face compilation error at line 8.

**Exception handling interview Question 50.**

Exception **Output** interview question **25.**

|  |
| --- |
| **import** java.io.IOException;  **import** java.sql.SQLException;  /\*\* Copyright (c), AnkitMittal JavaMadeSoEasy.com \*/  **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) {  **try** {  *m*();                   System.*out*.print("a");            } **catch** (Exception e) {                   System.*out*.print("b");            } **finally**{                   System.*out*.print("c");            }     }  **static** **void** m() **throws** IOException, SQLException{  **int** i=1;  **if**(i==1)  **throw** **new** IOException();  **else**  **throw** **new** SQLException();     }  } |

**Answer**.

Output of program =

bc

**Exception handling interview Question 51.**

Exception **Output** interview question **26.**

|  |
| --- |
| **class** SuperClass{  **void** method() **throws NullPointerException**{            System.*out*.println("superClass method");     }  }  **class** SubClass **extends** SuperClass{  **void** method() **throws RuntimeException**{            System.*out*.println("SubClass method");     }  }  /\*\* Copyright (c), AnkitMittal JavaMadeSoEasy.com  \* Main class \*/  **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) {            SuperClass obj=**new** SubClass();            obj.method();     }  } |

**Answer**. *If superclass method does* ***not throw****/declare any* ***exception -*** [*overridden method of subclass*](http://www.javamadesoeasy.com/2015/05/throwdeclare-checked-and-unchecked.html)***can*** *declare/****throw any unchecked /RuntimeException (superclass or subclass) in java.***

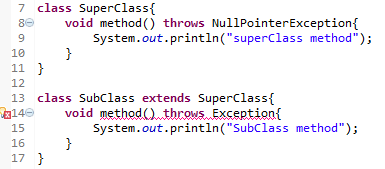
RuntimeException is superclass of NullPointerException.

Output of program =

SubClass method

**Exception handling interview Question 52.**

Exception **Output** interview question **27.**



**Answer**. *If superclass method does* ***not throw****/declare any* ***exception*** *- overridden method of subclass* ***cannot*** *declare/****throw******any checked exception in java.***

Any attempt to throw checked exception in overridden method of subclass will cause compilation error.

**Exception handling interview Question 53.**

Exception **Output** interview question **28.**

|  |
| --- |
| **import** java.io.FileNotFoundException;  **import** java.io.IOException;  **class** SuperClass{  **void** method() **throws IOException**{            System.*out*.println("superClass method");     }  }  **class** SubClass **extends** SuperClass{  **void** method() **throws FileNotFoundException**{            System.*out*.println("SubClass method");     }  }  /\*\* Copyright (c), AnkitMittal JavaMadeSoEasy.com  \* Main class \*/  **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) **throws** Exception {            SuperClass obj=**new** SubClass();            obj.method();     }  } |

**Answer**. *If superclass method throws/declare* ***checked****/****compileTime******exception -*** *overridden method of subclass* ***can*** *declare/****throw******narrower*** *(subclass of)* ***checked exception in java.***

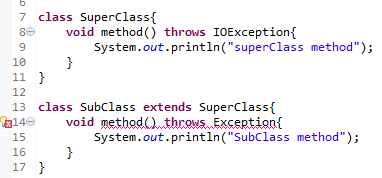
IOException is superclass of FileNotFoundException.

Output of program =

SubClass method

**Exception handling interview Question 54.**

Exception **Output** interview question **29.**

****

**Answer**.

*If superclass method throws/declare* ***checked****/****compileTime******exception -*** *overridden method of subclass* ***cannot*** *declare/****throw******broader*** *(superclass of)* ***checked exception in java.***

Any attempt to throw broader (superclass of) checked exception in overridden method of subclass will cause compilation error.

Exception is superclass of IOException in java.

**Exception handling interview Question 55.**

Exception **Output** interview question **30.**

|  |
| --- |
| **import** java.io.IOException;  **class** SuperClass{  **void** method() **throws IOException**{            System.*out*.println("superClass method");     }  }  **class** SubClass **extends** SuperClass{  **void** method() **throws NullPointerException**{            System.*out*.println("SubClass method");     }  }  /\*\* Copyright (c), AnkitMittal JavaMadeSoEasy.com  \* Main class \*/  **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) **throws** Exception {            SuperClass obj=**new** SubClass();            obj.method();     }  } |

**Answer**. *If superclass method throws/declare* ***checked****/****compileTime******exception******-*** *overridden method of subclass* ***can*** *declare/****throw any unchecked /RuntimeException in java.***

Output of program =

SubClass method

**Exception handling interview Question 56.**

Exception **Output** interview question **31.**

|  |
| --- |
| **class** SuperClass{  **void** method(){            System.*out*.println("superClass method");     }  }  **class** SubClass **extends** SuperClass{  **void** method() **throws NullPointerException**{            System.*out*.println("SubClass method");     }  }  /\*\* Copyright (c), AnkitMittal JavaMadeSoEasy.com  \* Main class \*/  **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) **throws** Exception {            SuperClass obj=**new** SubClass();            obj.method();     }  } |

**Answer**. *If superclass method throws/declare* ***unchecked/RuntimeException******-*** *overridden method of subclass* ***can*** *declare/****throw any unchecked /RuntimeException in java.***

Output of program =

SubClass method

**Exception handling interview Question 57.**

Exception **Output** interview question **32.**

Is it a valid method overriding program and if yes what will be output of program?

|  |
| --- |
| **class** SuperClass{  **void** method(){            System.*out*.println("superClass method");     }  }  **class** SubClass **extends** SuperClass{  **void** method() {            System.*out*.println("SubClass method");     }  }  /\*\* Copyright (c), AnkitMittal JavaMadeSoEasy.com  \* Main class \*/  **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) **throws** Exception {            SuperClass obj=**new** SubClass();            obj.method();     }  } |

**Answer**. *Yes,* It’s a valid method overriding program.

/\*OUTPUT of program

SubClass method

\*/

*If superclass method does* ***not throw****/declare any* ***exception*** *then* overridden method of subclass **may not** declare/**throw any exception** in java.

**Exception handling interview Question 58.**

Exception **Output** interview question **33.**

Is it a valid method overriding program and if yes what will be output of program?

|  |
| --- |
| **import** java.io.FileNotFoundException;  **import** java.io.IOException;  **class** SuperClass{  **void** method() **throws IOException**{            System.*out*.println("superClass method");     }  }  **class** SubClass **extends** SuperClass{  **void** method(){            System.*out*.println("SubClass method");     }  }  /\*\* Copyright (c), AnkitMittal JavaMadeSoEasy.com  \* Main class \*/  **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) **throws** Exception {            SuperClass obj=**new** SubClass();            obj.method();     }  } |

**Answer**. *Yes,* It’s a valid method overriding program.

/\*OUTPUT of program

SubClass method

\*/

*If superclass method throws/declare* ***checked****/****compileTime******exception*** *then* overridden method of subclass **may not** declare/**throw any exception** in java.

**Exception handling interview Question 59.**

Exception **Output** interview question **34.**

Is it a valid method overriding program and if yes what will be output of program?

|  |
| --- |
| **class** SuperClass{  **void** method() **throws NullPointerException**{            System.*out*.println("superClass method");     }  }  **class** SubClass **extends** SuperClass{  **void** method(){            System.*out*.println("SubClass method");     }  }  /\*\* Copyright (c), AnkitMittal JavaMadeSoEasy.com  \* Main class \*/  **public** **class** ExceptionTest {  **public** **static** **void** main(String[] args) {            SuperClass obj=**new** SubClass();            obj.method();     }  } |

**Answer**. *Yes,* It’s a valid method overriding program.

/\*OUTPUT of program

SubClass method

\*/

*If superclass method throws/declare* ***unchecked/RuntimeException***overridden method of subclass **may not** declare/**throw any exception** in java**.**